Guida per gli sviluppatori

AWS IoT Core

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS IoT Core: Guida per gli sviluppatori

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Che cos'è AWS IoT? 1
Come accedono i tuoi dispositivi e le tue app AWS IoT 2
Cosa si può fare AWS IoT 2
IoT nel settore industriale 3
IoT nell’automazione domestica 3
Come AWS IoT funziona 4
L'universo IoT 4
AWS IoT panoramica dei servizi 7
AWS IoT Core servizi 12
Scopri di più su AWS IoT 16
Risorse di formazione per AWS IoT 16
AWS IoT risorse e guide 16
AWS IoT nei social media 17
AWS servizi utilizzati dal motore delle AWS IoT Core regole 17
Protocolli di comunicazione supportati da AWS IoT Core 19
Cosa c'è di nuovo nella AWS IoT console 19
Legenda 22
Lavorare con AWS SDKs 23
Tutorial introduttivi 25
Connect il tuo primo dispositivo a AWS IoT Core 25
Configurare Account AWS 27
Registrati per un Account AWS 27
Crea un utente con accesso amministrativo 28
Apri la console AWS IoT 29
Tutorial interattivo 29
Collegamento di dispositivi IoT 31
Salvataggio dello stato del dispositivo non in linea 31
Instradamento dei dati del dispositivo ai servizi 32
Tutorial di connessione rapida 33
Passaggio 1. Inizia il tutorial 34
Passaggio 2. Crea un oggetto 35
Fase 3. Scarica file sul tuo dispositivo 38
Passaggio 4. Esegui l'esempio 41
Fase 5. Esplora ulteriormente 44

Verifica la connettività ... 45
Tutorial di connessione avanzato ... 50
Quale opzione di dispositivo è la migliore per te? ... 52
Crea AWS IoT risorse ... 52
Configurazione del dispositivo ... 57
Visualizza i messaggi MQTT con il AWS IoT client MQTT ... 97
Visualizzazione dei messaggi MQTT nel client MQTT ... 97
Pubblicazione di messaggi MQTT dal client MQTT ... 100
Test delle sottoscrizioni condivise nel client MQTT ... 102
AWS IoT tutorial ... 104
Creazione di demo con AWS IoT Device Client ... 104
Prerequisiti per creare demo con Device Client AWS IoT ... 105
Preparazione all'utilizzo di IoT Device Client ... 108
Installazione e configurazione del client per dispositivi IoT ... 122
Comunica con il client del dispositivo utilizzando MQTT ... 135
Esegui lavori IoT con Device Client ... 155
Pulizia ... 170
Creazione di soluzioni con il AWS IoT dispositivo SDKs ... 180
Inizia a creare soluzioni con il dispositivo AWS IoT SDKs ... 180
Connessione di un dispositivo AWS IoT Core tramite AWS IoT Device SDK ... 181
Creazione di AWS IoT regole per instradare i dati del dispositivo verso altri servizi ... 206
Mantenimento dello stato del dispositivo mentre il dispositivo è disconnesso da Device Shadows ... 252
Creazione di un autorizzatore personalizzato per AWS IoT Core ... 283
Monitoraggio dell'umidità del suolo con AWS IoT Raspberry Pi ... 302
Connect a AWS IoT Core ... 316
Endpoint del piano di controllo AWS IoT Core ... 316
AWS IoT endpoint del dispositivo ... 318
AWS IoT Core per gateway e dispositivi WAN LoRa ... 319
Connect agli endpoint AWS IoT Core del servizio ... 320
AWS CLI per AWS IoT Core ... 321
AWS SDKs ... 322
AWS Mobile SDKs ... 327
RESTO APIs dei servizi AWS IoT Core ... 328
Connect i dispositivi a AWS IoT ... 329
AWS IoT dati del dispositivo e endpoint di servizio ... 330

AWS IoT Device SDKs 332
Protocolli di dispositivo di comunicazione 335
Argomenti MQTT 386
Configurazioni del dominio 414
Connect agli AWS IoT endpoint FIPS 443
Endpoint del piano di controllo AWS IoT Core 443
AWS IoT Core- Endpoint del piano dati 443
AWS IoT Core- endpoint del fornitore di credenziali 444
AWS IoT Device Management - endpoint dati di processo 444
AWS IoT Device Management - endpoint Fleet Hub 445
AWS IoT Device Management - Endpoint di tunneling sicuro 445
AWS IoT Device Management - Endpoint di integrazioni gestite 445
Gestione dei dispositivi 447
Registro 448
Crea un oggetto 448
Visualizzazione dell'elenco di oggetti 449
Descrizione di oggetti 451
Aggiornamento di un oggetto 452
Eliminazione di un oggetto 452
Collegamento di un'entità principale a un oggetto 453
Elenca le cose associate a un preside 454
Elenca i principi associati a una cosa 454
Elenca le cose associate a una V2 principale 455
Elenca i principi associati a una cosa V2 456
Scollegamento di un'entità principale da un oggetto 456
Tipi di oggetti 457
Creazione di un tipo di oggetto 457
Visualizzazione dell'elenco di tipi di oggetti 458
Descrizione di un tipo di oggetto 459
Associazione di un tipo di oggetto a un oggetto 459
Aggiornare un tipo di oggetto 460
Dichiarazione di un tipo di oggetto come obsoleto 460
Eliminazione di un tipo di oggetto 461
Gruppi di oggetti statici 462
Creazione di un gruppo di oggetti statico 464
Descrizione di un gruppo di oggetti 465

Aggiunta di un oggetto a un gruppo di oggetti statico ... 466
Rimozione di un oggetto da un gruppo di oggetti statico ... 466
Elenco di oggetti in un gruppo di oggetti ... 467
Visualizzazione dell'elenco di gruppi di oggetti ... 468
Elenco dei gruppi per un oggetto ... 470
Aggiornamento di un gruppo di oggetti statico ... 471
Eliminazione di un gruppo di oggetti ... 471
Collegamento di una policy a un gruppo di oggetti statico ... 472
Scollegamento di una policy da un gruppo di oggetti statico ... 472
Elenco di policy collegate a un gruppo di oggetti statico ... 473
Visualizzazione dell'elenco di gruppi per una policy ... 473
Recupero delle policy valide per un oggetto ... 474
Test dell'autorizzazione per le operazioni MQTT ... 475
Gruppo di oggetti dinamici ... 476
Casi d'uso di gruppi di oggetti dinamici ... 477
Creazione di un gruppo di oggetti dinamico ... 479
Descrizione di un gruppo di oggetti dinamico ... 480
Aggiornamento di un gruppo di oggetti dinamico ... 481
Eliminazione di un gruppo di oggetti dinamico ... 482
Limitazioni dei gruppi di oggetti dinamici e statici ... 482
Limitazioni di Dynamic Thing Group ... 482
Associa un oggetto alla connessione ... 485
Casi d’uso ... 485
Come associare un oggetto a una connessione ... 486
Aggiungi attributi di propagazione ... 489
Console di gestione AWS ... 490
AWS CLI ... 491
Aggiunta di tag alle risorse ... 493
Nozioni di base sui tag ... 493
Restrizioni e limitazioni di tag ... 494
Etichetta con le politiche IAM ... 495
Gruppi di fatturazione ... 497
Visualizzazione dell'allocazione dei costi e dei dati di utilizzo ... 498
Sicurezza ... 500
Sicurezza in AWS IoT ... 501
Autenticazione ... 502

Panoramica sui certificati X.509 ... 502
Autenticazione del server ... 502
Autenticazione client ... 506
Autenticazione e autorizzazione personalizzata ... 547
Autorizzazione ... 577
AWS formazione e certificazione ... 580
AWS IoT Core politiche ... 580
Autorizzazione di chiamate dirette ai AWS servizi utilizzando il provider di AWS IoT Core credenziali ... 662
Accesso tra account con IAM ... 669
Protezione dei dati ... 671
Crittografia dei dati in AWS IoT ... 672
Sicurezza dei trasporti in AWS IoT Core ... 672
Crittografia dei dati ... 678
Gestione dell'identità e degli accessi ... 694
Destinatari ... 695
Autenticazione con identità IAM ... 695
Gestione dell'accesso tramite policy ... 696
Come AWS IoT funziona con IAM ... 698
Esempi di policy basate su identità ... 730
AWS politiche gestite ... 735
Risoluzione dei problemi ... 750
Registrazione e monitoraggio ... 752
Strumenti di monitoraggio ... 752
Convalida della conformità ... 754
Resilienza ... 755
Utilizzo AWS IoT Core con endpoint VPC ... 755
Creazione di endpoint VPC per il piano di controllo AWS IoT Core ... 756
Creazione di endpoint VPC per il piano dati AWS IoT Core ... 756
Creazione di endpoint VPC per provider di credenziali AWS IoT Core ... 757
Creazione di un endpoint di interfaccia Amazon VPC ... 758
Configura una zona ospitata privata ... 760
Controllo dell'accesso agli AWS IoT Core endpoint tramite VPC ... 763
Limitazioni ... 764
Scalabilità degli endpoint VPC con AWS IoT Core ... 765
Utilizzo di domini personalizzati con endpoint VPC ... 765

Disponibilità di endpoint VPC per AWS IoT Core ... 766
Utilizzo AWS IoT Device Management con endpoint VPC ... 766
Sicurezza dell'infrastruttura ... 771
Controllo della sicurezza ... 772
Best practice di sicurezza ... 773
Protezione delle connessioni MQTT in AWS IoT ... 773
Tieni sincronizzato l'orologio del dispositivo ... 776
Convalidare il certificato server ... 776
Utilizzare una singola identità per dispositivo ... 777
Usa un secondo Regione AWS come backup ... 777
Utilizzare il provisioning Just In Time ... 777
Autorizzazioni per eseguire i test di AWS IoT Device Advisor ... 778
Prevenzione del problema "confused deputy" tra servizi per Device Advisor ... 779
AWS formazione e certificazione ... 781
Monitorare AWS IoT ... 782
Configurare la registrazione AWS IoT ... 783
Configurazione dell'accesso V2 AWS IoT ... 784
Configurare il ruolo e il criterio di registrazione ... 784
Configura la registrazione nella (console) AWS IoT ... 787
Configurazione dell'accesso a livello di account ed evento (AWS IoT CLI) ... 789
Configura le sostituzioni specifiche per le risorse in (CLI) AWS IoT ... 792
Livelli di log ... 794
Monitora AWS IoT allarmi e metriche con Amazon CloudWatch ... 795
Utilizzo delle metriche AWS IoT ... 795
Creare allarmi CloudWatch ... 796
Metriche e dimensioni ... 801
Monitora AWS IoT utilizzando i log CloudWatch ... 825
Visualizzazione dei AWS IoT log nella console CloudWatch ... 826
CloudWatch Registra le voci di registro AWS IoT ... 827
Carica i log lato dispositivo su Amazon CloudWatch ... 866
Come funziona ... 867
Caricamento dei log lato dispositivo utilizzando le regole AWS IoT ... 868
Registra le AWS IoT chiamate API ... 878
AWS IoT informazioni in CloudTrail ... 878
Comprendere le AWS IoT voci dei file di registro ... 879
Regole ... 882

Concessione dell'accesso 883
Revoca l'accesso al motore delle regole 886
Passaggio delle autorizzazioni di un ruolo 886
Creazione di una regola 887
Crea una regola (Console) 889
Creare una regola (CLI) 890
Gestire una regola 894
Taggare una regola 895
Visualizzazione di una regola 896
Eliminazione di una regola 896
AWS IoT azioni relative alle regole 897
Apache Kafka 900
CloudWatch allarmi 913
CloudWatch Registri 915
CloudWatch metriche 917
DynamoDB 919
Dynamo DBv2 923
Elasticsearch 925
HTTP 928
AWS IoT Events 942
AWS IoT SiteWise 944
Firehose 950
Flussi di dati Kinesis 953
Lambda 955
Location (Ubicazione) 958
OpenSearch 962
Ripubblica 965
S3 968
IoT di Salesforce 970
SNS 972
SQS 974
Step Functions 977
Timestream 978
Risoluzione dei problemi relativi a una regola 986
Accedi alle risorse di più account 986
Prerequisiti 987

Configurazione tra account per Amazon SQS 987
Configurazione tra account per Amazon SNS 989
Configurazione tra account per Amazon S3 991
Configurazione tra più account per AWS Lambda 993
Gestione degli errori (operazione in caso di errore) 996
Formato del messaggio dell'operazione da eseguire in caso di errore 996
Esempio di operazione in caso di errore 998
Basic Ingest 999
Utilizzo di Basic Ingest 999
AWS IoT Riferimento SQL 1000
Clausola SELECT 1002
Clausola FROM 1004
Clausola SET 1006
Clausola WHERE 1009
Tipi di dati 1010
Operatori 1016
Funzioni 1027
Valori letterali 1103
Istruzioni case 1104
Estensioni JSON 1105
Modelli di sostituzione 1107
Query di oggetti nidificati 1110
Payload binari 1111
Versioni SQL 1118
Shadows 1120
Uso delle copie shadow 1120
Scelta di utilizzare copie shadow con nome o senza nome 1121
Accesso alle copie shadow 1122
Utilizzo delle copie shadow in dispositivi, app e altri servizi cloud 1123
Ordine dei messaggi 1123
Taglio dei messaggi delle copie shadow 1125
Utilizzo delle copie shadow nei dispositivi 1126
Inizializzazione del dispositivo alla prima connessione a AWS IoT 1127
Elaborazione dei messaggi mentre il dispositivo è connesso a AWS IoT 1130
Elaborazione dei messaggi quando il dispositivo si riconnette a AWS IoT 1131
Utilizzo delle copie shadow in app e servizi 1131

Inizializzazione dell'app o del servizio durante la connessione a AWS IoT 1132
Lo stato di elaborazione cambia mentre l'app o il servizio sono connessi a AWS IoT 1133
Rilevamento della connessione di un dispositivo 1133
Simulazione delle comunicazioni del servizio Device Shadow 1135
Impostazione della simulazione 1135
Inizializzare il dispositivo 1136
Inviare un aggiornamento dall'app 1140
Rispondere all'aggiornamento nel dispositivo 1142
Osservare l'aggiornamento nell'app 1147
Andare oltre la simulazione 1149
Interazione con le copia shadow 1149
Supporto dei protocolli 1150
Stato di richiesta e segnalazione 1150
Aggiornamento di una copia shadow 1151
Recupero di un documento di una copia shadow 1155
Eliminazione di dati shadow 1156
API REST del servizio Device Shadow 1159
GetThingShadow 1160
UpdateThingShadow 1161
DeleteThingShadow 1163
ListNamedShadowsForThing 1164
Argomenti MQTT di Device Shadow 1165
/get 1167
/get/accepted 1167
/get/rejected 1168
/update 1169
/update/delta 1171
/update/accepted 1172
/update/documents 1173
/update/rejected 1174
/delete 1175
/delete/accepted 1176
/delete/rejected 1177
Documenti del servizio Device Shadow 1178
Esempi di documenti shadow 1178
Proprietà del documento 1184

Stato delta 1185
Controllo delle versioni documenti shadow 1188
I token client nei documenti shadow 1188
Proprietà documento shadow vuote 1188
Valori di array nei documenti shadow 1189
Messaggi di errore Device Shadow 1190
Software Package Catalog 1192
Preparazione all'uso di Software Package Catalog 1193
Ciclo di vita della versione del pacchetto 1193
Convenzioni di denominazione della versione del pacchetto 1195
Versione predefinita 1195
Attributi della versione 1195
Elenco degli elementi software 1196
Abilitare l'indicizzazione del AWS IoT parco veicoli 1200
Copia shadow con nome riservata 1200
Eliminazione di un pacchetto software 1202
Preparazione della sicurezza 1202
Autenticazione basata sulle risorse 1202
AWS IoT Job rights per distribuire le versioni dei pacchetti 1204
AWS IoT Diritti di lavoro per aggiornare l'ombra denominata riservata 1205
AWS IoT Autorizzazioni per il download di Jobs da Amazon S3 1208
Autorizzazioni per aggiornare la distinta base del software per una versione del pacchetto 1208
Preparazione dell'indicizzazione del parco istanze 1211
Impostazione della copia shadow `$package` come un'origine dati 1211
Parametri visualizzati nella console 1212
Modelli di query 1213
Raccolta della distribuzione delle versioni dei pacchetti tramite `getBucketsAggregation` 1216
Preparazione AWS IoT dei lavori 1216
Parametri di sostituzione per i lavori AWS IoT 1216
Preparazione del documento del processo e della versione del pacchetto per la distribuzione 1221
Assegnazione di un nome ai pacchetti e alle versioni durante la distribuzione 1225
Indirizzazione dei lavori tramite gruppi di AWS IoT oggetti dinamici 1225
Copia shadow con nome riservata e versioni dei pacchetti 1226
Disinstallazione di un pacchetto software 1227

Nozioni di base 1227
Creazione di un pacchetto e di una versione 1228
Distribuzione di una versione del pacchetto 1231
Associazione di una versione del pacchetto 1233
Jobs 1235
Accedere ai AWS IoT lavori 1235
AWS IoT Offerte di lavoro, regioni ed endpoint. 1235
Cos'è un'operazione remota? 1236
Vantaggi dell'utilizzo di AWS IoT Device Management Jobs per operazioni remote 1236
Che cos'è Jobs AWS IoT ? 1238
Concetti chiave dei processi 1240
Processi e stati di esecuzione dei processi 1244
Gestione dei processi 1249
Firma del codice per i processi 1249
Documento di processo 1249
Predefinito URLs 1250
URL predefinito per il caricamento dei file 1252
URL predefinito utilizzando il controllo delle versioni di Amazon S3 1253
Creazione e gestione dei processi utilizzando la console 1255
Creazione e gestione dei processi utilizzando la CLI 1258
Modelli di processo 1270
Modelli personalizzati e AWS gestiti 1271
Utilizza modelli AWS gestiti 1271
Creazione di un modello di processo personalizzato 1291
Configurazioni diprocesso 1300
Come funzionano le configurazioni di processo 1300
Specificare configurazioni aggiuntive 1316
Dispositivi e processi 1326
Programmazione dei dispositivi per l'uso di Jobs 1329
Flusso di lavoro dei dispositivi 1329
Flusso di lavoro dei processi 1331
Notifiche dei processi 1336
AWS IoT operazioni API per lavori 1344
Tipi di dati e API di gestione e controllo dei processi 1347
Tipi di dati e operazioni API MQTT e HTTPS per i dispositivi per il servizio Jobs 1367
Protezione di utenti e dispositivi per Jobs 1382

Tipo di policy richiesto per Jobs AWS IoT 1382
Autorizzazione di utenti e servizi cloud all'utilizzo di Jobs 1384
Autorizzazione dei dispositivi a utilizzare i processi 1397
AWS IoT Limiti dei lavori 1402
Limiti delle esecuzioni di Job 1402
Limiti dei processi attivi e simultanei 1403
Comandi 1408
Avvio rapido 1409
Comandi, concetti e stato 1410
Comandi, concetti chiave 1410
Stati dei comandi 1415
Stato di esecuzione del comando 1415
Workflow dei comandi 1419
Crea e gestisci i comandi 1420
Scegli gli obiettivi e iscriviti agli argomenti 1420
Avvia e monitora le esecuzioni di comandi 1422
(Facoltativo) Abilita le notifiche per gli eventi dei comandi 1423
Crea e gestisci i comandi 1424
Crea una risorsa di comando 1425
Recuperare informazioni su un comando 1430
Elenca i comandi nel tuo Account AWS 1432
Aggiornare una risorsa di comando 1434
Deprecare o ripristinare una risorsa di comando 1436
Eliminare una risorsa di comando 1437
Avvia e monitora le esecuzioni di comandi 1439
Avvia l'esecuzione di un comando 1439
Aggiorna il risultato dell’esecuzione di un comando 1446
Recupera l'esecuzione di un comando 1452
Visualizzazione degli aggiornamenti dei comandi utilizzando il client di test MQTT 1456
Elenca le esecuzioni dei comandi nel tuo Account AWS 1458
Eliminare l'esecuzione di un comando 1461
Deprecare una risorsa di comando 1462
Considerazioni chiave 1462
Deprecare una risorsa di comando (console) 1463
Deprecare una risorsa di comando (CLI) 1463
Controlla l'ora e lo stato di deprecazione 1463

Ripristina una risorsa di comando 1464
Tunneling sicuro 1465
Che cos'è il tunneling sicuro? 1465
Concetti di tunneling sicuri 1466
Come funziona il tunneling sicuro 1467
Ciclo di vita sicuro del tunnel 1468
Tutorial sul tunneling sicuro 1469
Tutorial in questa sezione 1470
Apertura di un tunnel e avvio di una sessione SSH su un dispositivo remoto 1470
Apertura di un tunnel per dispositivo remoto e utilizzo di SSH basato su browser 1489
Proxy locale 1493
Come utilizzare il proxy locale 1494
Configurare il proxy locale per i dispositivi che utilizzano il proxy web 1501
Multiplexing e connessioni TCP simultanee 1509
Multiplexing di più flussi di dati 1509
Utilizzo di connessioni TCP simultanee 1513
Configurazione di un dispositivo remoto e utilizzo dell'agente IoT 1516
Snippet dell'agente IoT 1516
Controllo dell'accesso ai tunnel 1518
Prerequisiti di accesso al tunnel 1518
Policy di accesso al tunnel 1518
Risoluzione dei problemi di connettività di tunneling sicuro 1526
Errore del token di accesso client non valido 1526
Errore di mancata corrispondenza del token client 1527
Problemi di connettività del dispositivo remoto 1528
Provisioning dei dispositivi 1531
Fornitura dei dispositivi in AWS IoT 1532
Approvvigionamento della flotta APIs 1533
Provisioning di dispositivi che non dispongono di certificati dispositivo mediante il provisioning del parco istanze dispositivi 1534
Provisioning tramite attestazione 1535
Provisioning tramite utente attendibile 1538
Utilizzo degli hook di pre-provisioning con la CLI AWS 1540
Provisioning dei dispositivi che dispongono di certificati dispositivo 1544
Provisioning di un singolo oggetto 1544
Just-in-time approvvigionamento 1545

Registrazione in blocco ... 1552
Modelli di provisioning ... 1553
Sezione Parametri ... 1553
Sezione Risorse ... 1554
Esempio di modello per la registrazione in blocco ... 1560
Esempio di modello per il just-in-time provisioning (JITP) ... 1561
Provisioning del parco istanze ... 1563
Hook di pre-provisioning ... 1567
Input dell'hook di pre-provisioning ... 1567
Valore restituito dall'hook di pre-provisioning ... 1568
Esempio di hook di pre-provisioning Lambda ... 1569
Firma dei certificati autogestita tramite provider di certificati AWS IoT Core ... 1571
Come funziona la firma dei certificati autogestita nel provisioning del parco veicoli ... 1572
Input della funzione Lambda del fornitore di certificati ... 1574
Valore restituito dalla funzione Lambda del fornitore di certificati ... 1574
Funzione Lambda di esempio ... 1575
Firma dei certificati autogestita per il rifornimento del parco veicoli ... 1577
AWS CLI comandi per il fornitore di certificati ... 1578
Creazione di policy e ruoli IAM per un utente che installa un dispositivo ... 1581
Creazione di una policy IAM per l'utente che installerà un dispositivo ... 1581
Creazione di un ruolo IAM per l'utente che installerà un dispositivo ... 1582
Aggiornamento di una policy esistente per autorizzare un nuovo modello ... 1583
API MQTT di provisioning del dispositivo ... 1585
CreateCertificateFromCsr ... 1585
CreateKeysAndCertificate ... 1588
RegisterThing ... 1590
Indicizzazione della flotta ... 1593
Gestione degli aggiornamenti degli indici ... 1593
Interrogazione dello stato della connettività per un dispositivo specifico ... 1593
Ricerca tra origini dei dati ... 1593
Interrogazione di dati aggregati ... 1594
Monitoraggio dei dati di aggregazione e creazione di allarmi utilizzando parametri del parco istanze ... 1594
Gestione dell'indicizzazione del parco istanze ... 1594
Indicizzazione degli oggetti ... 1594
Indicizzazione di gruppi di oggetti ... 1596

Campi gestiti ... 1596
Campi personalizzati ... 1598
Gestione dell'indicizzazione degli oggetti ... 1599
Gestione dell'indicizzazione di gruppi di oggetti ... 1615
Stato della connettività del dispositivo ... 1617
Come funziona ... 1618
Funzionalità ... 1618
Vantaggi ... 1618
Prerequisiti ... 1619
Esempi ... 1619
Interrogazione di dati aggregati ... 1621
GetStatistics ... 1621
GetCardinality ... 1624
GetPercentiles ... 1625
GetBucketsAggregation ... 1628
Autorizzazione ... 1629
Sintassi delle query ... 1629
Funzionalità supportate ... 1629
Caratteristiche non supportate ... 1630
Note ... 1630
Esempio di query per oggetti ... 1631
Esempio di query per gruppi di oggetti ... 1636
Indicizzazione dei dati sulla posizione ... 1637
Formati di dati supportati ... 1638
Come indicizzare i dati sulla posizione ... 1639
Aggiorna la configurazione di indicizzazione degli oggetti ... 1639
Geoquery di esempio ... 1642
Tutorial sulle nozioni di base ... 1643
Metriche della flotta ... 1648
Tutorial sulle nozioni di base ... 1649
Gestione dei parametri del parco istanze ... 1656
Distribuzione di file basata su MQTT ... 1664
Che cos'è un flusso? ... 1664
Gestisci uno stream ... 1665
Concedere le autorizzazioni ai tuoi dispositivi ... 1666
Connect i tuoi dispositivi a AWS IoT ... 1667

TagResource Utilizzo ... 1667
Utilizzate la distribuzione di AWS IoT file basata su MQTT nei dispositivi ... 1668
Utilizzato DescribeStream per ottenere dati in streaming ... 1669
Ottieni blocchi di dati da un file di flusso ... 1671
Gestione degli errori derivanti dalla consegna di file basata su MQTT AWS IoT ... 1677
Un esempio di caso d'uso in FreeRTOS OTA ... 1679
Device Advisor ... 1680
Configurazione ... 1682
Creare un oggetto IoT ... 1682
Creazione di un ruolo IAM da utilizzare come ruolo del dispositivo ... 1682
Crea una policy gestita su misura per un utente IAM per l'uso di Device Advisor ... 1685
Crea un utente IAM per utilizzare Device Advisor ... 1686
Configurazione del dispositivo ... 1689
Nozioni di base su Device Advisor nella console ... 1690
Flusso di lavoro di Device Advisor ... 1700
Prerequisiti ... 1700
Crea una definizione di suite di test ... 1700
Ottieni una definizione di suite di test ... 1703
Ottieni un endpoint di test ... 1703
Avvia una suite di test ... 1703
Ottieni un'esecuzione della suite di test ... 1704
Interrompi l'esecuzione di una suite di test ... 1705
Ottieni un rapporto di qualificazione per una suite di test di qualificazione riuscita ... 1705
Flusso di lavoro della console dettagliato di Device Advisor ... 1706
Prerequisiti ... 1706
Crea una definizione di suite di test ... 1706
Avvia una suite di test ... 1714
Interrompi l'esecuzione di una suite di test (facoltativo) ... 1716
Visualizza i dettagli e i registri di esecuzione della suite ... 1718
Scarica un rapporto di AWS IoT qualificazione ... 1720
Flusso di lavoro della console per test di lunga durata ... 1720
Endpoint VPC di Device Advisor (AWS PrivateLink) ... 1729
Considerazioni sugli endpoint AWS IoT Core Device Advisor VPC ... 1730
Crea un endpoint VPC di interfaccia per AWS IoT Core Device Advisor ... 1730
Controllo dell'accesso agli AWS IoT Core Device Advisor endpoint tramite VPC ... 1731
Case test di Device Advisor ... 1732

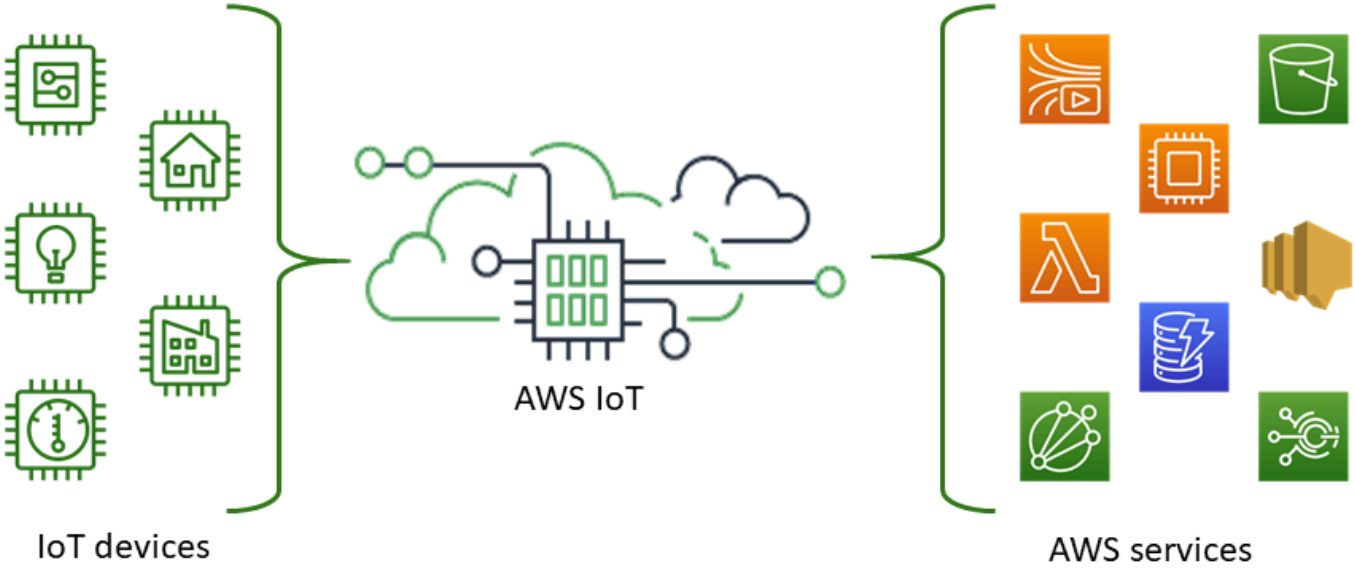
Casi di test di Device Advisor per qualificarsi per il AWS Device Qualification Program. 1733
TLS 1734
MQTT 1741
Shadow 1755
Esecuzione del processo 1757
Autorizzazioni e policy 1759
Test di lunga durata 1760
Ubicazione del dispositivo 1779
Tipi di misurazione e risolutori 1779
Come funziona AWS IoT Core Device Location 1781
Come usare Device Location AWS IoT Core 1782
Risoluzione della posizione di dispositivi IoT 1783
Risoluzione della posizione del dispositivo (console) 1783
Risoluzione della posizione del dispositivo (API) 1787
Risoluzione dei problemi relativi alla risoluzione della posizione 1789
Risoluzione della posizione del dispositivo utilizzando argomenti MQTT 1790
Formato degli argomenti relativi alla posizione del dispositivo 1790
Policy per gli argomenti MQTT relativi alla posizione dei dispositivi 1791
Argomenti relativi alla posizione del dispositivo e payload 1792
Risolutori di posizione e payload del dispositivo 1798
Risolutore basato su Wi-Fi 1798
Risolutore basato su rete cellulare 1799
Risolutore di ricerca inversa IP 1804
Risolutore GNSS 1805
Solver BLE per dispositivi compatibili con Amazon Sidewalk 1806
Messaggi di eventi 1808
Come vengono generati i messaggi di evento 1808
Policy per la ricezione di messaggi di evento 1808
Abilita eventi per AWS IoT 1809
Eventi del registro 1814
Eventi oggetto 1814
Eventi di tipo di oggetto 1816
Eventi del gruppo di oggetti 1819
Eventi del servizio Jobs 1825
Eventi di esecuzione dei comandi 1830
Eventi del ciclo di vita 1831

Eventi di connessione/disconnessione 1832
Evento di tentativo di connessione fallito 1837
Eventi di sottoscrizione/annullamento della sottoscrizione 1838
Risoluzione dei problemi 1841
AWS IoT Core guida alla risoluzione dei problemi 1841
Diagnosi dei problemi di connettività 1842
Diagnosi dei problemi relativi alle regole 1845
Diagnosi dei problemi relativi a Shadows 1847
Diagnosi dei problemi relativi alle operazioni di Salesforce 1849
Diagnosi dei limiti di flusso 1851
Risoluzione degli errori di disconnessione del parco istanze dei dispositivi 1851
AWS IoT Device Management guida alla risoluzione dei problemi 1852
AWS IoT Risoluzione dei problemi dei lavori 1853
Risoluzione dei problemi di indicizzazione della flotta 1858
AWS IoT Risoluzione dei problemi del Device Management Software Package Catalog 1861
AWS IoT Guida alla risoluzione di Device Advisor 1868
AWS IoT errori 1871
AWS IoT Client per dispositivi SDKs SDKs, dispositivi mobili e AWS IoT dispositivi mobili 1873
AWS IoT Device SDKs 1873
AWS IoT SDK per dispositivi per Embedded C 1875
AWS Mobile SDKs 1876
AWS IoT Client del dispositivo 1877
Versioni precedenti AWS IoT del dispositivo SDKs 1878
Esempi di codice 1879
Nozioni di base 1880
ciao AWS IoT 1880
Informazioni di base 1889
Azioni 2000
AWS IoT quote 2095
AWS IoT Core prezzi 2096
..... mmxcvii

Che cos'è AWS IoT?

AWS IoT fornisce i servizi cloud che collegano i dispositivi IoT ad altri dispositivi e servizi AWS cloud. AWS IoT fornisce software per dispositivi che possono aiutarti a integrare i tuoi dispositivi IoT in soluzioni AWS IoT basate. Se i tuoi dispositivi sono in grado di connettersi a AWS IoT, AWS IoT puoi connetterli ai servizi cloud che AWS fornisce.

Per un'introduzione pratica a AWS IoT, visita. Tutorial introduttivi



Con AWS IoT, puoi selezionare le up-to-date tecnologie più appropriate per la tua soluzione. Per aiutarti a gestire e supportare i tuoi dispositivi IoT sul campo, AWS IoT Core supporta questi protocolli:

- MQTT (Message Queuing and Telemetry Transport)
- MQTT su WSS (Websocket Secure)
- HTTPS (Hypertext Transfer Protocol - Secure)
- LoRaWAN (Long Range Wide Area Network)

Il broker di AWS IoT Core messaggi supporta dispositivi e client che utilizzano i protocolli MQTT e MQTT su WSS per pubblicare e sottoscrivere i messaggi. Supporta inoltre dispositivi e client che utilizzano il protocollo HTTPS per pubblicare messaggi.

AWS IoT Core for LoRa WAN consente di connettere e gestire dispositivi LoRa WAN wireless (Wide Area Network a lungo raggio a basso consumo). AWS IoT Core for LoRa WAN sostituisce la necessità di sviluppare e gestire un server di rete LoRa WAN (LNS).

Se non sono necessarie AWS IoT funzionalità come le comunicazioni tra dispositivi, le regole o i processi, consulta AWS Messaggistica per informazioni su altri servizi di AWS IoT messaggistica che potrebbero soddisfare meglio le tue esigenze.

Come accedono i tuoi dispositivi e le tue app AWS IoT

AWS IoT fornisce le seguenti interfacce perAWS IoT tutorial:

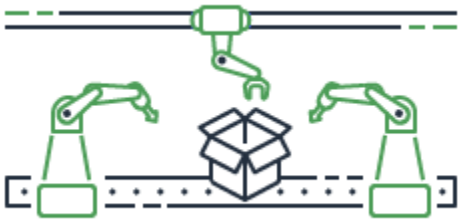
- AWS IoT Dispositivo SDKs: crea applicazioni sui tuoi dispositivi che inviano e ricevono messaggi da. AWS IoT Per ulteriori informazioni, consulta AWS IoT Client per dispositivi SDKs SDKs, dispositivi mobili e AWS IoT dispositivi mobili.
- AWS IoT Core per LoRa WAN: collega e gestisci i dispositivi e i gateway LoRa WAN a lungo raggio (WAN) utilizzando AWS IoT Core for LoRa WAN.
- AWS Command Line Interface (AWS CLI) —Esegue AWS IoT comandi per Windows, macOS e Linux. Con questi comandi puoi creare e gestire oggetti, certificati, regole, processi e policy. Per iniziare, consulta la AWS Command Line Interface Guida per l'utente di . Per ulteriori informazioni sui comandi per AWS IoT, consulta iot nel AWS CLI Command Reference.
- AWS IoT API: crea le tue applicazioni IoT utilizzando richieste HTTP o HTTPS. Con queste operazioni API puoi creare e gestire in modo programmatico oggetti, certificati, regole e policy. Per ulteriori informazioni sulle azioni API per AWS IoT, consulta Azioni nell'AWS IoT API Reference.
- AWS SDKs—Crea le tue applicazioni IoT utilizzando specifiche lingue APIs. Questi SDKs racchiudono l' HTTP/HTTPS API e consentono di programmare in una qualsiasi delle lingue supportate. Per ulteriori informazioni, consulta AWS SDKs e Strumenti.

Puoi accedere anche AWS IoT tramite la AWS IoT console, che fornisce un'interfaccia utente grafica (GUI) tramite la quale puoi configurare e gestire oggetti, certificati, regole, lavori, politiche e altri elementi delle tue soluzioni IoT.

Cosa si può fare AWS IoT

In questo argomento vengono descritte alcune delle soluzioni che potresti aver bisogno che AWS IoT supporti.

IoT nel settore industriale



Questi sono alcuni esempi di AWS IoT soluzioni per casi d'uso industriali che applicano le tecnologie IoT per migliorare le prestazioni e la produttività dei processi industriali.

Soluzioni per casi d'uso industriali

- AWS IoT Utilizzatelo per creare modelli di qualità predittivi nelle operazioni industriali

  Scopri come AWS IoT raccogliere e analizzare i dati delle operazioni industriali per creare modelli di qualità predittivi. Ulteriori informazioni
- AWS IoT Utilizzatelo per supportare la manutenzione predittiva nelle operazioni industriali

  Scoprite come AWS IoT può aiutarvi a pianificare la manutenzione preventiva per ridurre i tempi di inattività non pianificati. Ulteriori informazioni

IoT nell’automazione domestica

Questi sono alcuni esempi di AWS IoT soluzioni per l'automazione domestica che applicano le tecnologie IoT per creare applicazioni IoT scalabili che automatizzano le attività domestiche utilizzando dispositivi domestici connessi.

Soluzioni per l’automazione domestica

- Utilizzalo AWS IoT nella tua casa connessa

Scopri come AWS IoT è possibile fornire soluzioni di automazione domestica integrate.

- Utilizzare AWS IoT per fornire sicurezza e monitoraggio domestici

  Scopri come AWS IoT applicare l'apprendimento automatico e l'edge computing alla tua soluzione di automazione domestica.

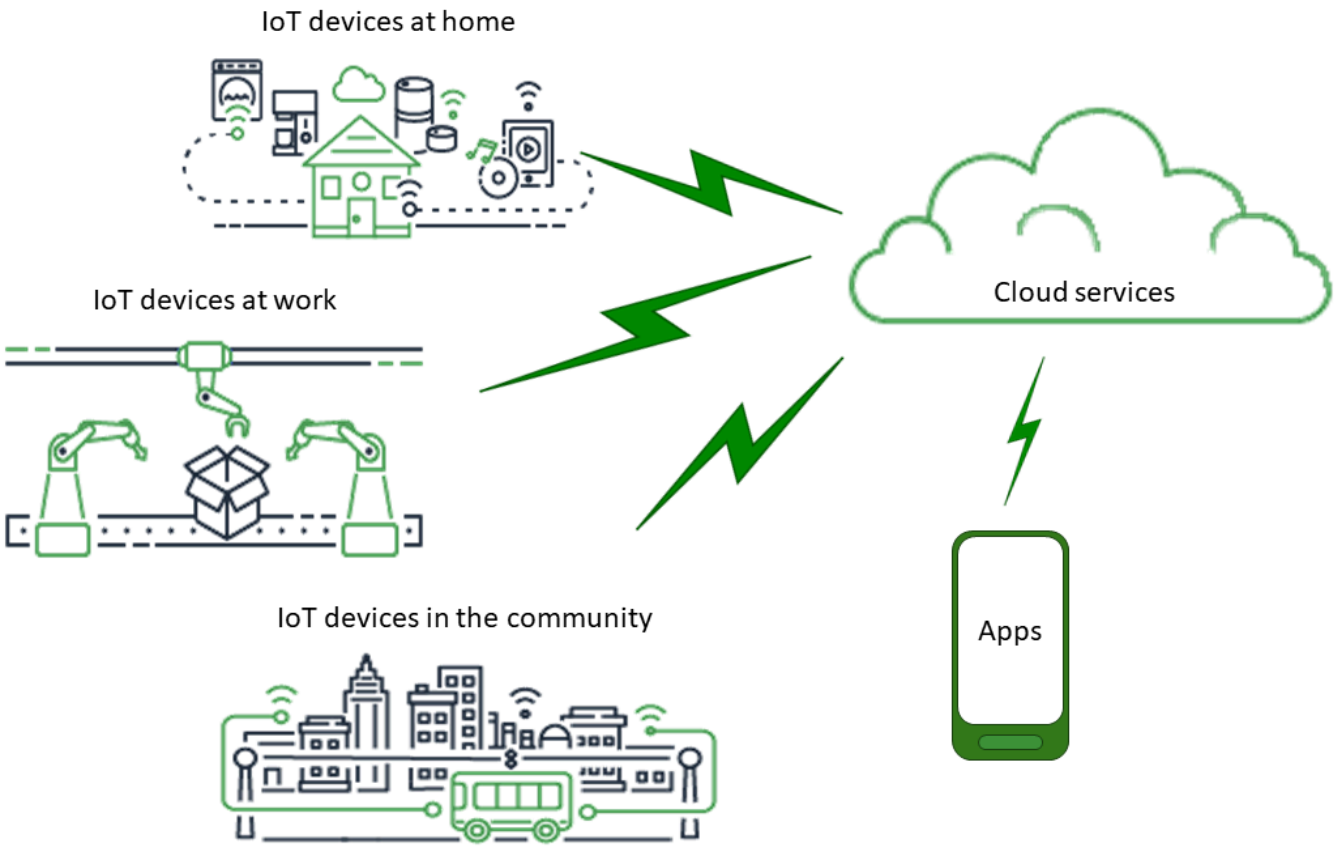
Per un elenco delle soluzioni per i casi d'uso industriali, consumer e commerciali, consulta Repository soluzione AWS IoT.

Come AWS IoT funziona

AWS IoT fornisce servizi cloud e supporto per dispositivi che puoi utilizzare per implementare soluzioni IoT. AWS fornisce molti servizi cloud per supportare applicazioni basate su IoT. Quindi, per aiutarti a capire da dove cominciare, questa sezione fornisce un diagramma e una definizione di concetti essenziali per introdurti nell'universo IoT.

L'universo IoT

In generale, l'Internet of things (IoT) è costituito dai componenti chiave illustrati in questo diagramma.



App

Le app offrono agli utenti finali l'accesso ai dispositivi IoT e alle funzionalità fornite dai servizi cloud a cui tali dispositivi sono connessi.

Servizi cloud

I servizi cloud sono servizi distribuiti, di archiviazione ed elaborazione di dati su larga scala che sono connessi a Internet. Gli esempi includono:

- Servizi di connessione e gestione IoT

  AWS IoT è un esempio di servizio di connessione e gestione IoT.
- Servizi di elaborazione, come Amazon Elastic Compute Cloud e AWS Lambda
- Servizi di database, come Amazon DynamoDB

Comunicazioni

I dispositivi comunicano con i servizi cloud utilizzando diverse tecnologie e protocolli. Gli esempi includono:

- Internet Wi-Fi/a banda larga
- Dati cellulari a banda larga
- Dati cellulari a banda stretta
- WAN (Wide Area Network) a lungo raggio LoRa
- Comunicazioni Proprietary RF

dispositivi

Un dispositivo è un tipo di hardware che gestisce interfacce e comunicazioni. Generalmente, i dispositivi si trovano in prossimità delle interfacce del mondo reale che monitorano e controllano. I dispositivi possono includere risorse di elaborazione e archiviazione, come microcontroller, CPU e memoria. Gli esempi includono:

- Raspberry Pi
- Arduino
- Assistenti di interfaccia vocale
- LoRaWAN e dispositivi
- Dispositivi Amazon Sidewalk
- Dispositivi IoT personalizzati

Interfacce

Un'interfaccia è un componente che connette un dispositivo al mondo fisico.

- Interfacce utente

  Componenti che permettono a dispositivi e utenti di comunicare tra loro.

  - Interfacce di input

    Consenti a un utente di comunicare con un dispositivo

Esempi: tastiera, pulsante

- Interfacce di output

  Consenti a un dispositivo di comunicare con un utente

  Esempi: display alfanumerico, display grafico, spia luminosa, campanello di allarme

- Sensori

  Componenti di input che misurano o percepiscono impulsi provenienti dal mondo esterno per favorire la comprensione da parte del dispositivo. Gli esempi includono:

  - Sensore di temperatura (converte la temperatura in un segnale analogico o digitale)
  - Sensore di umidità (converte l'umidità relativa in un segnale analogico o digitale)
  - Convertitore da analogico a digitale (converte una tensione analogica in un valore numerico)
  - Unità per la misurazione della distanza ad ultrasuoni (converte una distanza in un valore numerico)
  - Sensore ottico (converte un livello di luce in un valore numerico)
  - Fotocamera (converte i dati immagine in dati digitali)

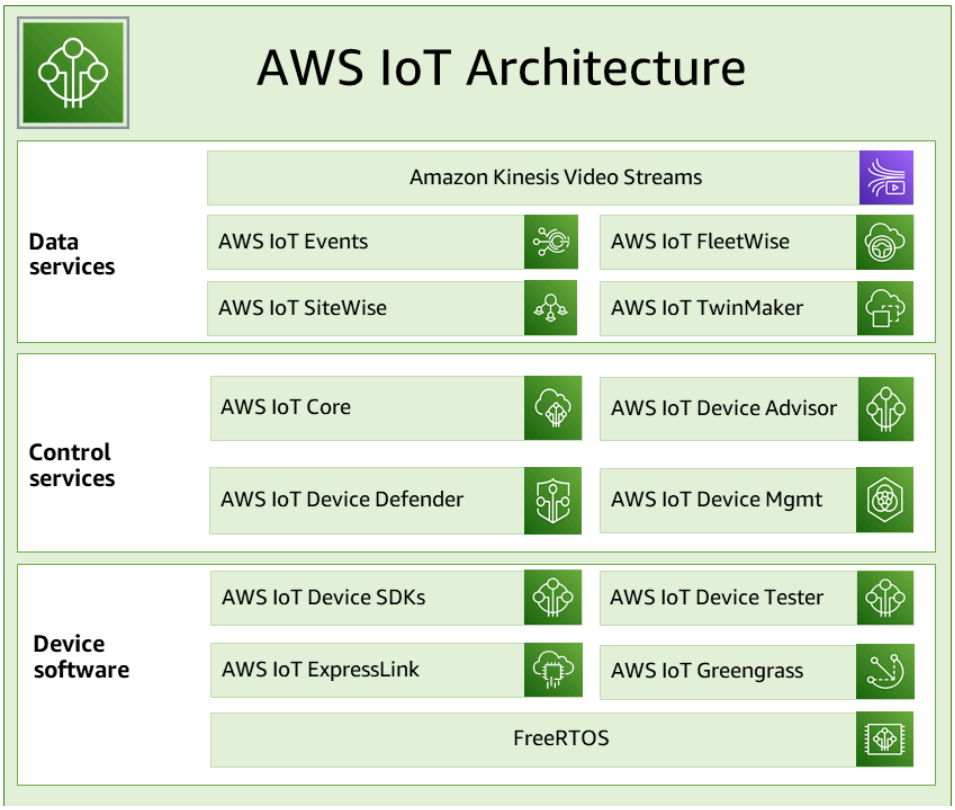
- Attuatori

  Componenti di output che il dispositivo può utilizzare per controllare qualcosa nel mondo esterno. Gli esempi includono:

  - Motori passo-passo (convertono segnali elettrici in movimento)
  - Relè (controllano tensioni e correnti elettriche elevate)

AWS IoT panoramica dei servizi

Nell'universo IoT, AWS IoT fornisce i servizi che supportano i dispositivi che interagiscono con il mondo e i dati che passano tra loro e AWS IoT. AWS IoT è costituito dai servizi mostrati in questa illustrazione per supportare la tua soluzione IoT.



AWS IoT software del dispositivo

AWS IoT fornisce questo software per supportare i tuoi dispositivi IoT.

AWS IoT Device SDKs

The AWS IoT Device and Mobile ti SDKs aiutano a connettere in modo efficiente i tuoi dispositivi a AWS IoT. The AWS IoT Device and Mobile SDKs include librerie open source, guide per sviluppatori con esempi e guide al porting in modo da poter creare prodotti o soluzioni IoT innovativi sulle piattaforme hardware che preferisci.

AWS IoT Device Tester

AWS IoT Device Testerper FreerTOS AWS IoT Greengrass ed è uno strumento di automazione dei test per microcontrollori. AWS IoT Device Tester testa il tuo dispositivo per determinare se eseguirà FreerTOS AWS IoT Greengrass o interopererà con i servizi. AWS IoT

AWS IoT ExpressLink

AWS IoT ExpressLink alimenta una gamma di moduli hardware sviluppati e offerti dai partner.AWS I moduli di connettività includono software AWS convalidato, che semplifica e velocizza la connessione sicura dei dispositivi al cloud e l'integrazione perfetta con una gamma

di servizi. AWS Per ulteriori informazioni, visita la pagina di AWS IoT ExpressLinkpanoramica o consulta la Guida per programmatori.AWS IoT ExpressLink

AWS IoT Greengrass

AWS IoT Greengrasssi estende AWS IoT ai dispositivi periferici in modo che possano agire localmente sui dati generati, eseguire previsioni basate su modelli di apprendimento automatico e filtrare e aggregare i dati dei dispositivi. AWS IoT Greengrass consente ai dispositivi di raccogliere e analizzare i dati più vicino al luogo in cui vengono generati, reagire in modo autonomo agli eventi locali e comunicare in modo sicuro con altri dispositivi sulla rete locale. È possibile utilizzarli AWS IoT Greengrass per creare applicazioni edge utilizzando moduli software predefiniti, denominati componenti, in grado di connettere i dispositivi perimetrali a AWS servizi o servizi di terze parti.

FreeRTOS

FreeRTOS è un sistema operativo open source in tempo reale per microcontroller che consente di includere piccoli dispositivi edge a basso consumo nella soluzione IoT. FreeRTOS include un kernel e un insieme crescente di librerie software che supportano molte applicazioni. I sistemi FreeRTOS possono connettere in modo sicuro i tuoi dispositivi di piccole dimensioni a basso consumo AWS IoT e supportare dispositivi edge più potenti in esecuzione AWS IoT Greengrass.

AWS IoT servizi di controllo

Connect ai seguenti AWS IoT servizi per gestire i dispositivi nella tua soluzione IoT.

AWS IoT Core

AWS IoT Coreè un servizio cloud gestito che consente ai dispositivi connessi di interagire in modo sicuro con le applicazioni cloud e altri dispositivi. AWS IoT Core può supportare molti dispositivi e messaggi e può elaborare e indirizzare tali messaggi verso AWS IoT endpoint e altri dispositivi. Con AWS IoT Core, le tue applicazioni possono interagire con tutti i tuoi dispositivi anche quando non sono connessi.

AWS IoT Core Device Advisor

AWS IoT Core Device Advisor è una funzionalità di test completamente gestita e basata sul cloud per la convalida dei dispositivi IoT durante lo sviluppo del software dei dispositivi. Device Advisor fornisce test predefiniti che è possibile utilizzare per convalidare i dispositivi IoT per una connettività affidabile e sicura prima di distribuirli in produzione. AWS IoT Core

AWS IoT Device Defender

AWS IoT Device Defender ti aiuta a proteggere la tua flotta di dispositivi IoT. AWS IoT Device Defender verifica continuamente le configurazioni IoT per assicurarsi che non si discostino dalle best practice di sicurezza. AWS IoT Device Defender invia un avviso quando rileva eventuali lacune nella configurazione IoT che potrebbero creare un rischio per la sicurezza, ad esempio certificati di identità condivisi tra più dispositivi o un dispositivo con un certificato di identità revocato che tenta di connettersi. AWS IoT Core

AWS IoT Gestione dei dispositivi

AWS IoT I servizi di gestione dei dispositivi ti aiutano a tracciare, monitorare e gestire la pletora di dispositivi connessi che compongono le tue flotte di dispositivi. AWS IoT I servizi di gestione dei dispositivi aiutano a garantire che i dispositivi IoT funzionino correttamente e in modo sicuro dopo la loro implementazione. Forniscono inoltre tunneling sicuro per accedere ai dispositivi, monitorarne l'integrità, rilevare e risolvere problemi da remoto, nonché servizi per gestire gli aggiornamenti del software e del firmware del dispositivo.

AWS IoT servizi dati

Analizza i dati dei dispositivi della tua soluzione IoT e intraprendi le azioni appropriate utilizzando i seguenti AWS IoT servizi.

Amazon Kinesis Video Streams

Amazon Kinesis Video Streams ti consente di trasmettere video in diretta dai dispositivi AWS al cloud, dove vengono archiviati, crittografati e indicizzati in modo duraturo, consentendoti di accedere ai tuoi dati tramite. easy-to-use APIs È possibile utilizzare Amazon Kinesis Video Streams per acquisire in tempo reale grandi volumi di dati video provenienti da milioni di sorgenti, compresi smartphone, telecamere di sicurezza, webcam, telecamere integrate nelle automobili, droni e altre origini. Amazon Kinesis Video Streams consente di riprodurre video per la visualizzazione live e on demand e di creare rapidamente applicazioni che sfruttano la visione artificiale e l'analisi video attraverso l'integrazione con Amazon Rekognition Video e librerie per framework ML, ma anche inviare dati non video serializzati nel tempo, ad esempio dati audio, immagini termiche, dati RADAR o sulla profondità e molto altro.

Amazon Kinesis Video Streams con WebRTC

Amazon Kinesis Video Streams con WebRTC fornisce un'implementazione WebRTC conforme agli standard come funzionalità completamente gestita. Puoi utilizzare Amazon Kinesis Video

Streams con WebRTC per trasmettere contenuti multimediali live in modo sicuro o eseguire interazioni audio o video bidirezionali tra qualsiasi dispositivo IoT dotato di fotocamera e lettori mobili o Web conformi allo standard WebRTC. Essendo una funzionalità completamente gestita, non è necessario creare, gestire o scalare alcuna infrastruttura cloud correlata a WebRTC, come server di segnalazione o media relay, per trasmettere in modo sicuro i contenuti multimediali tra applicazioni e dispositivi. Utilizzando Amazon Kinesis Video Streams con WebRTC, puoi creare facilmente peer-to-peer applicazioni per lo streaming multimediale live o l'interattività audio o video in tempo reale tra videocamere, dispositivi IoT, browser Web e dispositivi mobili per una varietà di casi d'uso.

AWS IoT Eventi

AWS IoT Events rileva e risponde agli eventi provenienti da sensori e applicazioni IoT. Gli eventi sono modelli di dati che identificano circostanze più complicate del previsto, come i rilevatori di movimento che utilizzano segnali di movimento per attivare luci e telecamere di sicurezza. AWS IoT Events monitora continuamente i dati provenienti da più sensori e applicazioni IoT e si integra con altri servizi, come AWS IoT Core IoT, SiteWise DynamoDB e altri per consentire il rilevamento precoce e approfondimenti unici.

AWS IoT FleetWise

AWS IoT FleetWiseè un servizio gestito che puoi utilizzare per raccogliere e trasferire i dati dei veicoli sul cloud quasi in tempo reale. Con AWS IoT FleetWise, puoi raccogliere e organizzare facilmente i dati dei veicoli che utilizzano protocolli e formati di dati diversi. AWS IoT FleetWise aiuta a trasformare i messaggi di basso livello in valori leggibili dall'uomo e a standardizzare il formato dei dati nel cloud per l'analisi dei dati. Puoi anche definire schemi di raccolta dati per controllare quali dati raccogliere sui veicoli e quando trasferirli nel cloud.

AWS IoT SiteWise

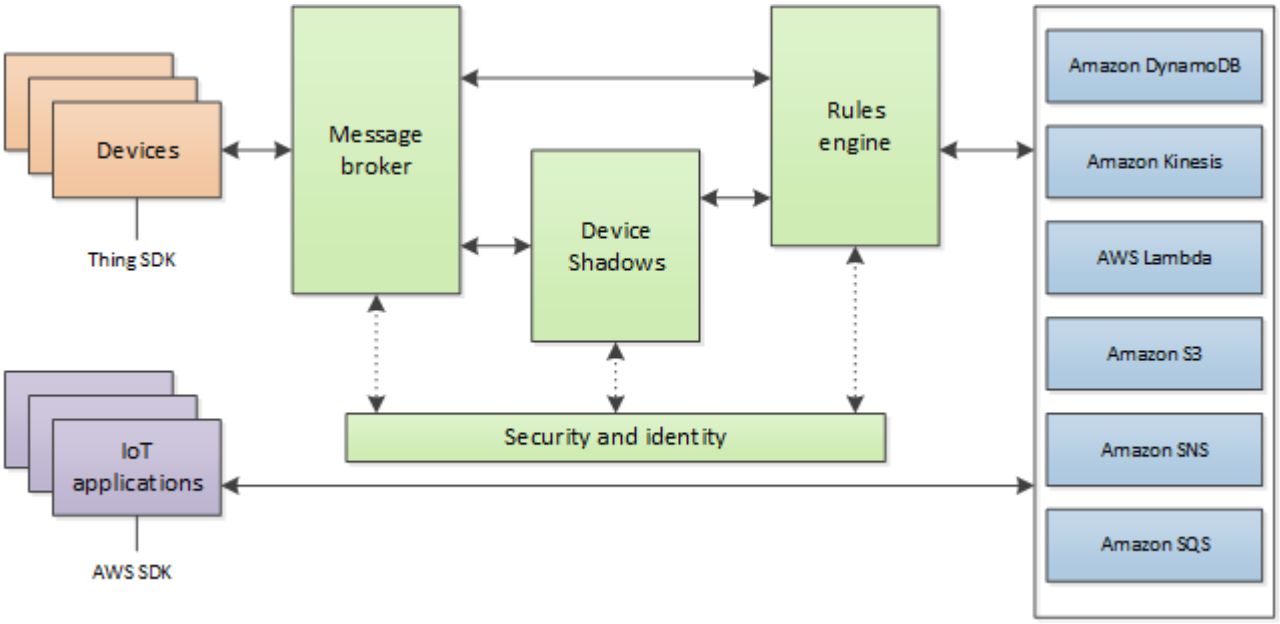
AWS IoT SiteWiseraccoglie, archivia, organizza e monitora i dati trasmessi dalle apparecchiature industriali tramite messaggi MQTT o su larga scala fornendo un software che funziona su un APIs gateway nelle vostre strutture. Il gateway si connette in modo sicuro ai server di dati locali e automatizza il processo di raccolta e organizzazione dei dati e il loro invio al Cloud. AWS

AWS IoT TwinMaker

AWS IoT TwinMakercrea gemelli digitali operativi di sistemi fisici e digitali. AWS IoT TwinMaker crea visualizzazioni digitali utilizzando misurazioni e analisi da una varietà di sensori, telecamere e applicazioni aziendali del mondo reale per aiutarvi a tenere traccia della vostra fabbrica fisica, edificio o impianto industriale. È possibile utilizzare dati reali per monitorare le attività, diagnosticare e correggere gli errori e ottimizzare l'operatività.

AWS IoT Core servizi

AWS IoT Core fornisce i servizi che connettono i tuoi dispositivi IoT al AWS Cloud in modo che altri servizi e applicazioni cloud possano interagire con i tuoi dispositivi connessi a Internet.



La sezione successiva descrive ciascuno dei AWS IoT Core servizi mostrati nell'illustrazione.

AWS IoT Core servizi di messaggistica

I servizi di AWS IoT Core connettività forniscono comunicazioni sicure con i dispositivi IoT e gestiscono i messaggi che passano tra loro e AWS IoT.

Gateway dei dispositivi

Permette ai dispositivi di comunicare in modo sicuro ed efficiente con AWS IoT. La comunicazione del dispositivo è protetta da protocolli sicuri che utilizzano certificati X.509.

Broker di messaggi

Fornisce un meccanismo sicuro per dispositivi e AWS IoT applicazioni per pubblicare e ricevere messaggi gli uni dagli altri. È possibile utilizzare direttamente il protocollo MQTT o MQTT over WebSocket per pubblicare e sottoscrivere. Per ulteriori informazioni sui protocolli che AWS IoT supporta, consulta the section called “Protocolli di dispositivo di comunicazione”. I dispositivi e i client possono anche utilizzare l'interfaccia HTTP REST per pubblicare dati sul broker di messaggi.

Il broker di messaggi distribuisce i dati del dispositivo ai dispositivi che lo hanno sottoscritto e ad altri AWS IoT Core servizi, come il servizio Device Shadow e il motore delle regole.

AWS IoT Core per WAN LoRa

AWS IoT Core per LoRa WAN consente di configurare una rete LoRa WAN privata collegando i dispositivi e i gateway LoRa WAN AWS senza la necessità di sviluppare e gestire un server di rete LoRa WAN (LNS). I messaggi ricevuti dai dispositivi LoRa WAN vengono inviati al motore delle regole, dove possono essere formattati e inviati ad altri servizi. AWS IoT

Motore di regole

Il motore delle regole connette i dati dal broker di messaggi ad altri servizi AWS IoT per l'archiviazione e l'elaborazione aggiuntiva. Ad esempio, puoi inserire, aggiornare o eseguire le query su una tabella Dynamo DB oppure richiamare una funzione Lambda basata su un'espressione definita nel motore delle regole. Puoi usare un linguaggio basato su SQL per selezionare i dati dai payload dei messaggi e quindi elaborare e inviare i dati ad altri servizi, come Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB e AWS Lambda. Puoi anche creare delle regole che ripubblicano dei messaggi al broker di messaggi e ad altri sottoscrittori. Per ulteriori informazioni, consulta Regole per AWS IoT.

AWS IoT Core servizi di controllo

I servizi AWS IoT Core di controllo forniscono funzionalità di sicurezza, gestione e registrazione dei dispositivi.

Servizio di autenticazione personalizzata

Puoi definire autorizzazioni ad hoc che ti permettono di gestire la tua strategia di autenticazione e autorizzazione tramite un servizio di autenticazione personalizzato e una funzione Lambda. Gli autorizzatori personalizzati consentono di AWS IoT autenticare i dispositivi e autorizzare le operazioni utilizzando strategie di autenticazione e autorizzazione con token portatori.

Gli autorizzatori personalizzati possono implementare varie strategie di autenticazione; ad esempio, la verifica tramite token Web JSON o il callout del provider. OAuth Devono restituire documenti di policy che vengono utilizzati dal gateway del dispositivo per autorizzare le operazioni MQTT. Per ulteriori informazioni, consulta Autenticazione e autorizzazione personalizzata.

Servizio di provisioning dei dispositivi

Permette di effettuare il provisioning dei dispositivi tramite un modello che descrive le risorse necessarie per il dispositivo: un oggetto, un certificato e una o più policy. Un oggetto è una

voce nel registro contenente gli attributi che descrivono un dispositivo. I dispositivi utilizzano certificati con cui effettuare l'autenticazione. AWS IoT Le policy determinano le operazioni che un dispositivo può eseguire in AWS IoT.

I modelli contengono variabili che vengono sostituite da valori in un dizionario (mappa). Puoi usare lo stesso modello per effettuare il provisioning di più dispositivi semplicemente passando valori diversi per le variabili del modello nel dizionario. Per ulteriori informazioni, consulta Provisioning dei dispositivi.

Registro di gruppi

I gruppi ti permettono di gestire diversi dispositivi contemporaneamente classificandoli in gruppi. I gruppi possono contenere anche altri gruppi, quindi puoi creare una gerarchia di gruppi. Qualsiasi operazione eseguita su un gruppo padre verrà applicata ai rispettivi gruppi figlio. La stessa operazione si applica anche a tutti i dispositivi del gruppo padre e a tutti i dispositivi nei gruppi figlio. Le autorizzazioni concesse a un gruppo vengono applicate a tutti i dispositivi nel gruppo e in tutti i rispettivi gruppi figlio. Per ulteriori informazioni, consulta Gestione dei dispositivi con AWS IoT.

Servizio Jobs

Ti permette di definire un set di azioni remote inviate ed eseguite in uno o più dispositivi connessi a AWS IoT. Puoi ad esempio definire un processo che indichi a un set di dispositivi di scaricare e installare aggiornamenti per le applicazioni o il firmware, eseguire il riavvio, ruotare i certificati o eseguire operazioni di risoluzione dei problemi in remoto.

Per creare un processo, devi specificare una descrizione delle operazioni remote da eseguire e un elenco di target che devono eseguirle. I target possono essere singoli dispositivi, gruppi o entrambi. Per ulteriori informazioni, consulta AWS IoT Lavori.

Registro

Organizza le risorse associate a ogni dispositivo nel Cloud AWS . Puoi registrare i dispositivi e associare fino a tre attributi personalizzati a ognuno. È inoltre possibile associare certificati e client IDs MQTT a ciascun dispositivo per migliorare la capacità di gestione e risoluzione dei problemi. Per ulteriori informazioni, consulta Gestione dei dispositivi con AWS IoT.

Servizio di sicurezza e identità

Fornisce una responsabilità condivisa per la sicurezza nel cloud. AWS I dispositivi devono mantenere protette le proprie credenziali, in modo tale da poter inviare dati al broker di messaggi in tutta sicurezza. Il broker di messaggi e il motore di regole usano le funzionalità di sicurezza di

AWS per inviare dati in modo sicuro ai dispositivi o ad altri servizi AWS . Per ulteriori informazioni, consulta Autenticazione.

AWS IoT Core servizi dati

AWS IoT Core I servizi dati aiutano le tue soluzioni IoT a fornire un'esperienza applicativa affidabile anche con dispositivi che non sono sempre connessi.

Device Shadow

Documento JSON usato per archiviare e recuperare informazioni sullo stato corrente per un dispositivo.

Servizio Device Shadow

Il servizio Device Shadow mantiene lo stato di un dispositivo in modo che le applicazioni possano comunicare con un dispositivo indipendentemente dal fatto che il dispositivo sia connesso o meno. Quando un dispositivo è disconnesso, il servizio Device Shadow gestisce i dati per le applicazioni connesse. Quando il dispositivo si riconnette, sincronizza il suo stato con quello della sua shadow nel servizio Device Shadow. I dispositivi possono anche pubblicare il proprio stato corrente in una copia shadow, usata da applicazioni o altri dispositivi che potrebbero non essere sempre connessi. Per ulteriori informazioni, consulta AWS IoT Servizio Device Shadow.

AWS IoT Core servizio di supporto

Integrazione con Amazon Sidewalk per AWS IoT Core

Amazon Sidewalk è una rete condivisa che migliora le opzioni di connettività per consentire ai dispositivi di lavorare meglio insieme. Amazon Sidewalk supporta una vasta gamma di dispositivi per i clienti, come quelli che localizzano animali domestici o oggetti di valore, o che forniscono sicurezza domestica intelligente e controllo dell'illuminazione o, ancora, che forniscono diagnostica remota per elettrodomestici e strumenti. Amazon Sidewalk Integration for AWS IoT Core consente ai produttori di dispositivi di aggiungere la propria flotta di dispositivi Sidewalk al AWS IoT cloud.

Per ulteriori informazioni, consulta AWS IoT Core per Amazon Sidewalk.

Scopri di più su AWS IoT

Questo argomento ti aiuta a familiarizzare con il mondo di AWS IoT. È possibile ottenere informazioni generali su come le soluzioni IoT vengono applicate in vari casi d'uso, risorse di formazione, collegamenti ai social media AWS IoT e a tutti gli altri AWS servizi e un elenco di servizi e protocolli di comunicazione AWS IoT utilizzati.

Risorse di formazione per AWS IoT

Offriamo questi corsi di formazione per aiutarvi a conoscerli AWS IoT e ad applicarli alla progettazione della vostra soluzione.

- Introduzione a AWS IoT

  Una panoramica video dei servizi principali AWS IoT e dei relativi servizi.
- Approfondimento sull' AWS IoT autenticazione e l'autorizzazione

  Un corso avanzato che esplora i concetti di AWS IoT autenticazione e autorizzazione. Imparerai come autenticare e autorizzare i client ad accedere al piano di AWS IoT controllo e al piano dati. APIs
- Internet of Things Foundation Series

  Un percorso di apprendimento dei moduli di eLearning IoT su diverse tecnologie e funzionalità IoT.

AWS IoT risorse e guide

Si tratta di risorse tecniche approfondite su aspetti specifici di AWS IoT.

- IoT Lens — AWS IoT Well-Architected Framework

  Un documento che descrive le migliori pratiche per progettare le tue applicazioni IoT su AWS.
- Progettazione di argomenti MQTT per AWS IoT Core

  Un white paper che descrive le migliori pratiche per la progettazione di argomenti MQTT AWS IoT Core e l'utilizzo delle funzionalità di MQTT. AWS IoT Core
- Riassunto e introduzione

  Un documento PDF che descrive i diversi modi in cui è possibile AWS IoT fornire grandi flotte di dispositivi.

- AWS IoT Core Device Advisor

  AWS IoT Core Device Advisor fornisce test predefiniti che è possibile utilizzare per convalidare i dispositivi IoT per le migliori pratiche di connettività affidabili e sicure prima di implementare i dispositivi in produzione. AWS IoT Core

- AWS IoT Risorse

  Risorse specifiche dell'IoT, come guide tecniche, architetture di riferimento, eBook e post di blog curati, presentate in un indice di ricerca.

- Atlas IoT

  Panoramica su come risolvere i problemi comuni di progettazione IoT. Atlas IoT fornisce un'analisi approfondita delle sfide di progettazione che si rischia di incontrare durante lo sviluppo della soluzione IoT.

- AWS Whitepaper e guide

  La nostra attuale raccolta di white paper e guide su e altre tecnologie. AWS IoT AWS

AWS IoT nei social media

Questi canali di social media forniscono informazioni su AWS IoT argomenti AWS correlati.

- L'Internet of Things on AWS IoT — Blog ufficiale
- AWS IoT video nel canale Amazon Web Services su YouTube

Questi account di social media coprono tutti i AWS servizi, tra cui AWS IoT

- Il canale Amazon Web Services su YouTube
- Amazon Web Services su Twitter
- Amazon Web Services su Facebook
- Amazon Web Services su Instagram
- Amazon Web Services su LinkedIn

AWS servizi utilizzati dal motore delle AWS IoT Core regole

Il motore AWS IoT Core delle regole può connettersi a questi AWS servizi.

- Amazon DynamoDB

  Amazon DynamoDB è un servizio di database NoSQL scalabile che fornisce prestazioni veloci e prevedibili.

- Amazon Kinesis

  Amazon Kinesis semplifica la raccolta, l'elaborazione e l'analisi dei dati in streaming in tempo reale, in modo da ottenere informazioni dettagliate e tempestive e reagire rapidamente alle nuove informazioni. Amazon Kinesis può importare dati in tempo reale come video, audio, registri delle applicazioni, clickstream dei siti Web e dati di telemetria IoT per il machine learning, l'analisi dei dati e altre applicazioni.

- AWS Lambda

  AWS Lambda consente di eseguire il codice senza effettuare il provisioning o la gestione di server. Puoi configurare il codice in modo che venga attivato automaticamente da AWS IoT dati ed eventi o chiamarlo direttamente da un'app Web o mobile.

- Amazon Simple Storage Service

  Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) può archiviare e recuperare qualsiasi quantità di dati in qualsiasi momento, da qualsiasi punto del Web. AWS IoT le regole possono inviare dati ad Amazon S3 per l'archiviazione.

- Amazon Simple Notification Service

  Amazon Simple Notification Service (Amazon SNS) è un servizio web che consente ad applicazioni, utenti finali e dispositivi di inviare e ricevere notifiche dal cloud in modo immediato.

- Amazon Simple Queue Service

  Amazon Simple Queue Service (Amazon SQS) è un servizio di accodamento dei messaggi che disaccoppia e ridimensiona microservizi, sistemi distribuiti e applicazioni serverless.

- OpenSearch Servizio Amazon

  Amazon OpenSearch Service (OpenSearch Service) è un servizio gestito che semplifica la distribuzione, il funzionamento e la scalabilità OpenSearch, un popolare motore di ricerca e analisi open source.

- Amazon SageMaker AI

Amazon SageMaker AI può creare modelli di machine learning (ML) trovando modelli nei dati IoT. Il servizio utilizza questi modelli per elaborare nuovi dati e generare previsioni per la tua applicazione.

- Amazon CloudWatch

  Amazon CloudWatch offre una soluzione di monitoraggio affidabile, scalabile e flessibile per aiutarti a configurare, gestire e scalare i tuoi sistemi e la tua infrastruttura di monitoraggio.

Protocolli di comunicazione supportati da AWS IoT Core

Questi argomenti forniscono ulteriori informazioni sui protocolli di comunicazione usati da AWS IoT. Per ulteriori informazioni sui protocolli utilizzati dai dispositivi AWS IoT e servizi ai quali connettere dispositivi e servizi AWS IoT, vedereConnect a AWS IoT Core.

- MQTT (Message Queuing Telemetry Transport)

  La home page del sito MQTT.org dove è possibile trovare le specifiche del protocollo MQTT. Per ulteriori informazioni su come AWS IoT supporta MQTT, vedereMQTT.

- HTTPS (Hypertext Transfer Protocol - Secure)

  I dispositivi e le app possono accedere ai AWS IoT servizi tramite HTTPS.

- LoRaWAN (Long Range Wide Area Network)

  LoRaI dispositivi e i gateway WAN possono connettersi AWS IoT Core utilizzando AWS IoT Core for LoRa WAN.

- TLS (Transport Layer Security) v1.3

  Le specifiche del TLS v1.3 (RFC 5246). AWS IoT utilizza TLS v1.3 per stabilire connessioni sicure tra dispositivi e. AWS IoT

Cosa c'è di nuovo nella AWS IoT console

Stiamo aggiornando l'interfaccia utente della AWS IoT console per una nuova esperienza. L'interfaccia utente viene aggiornata per gradi, quindi alcune pagine della console avranno una nuova esperienza, alcune potrebbero avere sia l'esperienza originale che la nuova, alcune potrebbero avere solo l'esperienza originale.

Questa tabella mostra lo stato delle singole aree dell'interfaccia utente della AWS IoT console al 27 gennaio 2022.

AWS IoT stato dell'interfaccia utente della console

Pagina della console	Esperienza originale	Esperienza nuova	Commenti
Monitor (Monitoraggio)	Non disponibile	Disponibilità	
Attività	Non disponibile	Disponibilità	
Onboard - Inizia	Non disponibile	Disponibilità	Non è disponibile nelle regioni CN
Onboard - Modelli di provisioning del parco istanze	Disponibilità	Disponibilità	
Gestione - Oggetti	Disponibilità	Disponibilità	
Gestione - Tipi	Disponibilità	Disponibilità	
Gestione - Gruppi di oggetti	Disponibilità	Disponibilità	
Gestione - Gruppi di fatturazione	Disponibilità	Disponibilità	
Gestione - Processi	Disponibilità	Disponibilità	
Gestione - Modelli di processi	Non disponibile	Disponibilità	
Gestione - Tunnel	Non disponibile	Disponibilità	
Hub del parco istanze - Inizia	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS
Hub del parco istanze - Applicazioni	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS

Pagina della console	Esperienza originale	Esperienza nuova	Commenti
Greengrass - Nozioni di base	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS
Greengrass - Dispositivi di base	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS
Greengrass - Componenti	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS
Greengrass - Implementazioni	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS
Greengrass - Classico (V1)	Disponibilità	Disponibilità	
Connettività Wireless - Introduzione	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS
Connettività Wireless - Gateway	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS
Connettività Wireless - Dispositivi	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS
Connettività Wireless - Profili	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS
Connettività Wireless - Destinazioni	Non disponibile	Disponibilità	Non disponibile in tutto Regioni AWS
Sicurezza - Certificati	Disponibilità	Disponibilità	
Sicurezza - Policy	Disponibilità	Disponibilità	
Sicuro - CAs	Disponibilità	Disponibilità	
Sicurezza - Alias di ruolo	Disponibilità	Disponibilità	

Pagina della console	Esperienza originale	Esperienza nuova	Commenti
Sicurezza - Autorizzazione	Disponibilità	Disponibilità	
Tutela - Introduzione	Non disponibile	Disponibilità	
Tutela - Verifica	Non disponibile	Disponibilità	
Tutela - Rilevamento	Non disponibile	Disponibilità	
Tutela - Operazioni di mitigazione	Non disponibile	Disponibilità	
Tutela - Impostazioni	Non disponibile	Disponibilità	
Atto - Regole	Disponibilità	Disponibilità	
Atto - Destinazioni	Disponibilità	Disponibilità	
Test - Device Advisor	Disponibilità	Disponibilità	Non disponibile in tutto Regioni AWS
Test - Client di test MQTT	Disponibilità	Disponibilità	
Software	Disponibilità	Disponibilità	
Settings (Impostazioni)	Non disponibile	Disponibilità	
Learn (Guida)	Disponibilità	Non ancora disponibile	

Legenda

Valori di stato

- Disponibilità

Questa esperienza di interfaccia utente può essere utilizzata.

- Non disponibile

  Questa esperienza di interfaccia utente non può essere utilizzata.

- Non ancora disponibile

  La nuova esperienza dell'interfaccia utente è in fase di sviluppo, ma non è ancora pronta.

- In corso

  La nuova esperienza dell'interfaccia utente è in fase di aggiornamento. Tuttavia, alcune pagine potrebbero ancora avere l'esperienza utente originale.

Utilizzo AWS IoT con un SDK AWS

AWS i kit di sviluppo software (SDKs) sono disponibili per molti linguaggi di programmazione più diffusi. Ogni SDK fornisce un'API, esempi di codice e documentazione che facilitano agli sviluppatori la creazione di applicazioni nel loro linguaggio preferito.

Documentazione sugli SDK	Esempi di codice
AWS SDK per C++	AWS SDK per C++ esempi di codice
AWS CLI	AWS CLI esempi di codice
AWS SDK per Go	AWS SDK per Go esempi di codice
AWS SDK per Java	AWS SDK per Java esempi di codice
AWS SDK per JavaScript	AWS SDK per JavaScript esempi di codice
AWS SDK per Kotlin	AWS SDK per Kotlin esempi di codice
AWS SDK per .NET	AWS SDK per .NET esempi di codice
AWS SDK per PHP	AWS SDK per PHP esempi di codice
AWS Strumenti per PowerShell	AWS Strumenti per PowerShell esempi di codice

Documentazione sugli SDK	Esempi di codice
AWS SDK per Python (Boto3)	AWS SDK per Python (Boto3) esempi di codice
AWS SDK per Ruby	AWS SDK per Ruby esempi di codice
AWS SDK per Rust	AWS SDK per Rust esempi di codice
AWS SDK per SAP ABAP	AWS SDK per SAP ABAP esempi di codice
AWS SDK per Swift	AWS SDK per Swift esempi di codice

Esempio di disponibilità

Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci un feedback nella parte inferiore di questa pagina.

Guida introduttiva ai AWS IoT Core tutorial

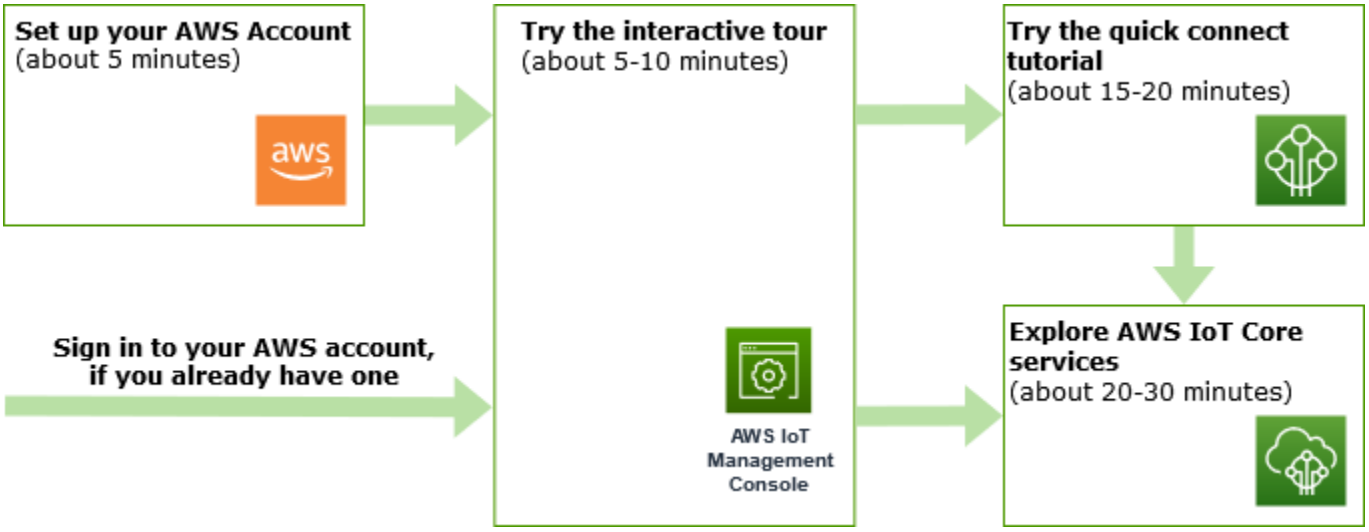
Che tu sia alle prime armi con l'IoT o che tu abbia anni di esperienza, queste risorse presentano i AWS IoT concetti e i termini che ti aiuteranno a iniziare a utilizzarlo AWS IoT.

- Guarda all'interno AWS IoT e ai suoi componentiCome AWS IoT funziona.
- Ulteriori informazioni su AWS IoT dalla nostra collezione di materiali e video per la formazione. Questo argomento include anche un elenco di servizi a cui AWS IoT può connettersi, collegamenti ai social media e alle specifiche del protocollo di comunicazione.
- the section called “Connect il tuo primo dispositivo a AWS IoT Core”.
- Sviluppa le tue soluzioni IoT con Connect a AWS IoT Core ed esplorando AWS IoT tutorial.
- Testa e convalida i dispositivi IoT per una comunicazione sicura e affidabile utilizzando Device Advisor.
- Gestisci la tua soluzione utilizzando il gestore di servizi AWS IoT Core come Indicizzazione del parco istanze, AWS IoT Lavori, e AWS IoT Device Defender.
- Analizza i dati dai tuoi dispositivi utilizzando AWS IoT servizi dati.

Connect il tuo primo dispositivo a AWS IoT Core

AWS IoT Core i servizi connettono i dispositivi IoT a AWS IoT servizi e altri AWS servizi. AWS IoT Core include il device gateway e il message broker, che connettono ed elaborano i messaggi tra i dispositivi IoT e il cloud.

Ecco come puoi iniziare con AWS IoT Core e AWS IoT.



Questa sezione presenta una panoramica dei AWS IoT Core suoi servizi principali e fornisce diversi esempi di come connettere un dispositivo AWS IoT Core e passare messaggi tra di essi. Lo scambio di messaggi tra dispositivi e cloud è fondamentale per ogni soluzione IoT ed è il modo in cui i dispositivi possono interagire con altri AWS servizi.

- Configurare Account AWS

  Prima di poter utilizzare AWS IoT i servizi, è necessario configurare un Account AWS. Se hai già un utente IAM Account AWS e un utente IAM, puoi utilizzarli e saltare questo passaggio.

- Prova il tutorial sulla connessione rapida

  Questo tutorial è ideale se vuoi iniziare rapidamente AWS IoT e vedere come funziona in uno scenario limitato. In questo tutorial, avrai bisogno di un dispositivo e installerai del AWS IoT software su di esso. Se non disponi di un dispositivo IoT, puoi utilizzare un personal computer Windows, Linux o macOS come dispositivo per questo tutorial. Se vuoi provare AWS IoT, ma non hai un dispositivo, prova l'opzione successiva.

- Prova il tutorial interattivo

  Questa demo è ideale se vuoi vedere cosa può fare una AWS IoT soluzione di base senza connettere un dispositivo o scaricare alcun software. Il tutorial interattivo presenta una soluzione simulata basata su AWS IoT Core servizi che illustra come interagiscono.

- Esplora i AWS IoT Core servizi con un tutorial pratico

  Questo tutorial è ideale per gli sviluppatori che vogliono iniziare e continuare a esplorare altre AWS IoT Core funzionalità come il motore delle regole e le ombre. AWS IoT Questo tutorial segue un processo simile al tutorial di connessione rapida, ma fornisce ulteriori dettagli su ogni passaggio per consentire una transizione più fluida ai tutorial più avanzati.

- Visualizza i messaggi MQTT con il AWS IoT client MQTT

  Scopri come utilizzare il client di test MQTT per guardare il tuo primo dispositivo pubblicare messaggi MQTT in AWS IoT. Il client di test MQTT è uno strumento utile per monitorare e risolvere i problemi relativi alle connessioni dei dispositivi.

> **Note**
>
> Se desideri provare più di uno di questi tutorial introduttivi o ripetere lo stesso tutorial, devi eliminare l'oggetto creato da un tutorial precedente prima di avviarne un altro. Se non elimini

l'oggetto da un tutorial precedente, sarà necessario utilizzare un nome diverso per i tutorial successivi. Questo perché il nome dell'oggetto deve essere univoco nel tuo account e nella regione Regione AWS.

Per ulteriori informazioni su AWS IoT Core, consulta What Is? AWS IoT Core

Configurare Account AWS

Prima di AWS IoT Core utilizzarlo per la prima volta, completa le seguenti attività:

Argomenti

- Registrati per un Account AWS
- Crea un utente con accesso amministrativo
- Apri la console AWS IoT

Registrati per un Account AWS

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a un Account AWS

1. Apri la https://portal.aws.amazon.com/billing/registrazione.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata o un messaggio di testo e ti verrà chiesto di inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, Utente root dell'account AWSviene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

AWS ti invia un'email di conferma dopo il completamento della procedura di registrazione. In qualsiasi momento, puoi visualizzare l'attività corrente del tuo account e gestirlo accedendo a https://aws.amazon.com/e scegliendo Il mio account.

Crea un utente con accesso amministrativo

Dopo esserti registrato Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi Console di gestione AWScome proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Accedere come utente root nella Guida per l'utente di Accedi ad AWS .
2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

   Per istruzioni, consulta Abilitare un dispositivo MFA virtuale per l'utente Account AWS root (console) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita il Centro identità IAM.

   Per istruzioni, consulta Abilitazione del AWS IAM Identity Center nella Guida per l'utente di AWS IAM Identity Center .
2. Nel Centro identità IAM, assegna l'accesso amministrativo a un utente.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere come utente del Centro identità IAM, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente del Centro identità IAM.

  Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta AWS Accedere al portale di accesso nella Guida per l'Accedi ad AWS utente.

Assegnazione dell'accesso ad altri utenti

1. Nel Centro identità IAM, crea un set di autorizzazioni conforme alla best practice per l'applicazione di autorizzazioni con il privilegio minimo.

   Segui le istruzioni riportate nella pagina Creazione di un set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

   Per istruzioni, consulta Aggiungere gruppi nella Guida per l'utente di AWS IAM Identity Center .

- Apri la console AWS IoT

Se hai già un utente Account AWS e un utente per te, puoi usarli e passare direttamente athe section called "Apri la console AWS IoT".

Apri la console AWS IoT

La maggior parte degli argomenti relativi alla console di questa sezione iniziano dalla AWS IoT console. Se non hai già effettuato l'accesso al tuo Account AWS, accedi, quindi apri la AWS IoT console e passa alla sezione successiva per continuare a usare. AWS IoT

Tutorial interattivo

Il tutorial interattivo mostra i componenti di una semplice soluzione IoT basata su AWS IoT. Il tutorial mostra come i dispositivi IoT interagiscono con AWS IoT Core i servizi. Questo argomento fornisce un'anteprima del tutorial AWS IoT Core interattivo.

> **Note**
>
> Le immagini nella console includono animazioni che non appaiono nelle immagini di questo tutorial.

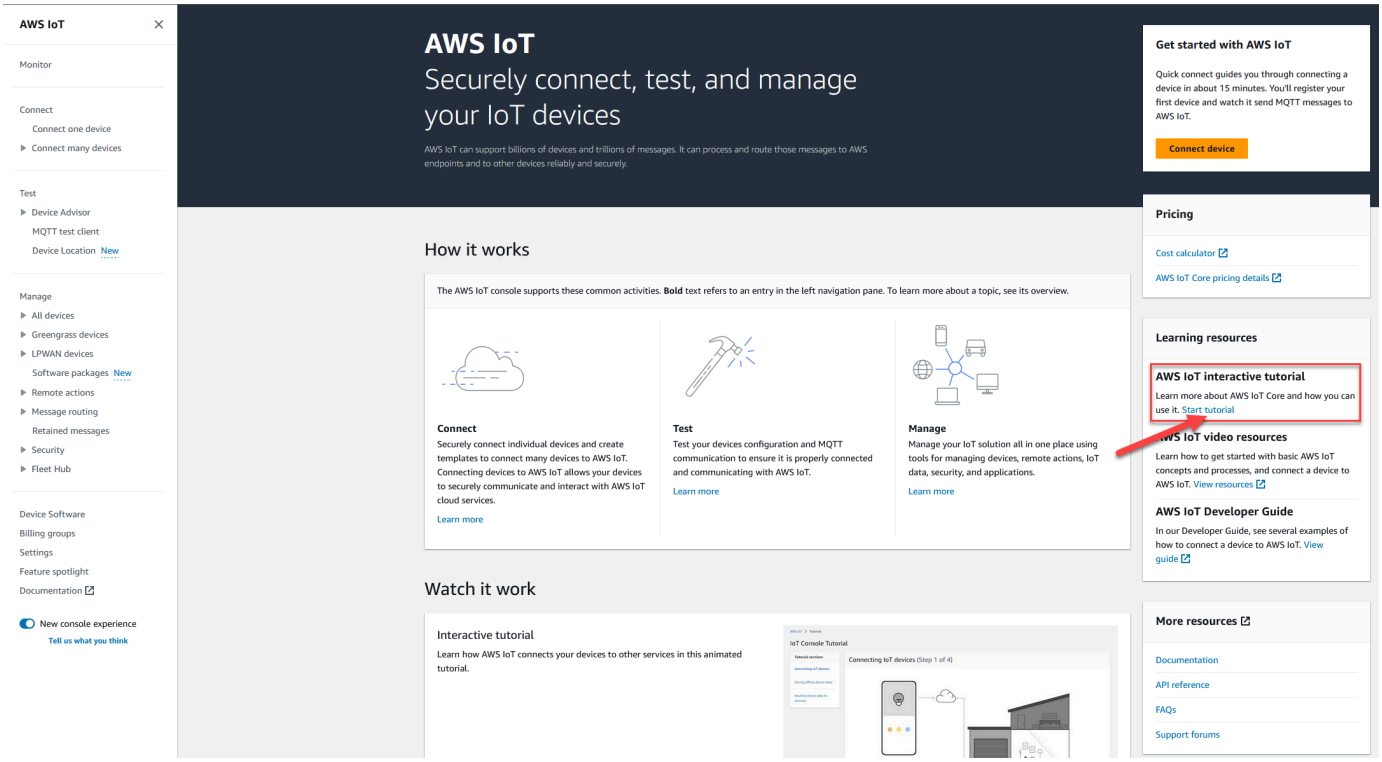
Per eseguire la demo, devi per prima cosa the section called "Configurare Account AWS". Il tutorial, tuttavia, non richiede AWS IoT risorse, software aggiuntivo o codifica.

La demo impiega circa 5-10 minuti. Concediti 10 minuti in modo da avere più tempo per comprendere ciascuno dei passaggi.

Per eseguire il tutorial AWS IoT Core interattivo

1. Apri la AWS IoT home page nella AWS IoT console.

   Nella home page di AWS IoT , nel riquadro della finestra Risorse di apprendimento, scegli Avvia tutorial.



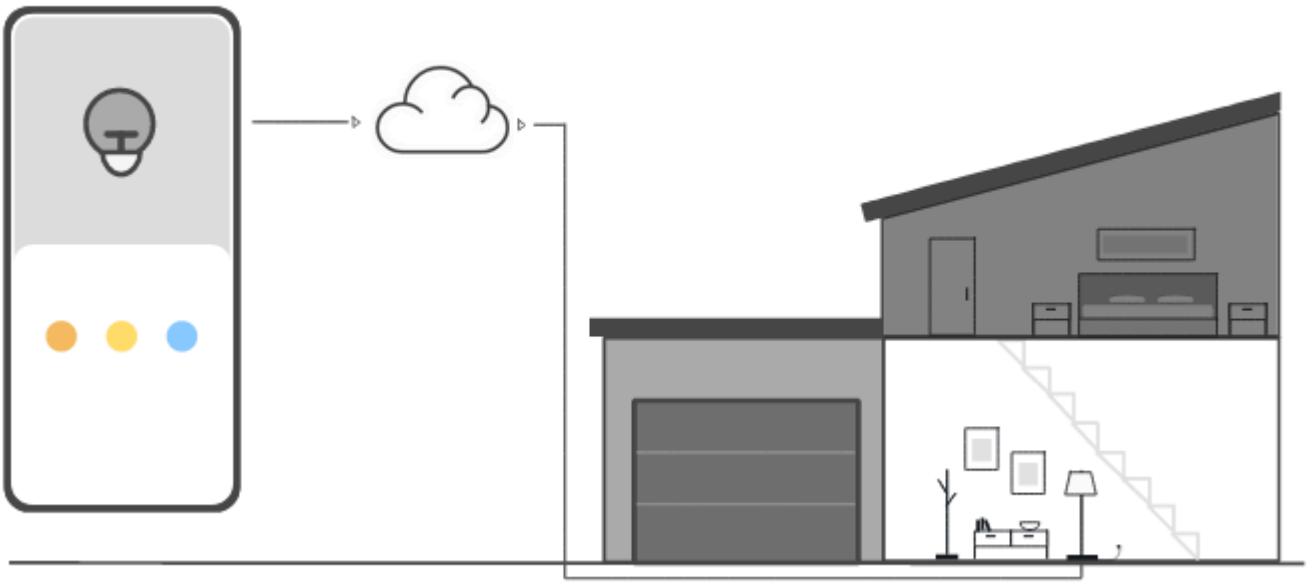
2. Nella pagina Tutorial della console AWS IoT , consulta le sezioni del tutorial e scegli la sezione Avvia quando sei pronto per continuare.

Le seguenti sezioni descrivono come il AWS IoT Console Tutorial presenta queste AWS IoT Core funzionalità:

- Collegamento di dispositivi IoT
- Salvataggio dello stato del dispositivo non in linea
- Instradamento dei dati del dispositivo ai servizi

Collegamento di dispositivi IoT

Scopri come comunicano i dispositivi IoT con AWS IoT Core.

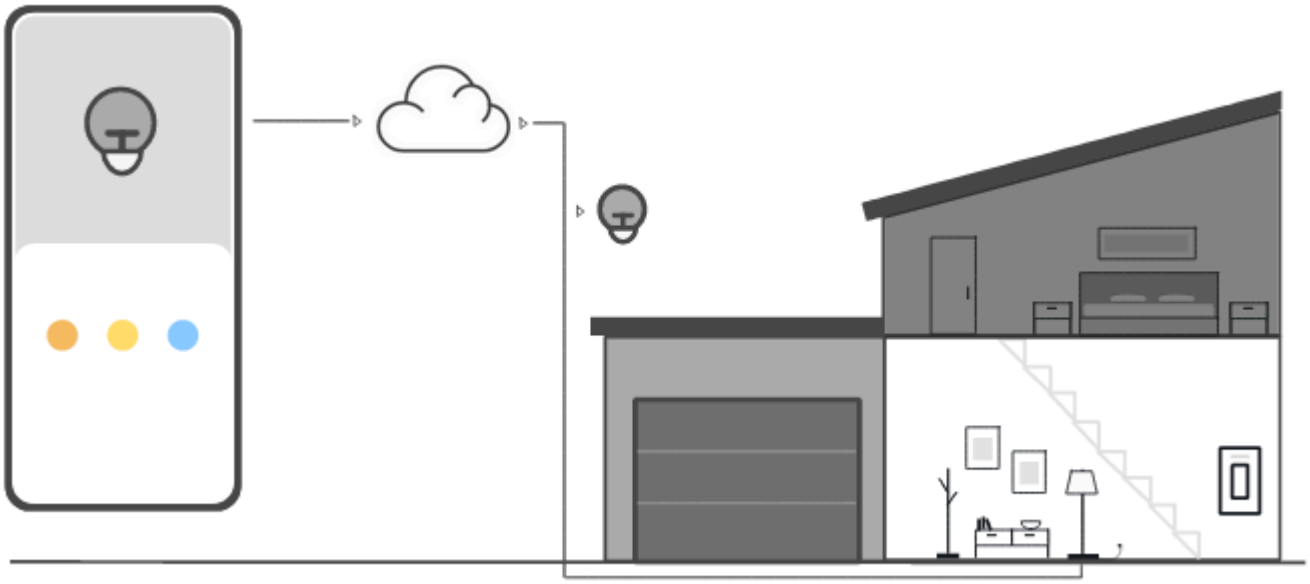


L'animazione in questa fase mostra come due dispositivi, il dispositivo di controllo a sinistra e una lampada intelligente nella casa a destra, si connettono e comunicano con AWS IoT Core nel cloud. L'animazione mostra i dispositivi che comunicano AWS IoT Core e reagiscono ai messaggi che ricevono.

Per ulteriori informazioni sulla connessione dei dispositivi a AWS IoT Core, vedere. Connect a AWS IoT Core

Salvataggio dello stato del dispositivo non in linea

Scopri come AWS IoT Core salvare lo stato del dispositivo per quando un dispositivo o un'app sono offline.



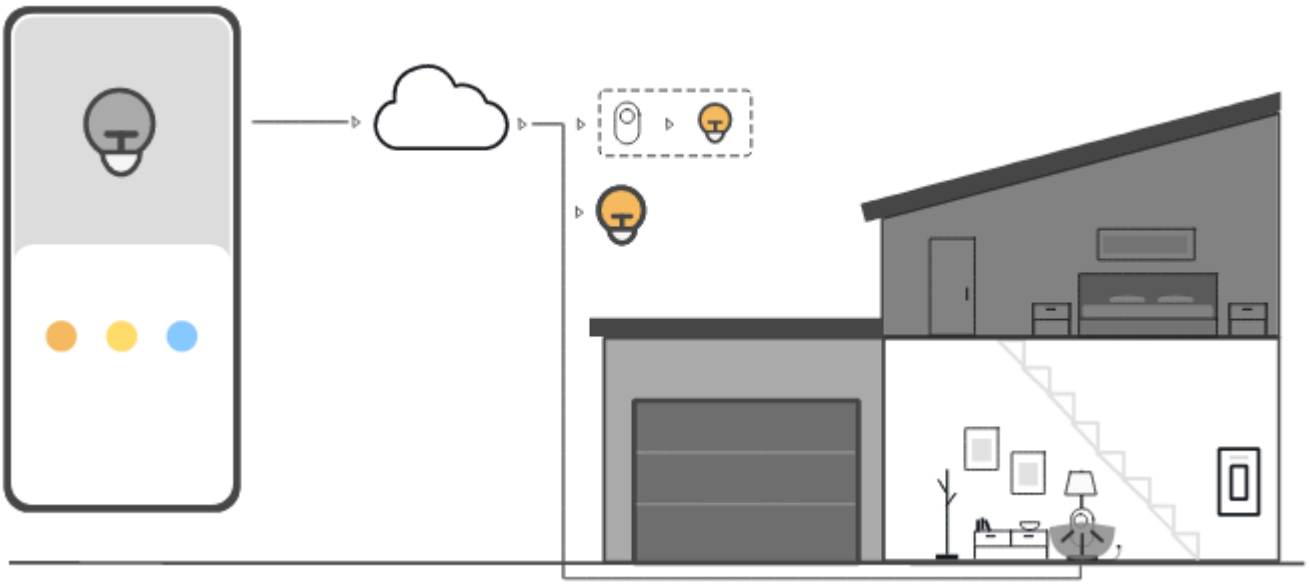
L'animazione di questo passaggio mostra come il servizio Device Shadow AWS IoT Core salva le informazioni sullo stato del dispositivo per il dispositivo di controllo e la lampada intelligente. Mentre la lampada intelligente è disconnessa, Device Shadow salva i comandi del dispositivo di controllo.

Quando la lampada intelligente si riconnette a AWS IoT Core, recupera quei comandi. Mentre la lampada intelligente è disconnessa, Device Shadow salva le informazioni di stato della lampada intelligente. Quando il dispositivo di controllo si riconnette, recupera lo stato corrente della lampada intelligente per aggiornare il display.

Per ulteriori informazioni su Device Shadows, consulta AWS IoT Servizio Device Shadow.

Instradamento dei dati del dispositivo ai servizi

Scopri come AWS IoT Core invia lo stato del dispositivo ad altri AWS servizi.



L'animazione di questo passaggio mostra come AWS IoT Core invia i dati dai dispositivi ad altri AWS servizi utilizzando AWS IoT le regole. AWS IoT le regole sottoscrivono messaggi specifici provenienti dai dispositivi, interpretano i dati contenuti in tali messaggi e indirizzano i dati interpretati ad altri servizi. In questo esempio, una AWS IoT regola interpreta i dati da un sensore di movimento e invia comandi a un Device Shadow, che poi li invia alla lampadina intelligente. Come nell'esempio precedente, Device Shadow archivia le informazioni sullo stato del dispositivo per il dispositivo di controllo.

Per ulteriori informazioni sulle AWS IoT regole, vedereRegole per AWS IoT.

Prova il tutorial sulla connessione AWS IoT Core rapida

In questo tutorial creerai come prima cosa il tuo oggetto, poi gli collegherai un dispositivo e lo guarderai inviare messaggi MQTT.

Puoi aspettarti di impiegare 15-20 minuti per questo tutorial.

Questo tutorial è ideale per chi vuole iniziare rapidamente AWS IoT a vedere come funziona in uno scenario limitato. Se stai cercando un esempio che ti consenta di iniziare, in modo da poter esplorare più funzionalità e servizi, prova AWS IoT Core Scopritelo in tutorial pratici.

In questo tutorial, scaricherai ed eseguirai il software su un dispositivo che si connette a una risorsa oggetto AWS IoT Core come parte di una soluzione IoT molto piccola. Il dispositivo può essere un dispositivo IoT, come un Raspberry Pi, oppure può anche essere un computer che esegue Linux, OS

e OSX o Windows. Se stai cercando di connettere un dispositivo WAN a lungo raggio (LoRaWAN) AWS IoT, consulta il tutorial >Connessione di dispositivi e gateway a AWS IoT Core una rete WAN. LoRa

Se il dispositivo supporta un browser in grado di eseguire la Console AWS IoT, ti consigliamo di completare questo tutorial su quel dispositivo.

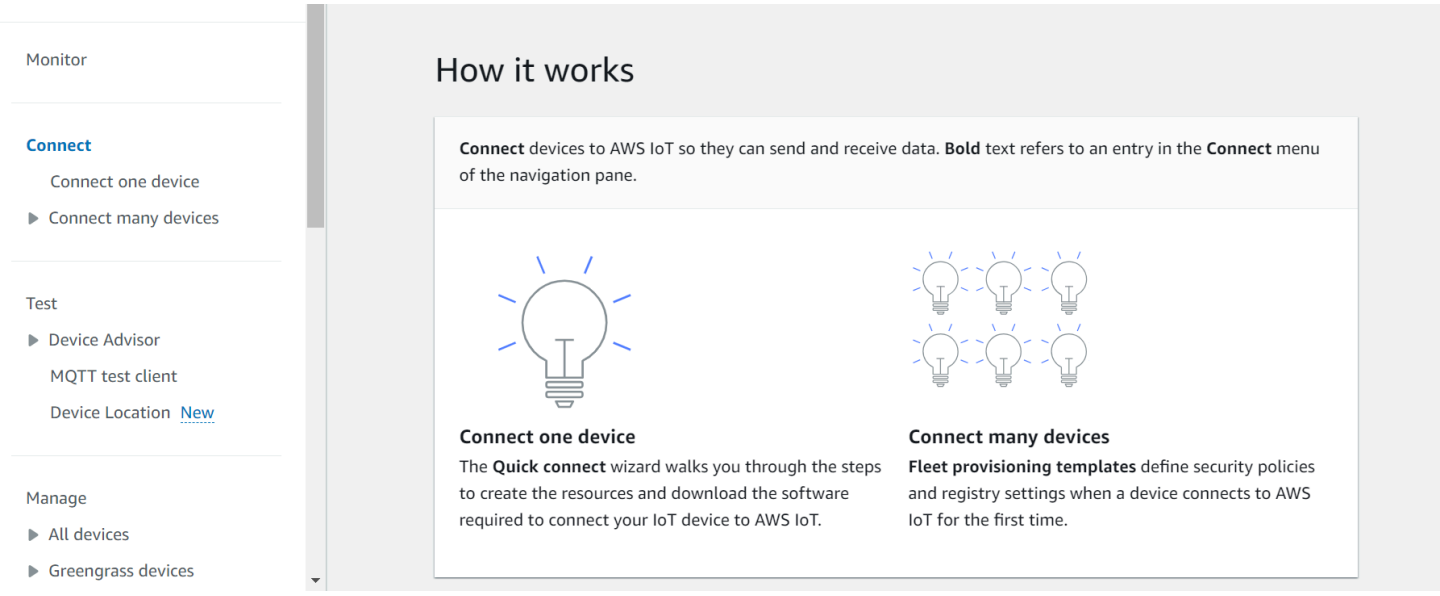
> **Note**
>
> Se il dispositivo non dispone di un browser compatibile, segui questo tutorial su un computer. Quando la procedura richiede di scaricare il file, scaricalo sul computer e quindi trasferisci il file scaricato sul dispositivo utilizzando Secure Copy (SCP) o un processo simile.

Il tutorial richiede che il dispositivo IoT comunichi con la porta 8443 sull'endpoint dati dispositivo dell' Account AWS. Per verificare se è in grado di accedere a quella porta, utilizza le procedure in Verifica la connettività con l'endpoint di dati del tuo dispositivo.

Passaggio 1. Inizia il tutorial

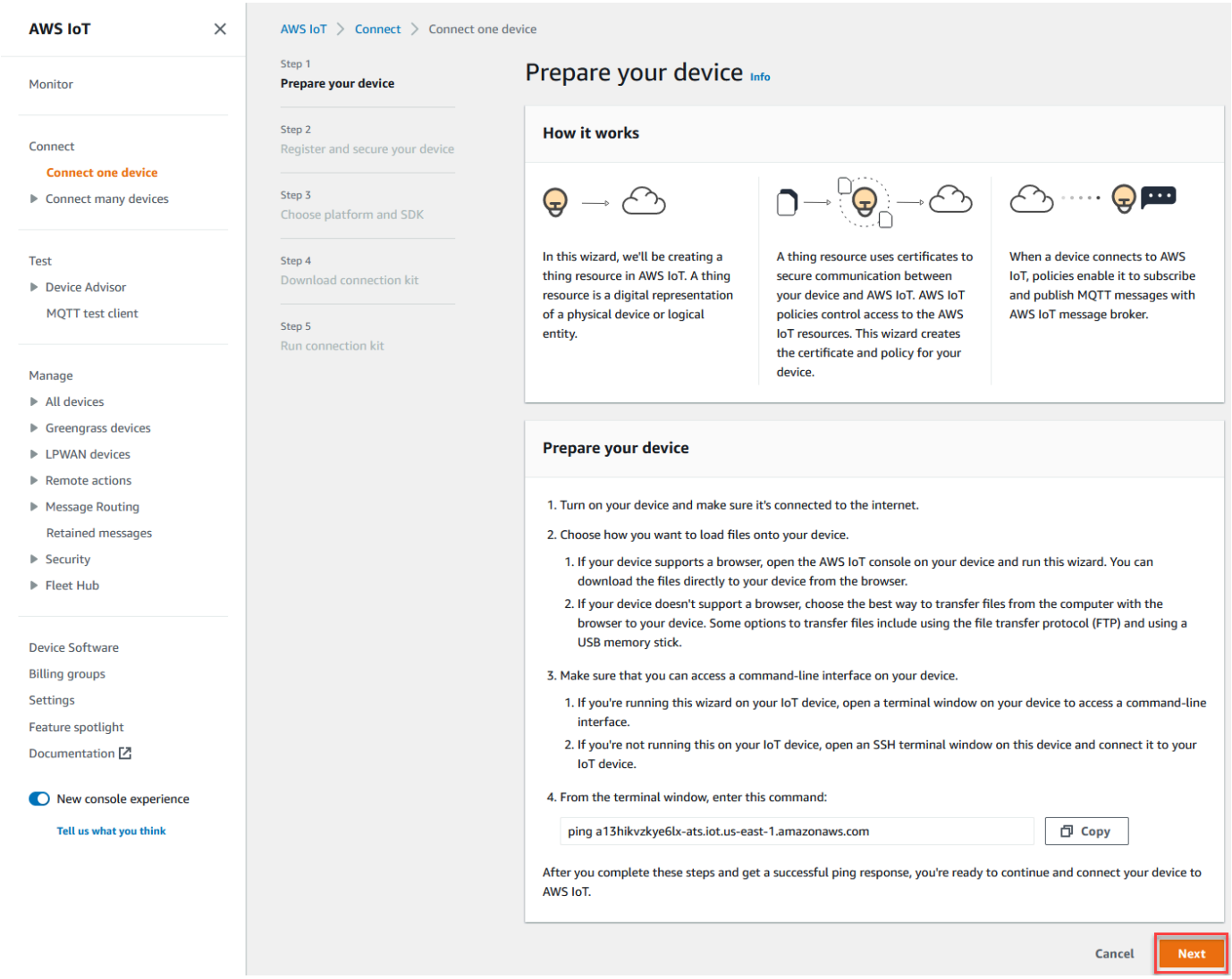
Se possibile, completa questa procedura sul dispositivo; in caso contrario, dovrai trasferire un file sul dispositivo più avanti in questa procedura.

Per iniziare il tutorial, accedi alla console AWS IoT. Nella home page della AWS IoT console, a sinistra, scegli Connetti, quindi scegli Connetti un dispositivo.



Passaggio 2. Crea un oggetto

1. Nella sezione Prepare your device (Preparazione del dispositivo), seguire le istruzioni a video per preparare il dispositivo per la connessione a AWS IoT.



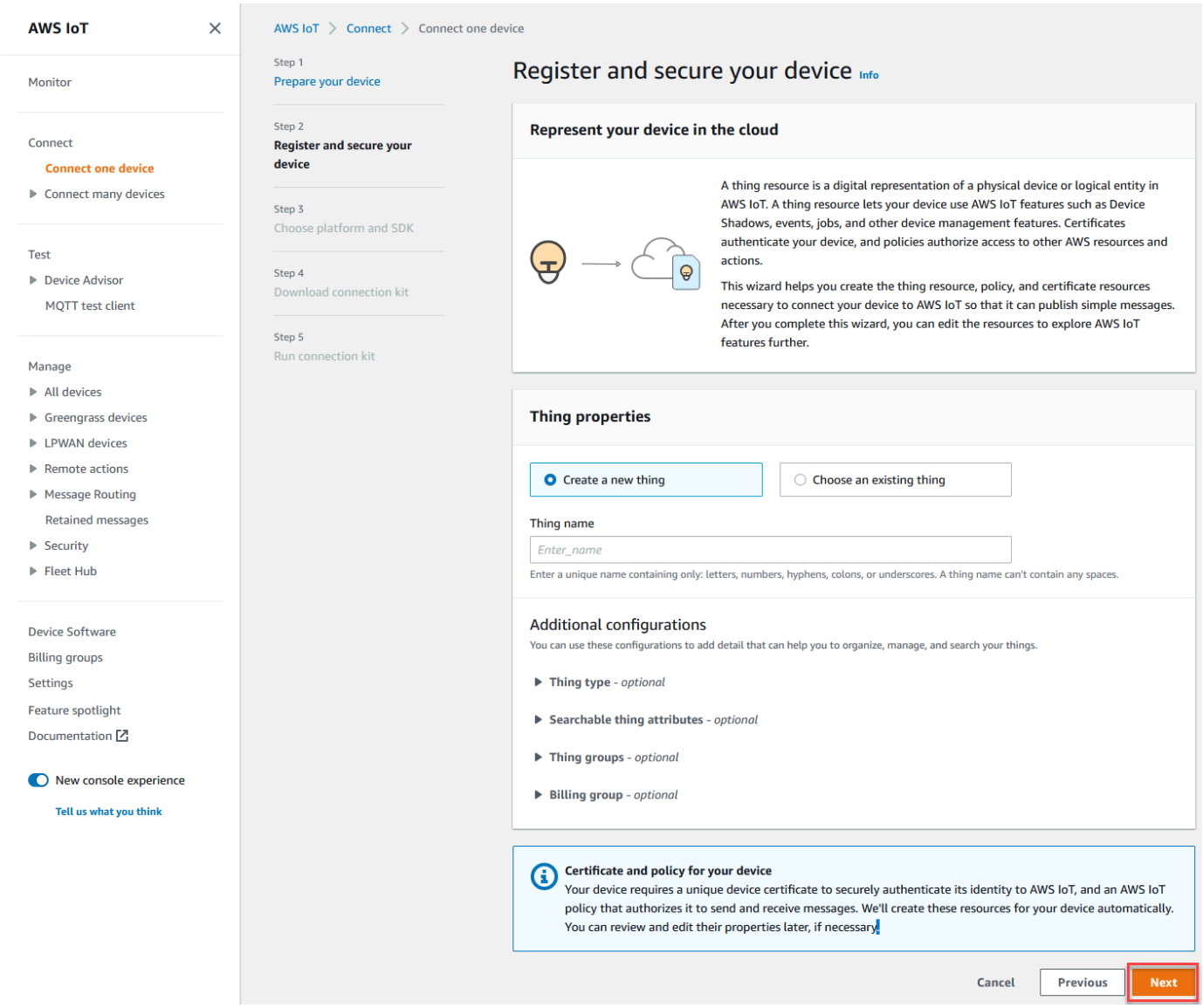
2. Nella sezione Register and secure your device (Registra e proteggi il dispositivo), scegliere Create a new thing (Crea un nuovo oggetto) o Choose an existing thing (Scegli un oggetto esistente). Nel campo Thing name (Nome oggetto), immettere il nome dell'oggetto. Il nome dell'oggetto usato in questo esempio è **`TutorialTestThing`**

> **Important**
>
> Controlla il nome dell'oggetto prima di continuare.

> Un nome di oggetto non può essere modificato dopo che l'oggetto è stato creato. Se desideri cambiare il nome di un oggetto, devi creare un nuovo oggetto con il nome corretto dell'oggetto e quindi eliminare quello con il nome non corretto.

Nella sezione Additional configurations (Configurazioni aggiuntive), personalizzare ulteriormente la risorsa oggetto utilizzando le configurazioni opzionali elencate.

Dopo aver dato un nome all'oggetto e aver selezionato eventuali configurazioni aggiuntive, scegliere Next (Avanti).

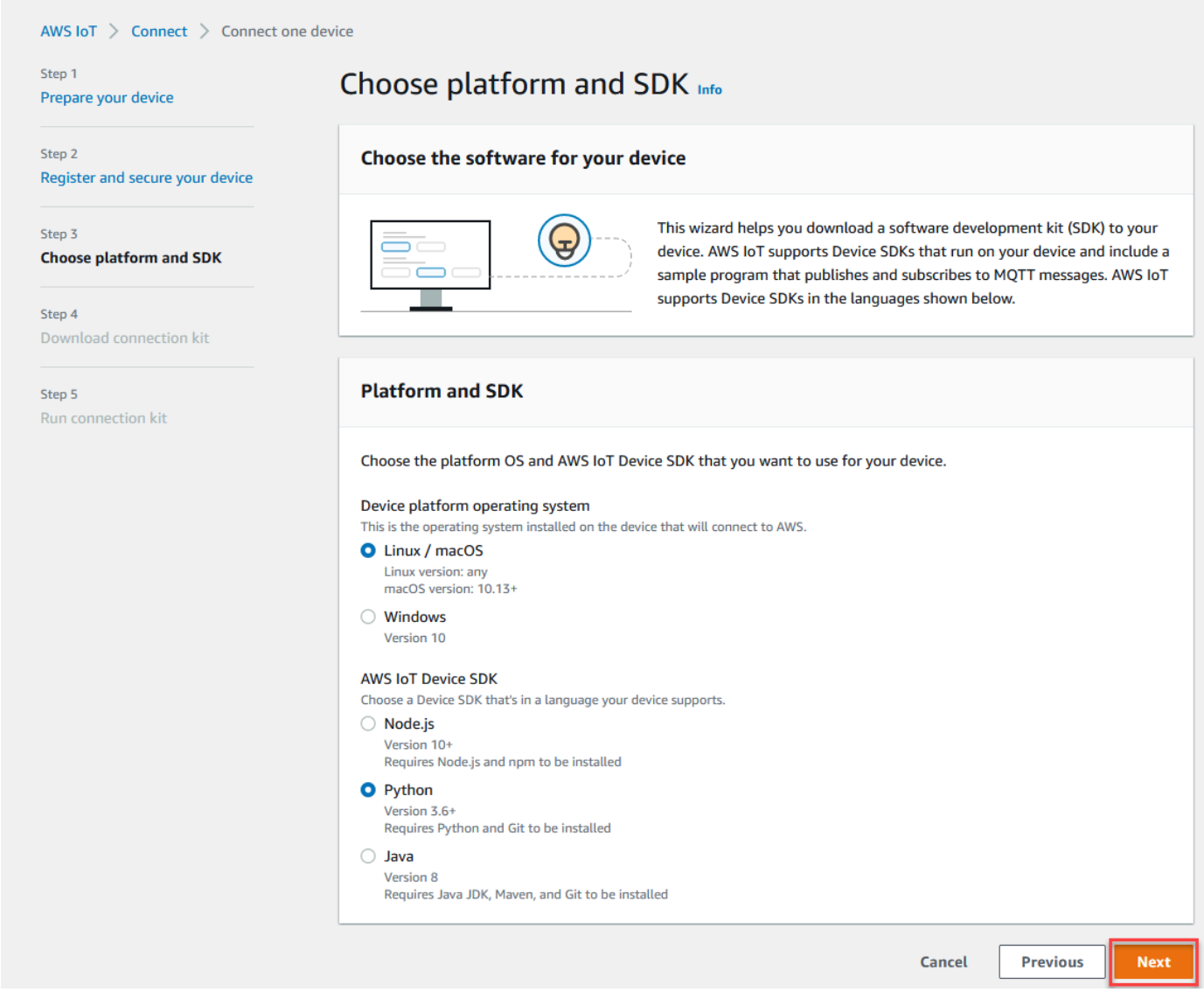


3. Nella sezione Scegli piattaforma e SDK, scegli la piattaforma e la lingua dell'SDK del AWS IoT dispositivo che desideri utilizzare. Questo esempio utilizza la Linux/OSX piattaforma e l'SDK

Python. Assicurati che python3 e pip3 siano installati sul dispositivo di destinazione prima di passare alla fase successiva.

> **Note**
>
> Assicurati di controllare l'elenco dei prerequisiti software richiesti dall'SDK scelto, nella parte inferiore della pagina della console.
> Prima di continuare con la fase successiva, devi disporre del software necessario installato nel computer di destinazione.

Dopo aver scelto la lingua SDK della piattaforma e del dispositivo, scegli Next (Successivo).

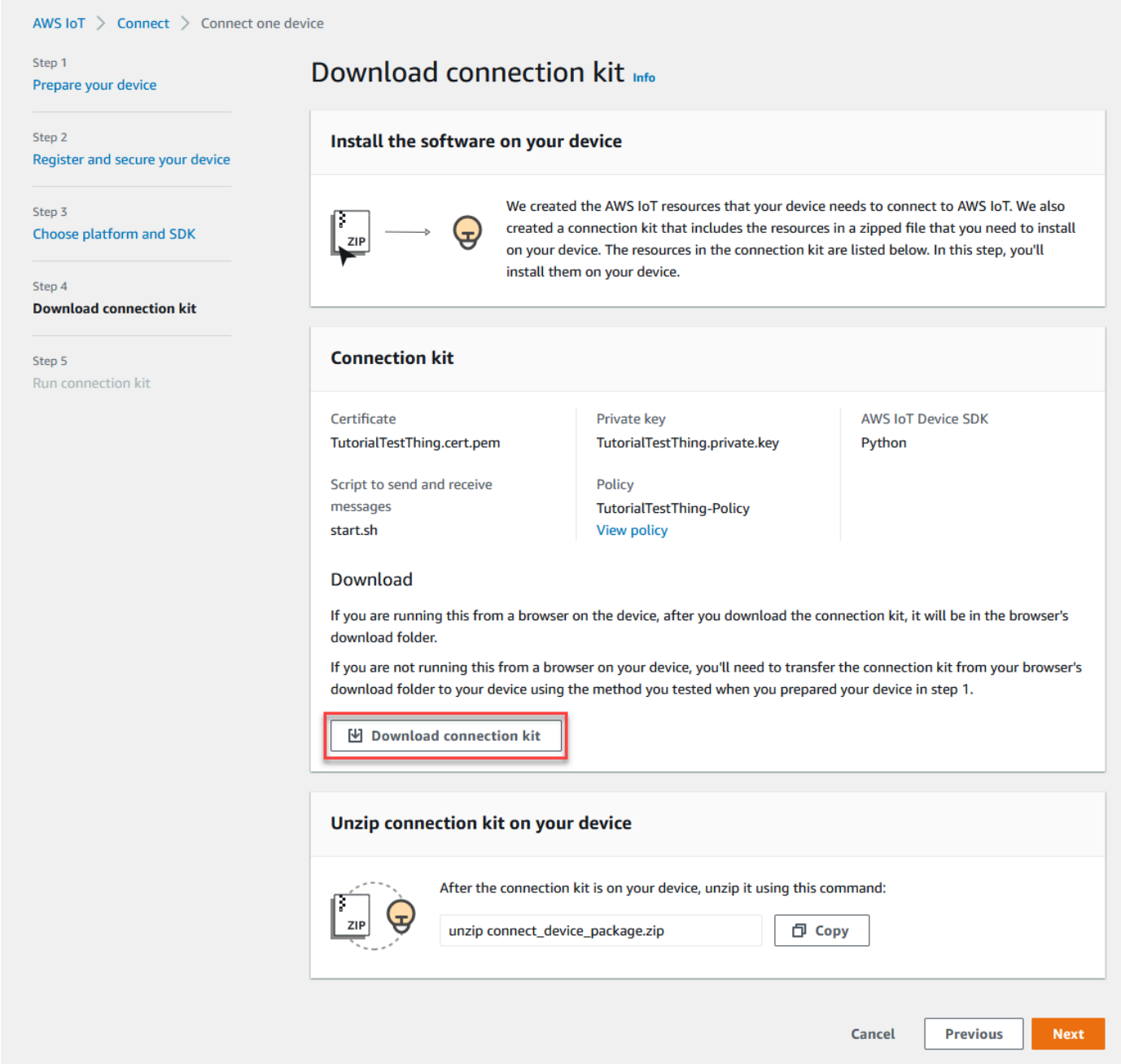


Fase 3. Scarica file sul tuo dispositivo

Questa pagina viene visualizzata dopo AWS IoT aver creato il kit di connessione, che include i seguenti file e risorse richiesti dal dispositivo:

- I file di certificato dell'oggetto utilizzati per autenticare il dispositivo
- Una risorsa policy per autorizzare il tuo oggetto a interagire con AWS IoT
- Lo script per scaricare AWS Device SDK ed eseguire il programma di esempio sul dispositivo

1. Quando sei pronto a continuare, scegli l'opzione Download connection kit for (Scarica il kit di connessione per) per scaricare il kit di connessione per la piattaforma scelta in precedenza.



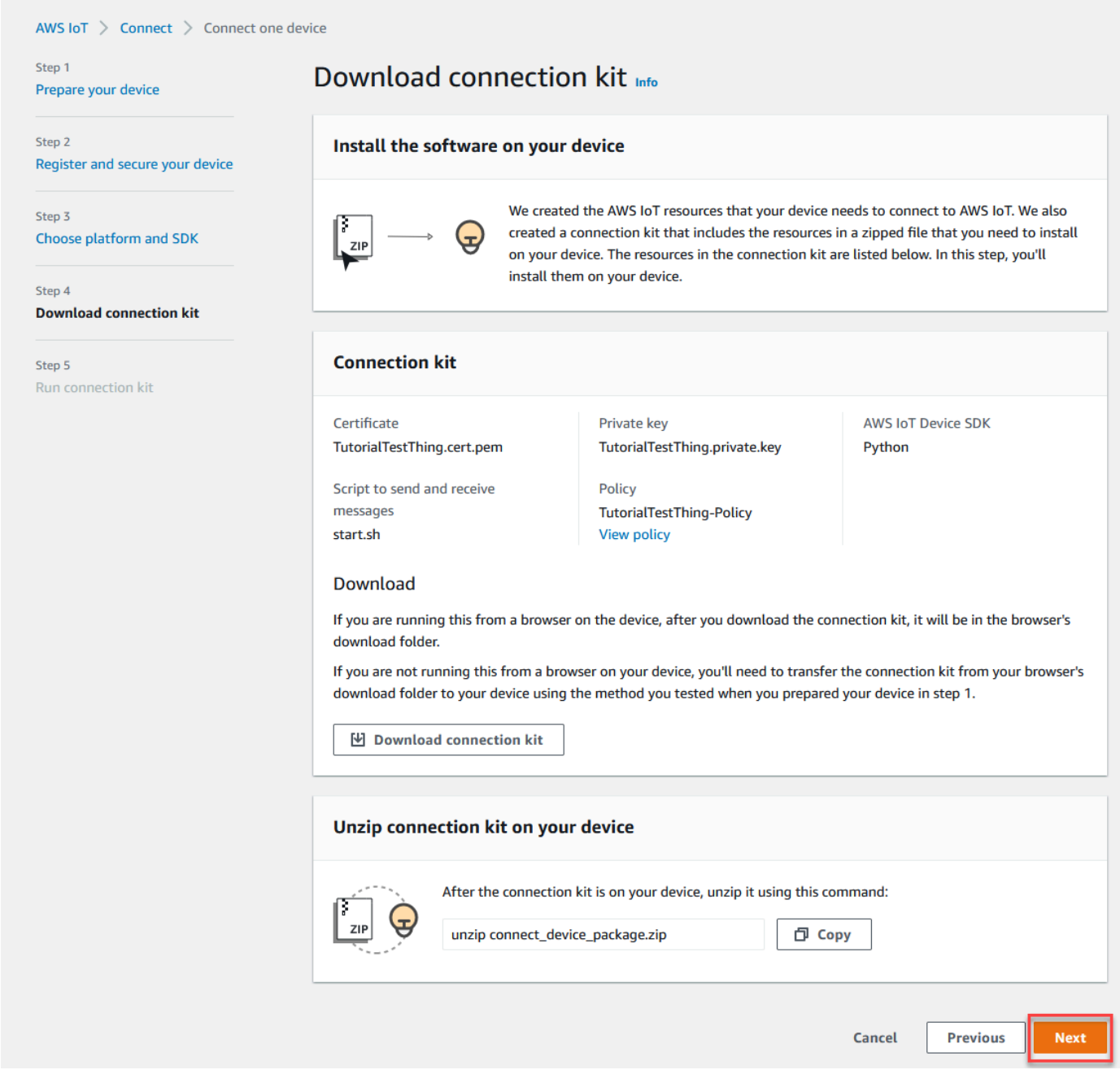
2. Se esegui questa procedura sul dispositivo, salva il file del kit di connessione in una directory da cui è possibile eseguire i comandi della riga di comando.

   Se non esegui questa procedura sul dispositivo, salva il file del kit di connessione in una directory locale e quindi trasferisci il file sul dispositivo.

3. Nella sezione Unzip connection kit on your device (Decomprimi il kit di connessione sul dispositivo), immettere unzip connect_device_package.zip nella directory in cui si trovano i file del kit di connessione.

Se utilizzi una finestra di PowerShell comando di Windows e il unzip comando non funziona, sostituiscilo unzip con expand-archive e riprova a utilizzare la riga di comando.

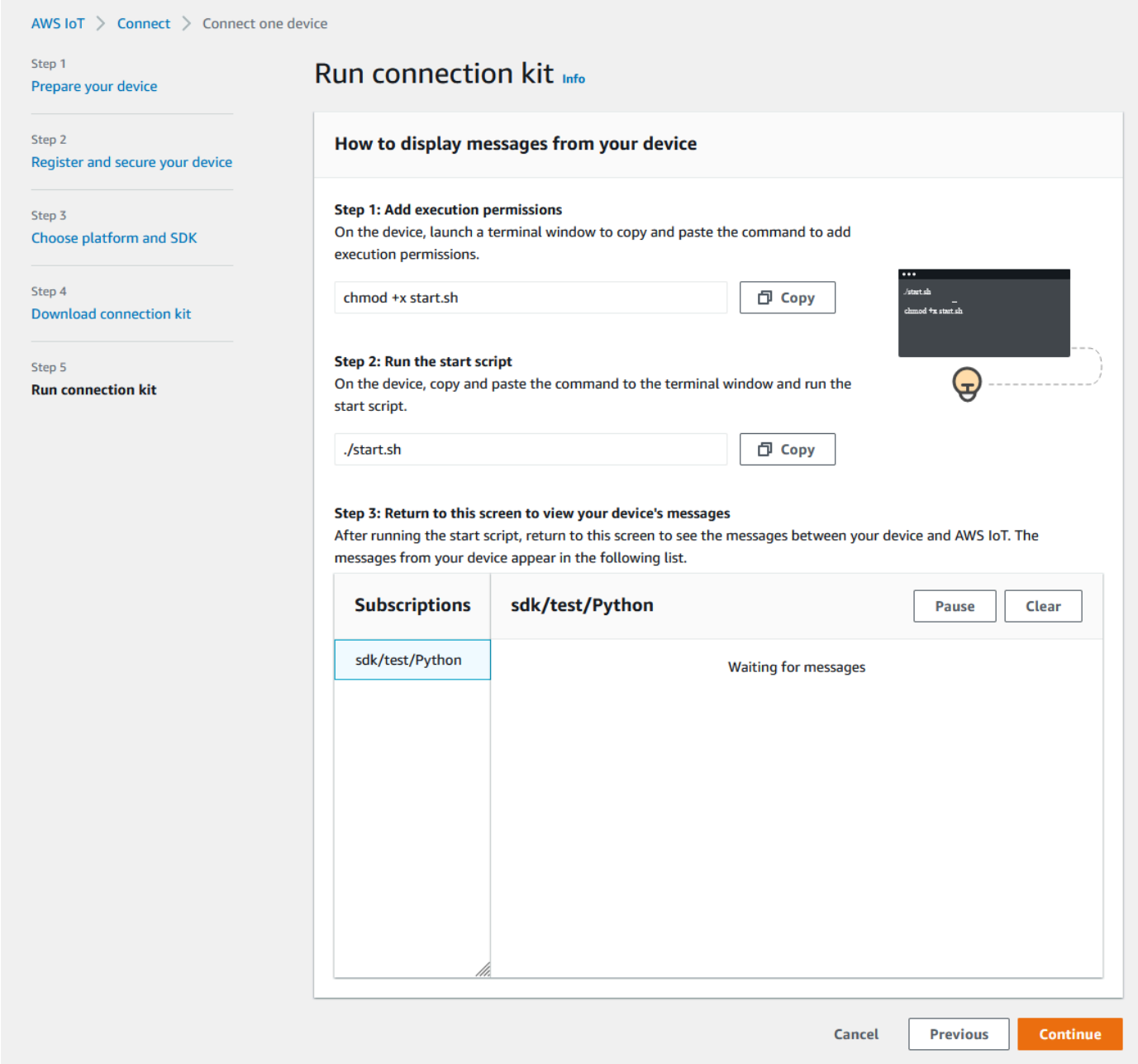
4. Dopo aver installato il file del kit di connessione sul dispositivo, continuare il tutorial scegliendo Next (Avanti).



Passaggio 4. Esegui l'esempio

Dovrai eseguire questa procedura in un terminale o in una finestra di comando del dispositivo seguendo le indicazioni visualizzate nella console. I comandi visualizzati nella console sono quelli del sistema operativo scelto in the section called “Passaggio 2. Crea un oggetto”. Quelli mostrati qui si riferiscono ai sistemi Linux/OSX operativi.

1. In una finestra di terminale o di comando del dispositivo, nella directory contenente il file del kit di connessione, esegui i passaggi mostrati nella AWS IoT console.



2. Dopo avere inserito il comando di Step 2 (Fase 2) nella console, nella finestra di comando o nel terminale del dispositivo si dovrebbe vedere un output simile al seguente. Questo output proviene dai messaggi che il programma sta inviando e quindi ricevendo di nuovo da AWS IoT Core.

```
Running pub/sub sample application...
Connecting to a13hikvzkye6lx-ats.iot.us-east-1.amazonaws.com with client ID 'basicPubSub'...
Connected!
Subscribing to topic 'sdk/test/Python'...
Subscribed with QoS.AT_LEAST_ONCE
Sending messages until program killed
Publishing message to topic 'sdk/test/Python': Hello World! [1]
Received message from topic 'sdk/test/Python': b'"Hello World! [1]"'
Publishing message to topic 'sdk/test/Python': Hello World! [2]
Received message from topic 'sdk/test/Python': b'"Hello World! [2]"'
Publishing message to topic 'sdk/test/Python': Hello World! [3]
Received message from topic 'sdk/test/Python': b'"Hello World! [3]"'
```

   Mentre il programma di esempio è in esecuzione, apparirà anche il messaggio di prova `Hello World!`. Il messaggio di prova viene visualizzato nel terminale o nella finestra di comando del dispositivo.

   > **Note**
   >
   > Per ulteriori informazioni sulla sottoscrizione e la pubblicazione degli argomenti, consulta il codice di esempio dell'SDK scelto.

3. Per eseguire nuovamente il programma di esempio, ripetere i comandi di Step 2 (Fase 2) di questa procedura nella console.
4. (Facoltativo) Se desideri visualizzare i messaggi del tuo client IoT nella AWS IoT console, apri il client di test MQTT nella pagina Test della AWS IoT console. In MQTT test client (Client MQTT di test), in Topic filter (Filtro argomenti), inserisci **`sdk/test/python`** per sottoscrivere i messaggi dal tuo dispositivo. I filtri degli argomenti fanno distinzione tra maiuscole e minuscole e dipendono dal linguaggio di programmazione dell'SDK scelto in Step 1 (Fase 1). Per ulteriori informazioni sulla sottoscrizione e la pubblicazione degli argomenti, consulta il codice di esempio dell'SDK scelto.
5. Dopo aver sottoscritto l'argomento del test, esegui ./start.sh sul dispositivo. Per ulteriori informazioni, consulta the section called “Visualizza i messaggi MQTT con il AWS IoT client MQTT”.

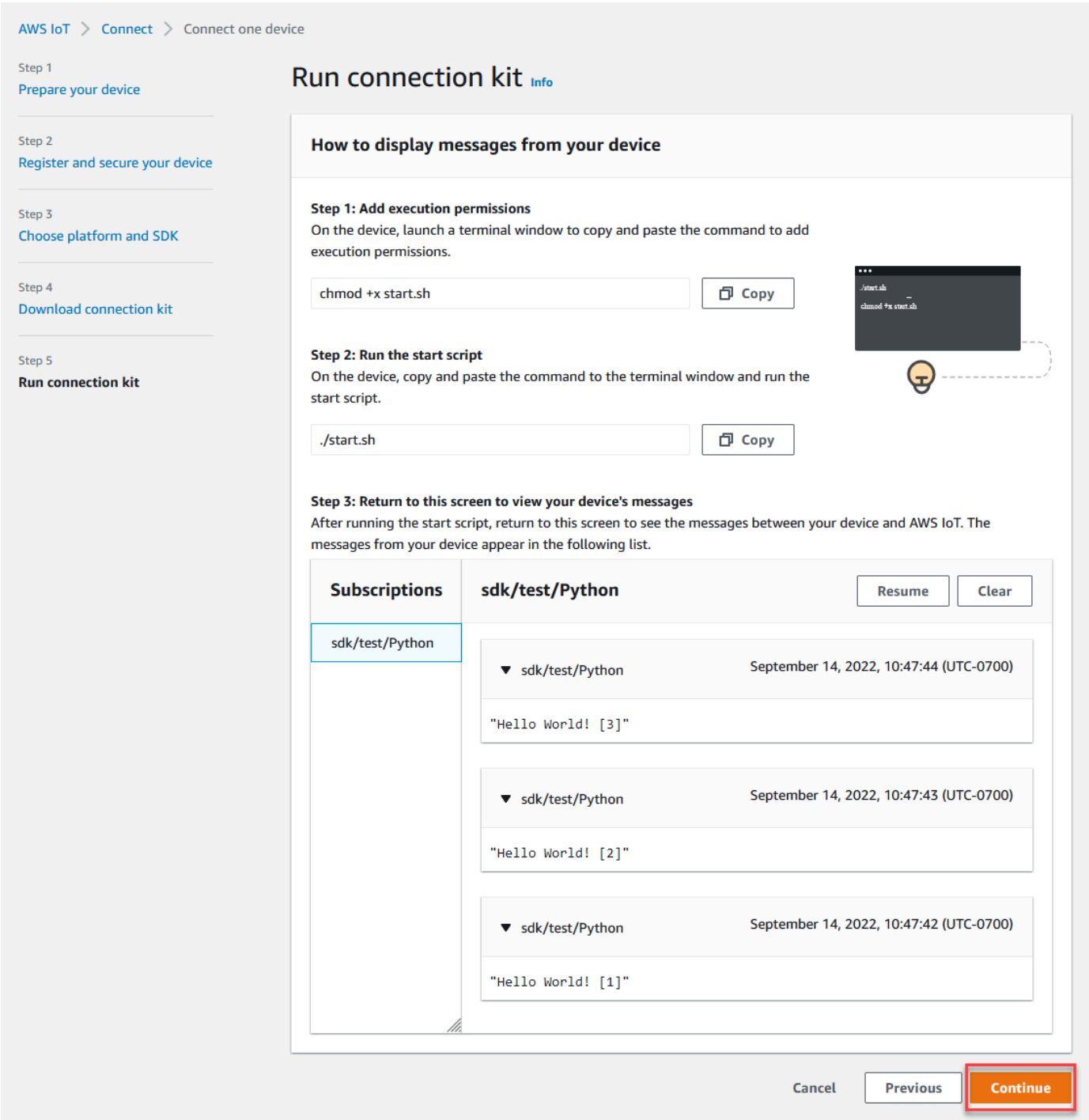
   Dopo aver eseguito ./start.sh, nel client MQTT verranno visualizzati messaggi simili ai seguenti:

```
{
```

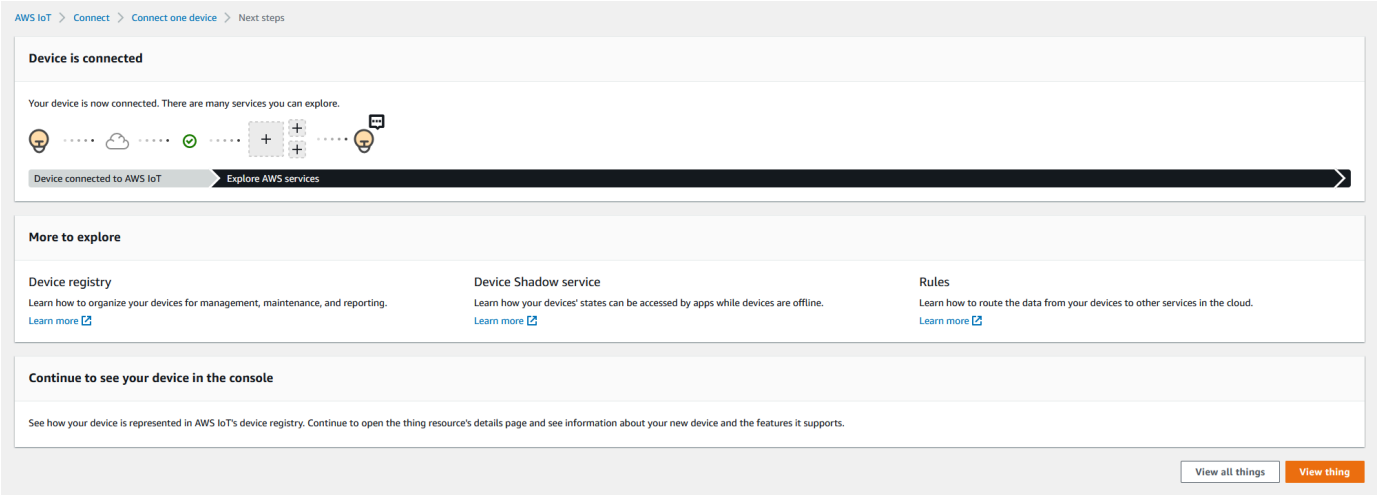
```
  "message": "Hello World!" [1]
}
```

Il numero `sequence` racchiuso in `[]` si incrementa di uno ogni volta che un nuovo messaggio `Hello World!` viene ricevuto e si interrompe quando il programma termina.

6. Per completare il tutorial e visualizzare un riepilogo, nella AWS IoT console, scegliete Continua.



7. Ora verrà visualizzato un riepilogo del tutorial di connessione AWS IoT rapida.



Fase 5. Esplora ulteriormente

Ecco alcune idee da AWS IoT approfondire dopo aver completato la guida rapida.

- Visualizza i messaggi MQTT nel client di test MQTT

  Nella console AWS IoT, apri MQTT client (Client MQTT) sulla pagina Test della console AWS IoT . In MQTT test client (Client di test MQTT), esegui la sottoscrizione a **#** e quindi esegui il programma ./start.sh sul dispositivo, come descritto nella fase precedente. Per ulteriori informazioni, consulta the section called “Visualizza i messaggi MQTT con il AWS IoT client MQTT”.

- Esecuzione di test sui dispositivi con Device Advisor

  Usa Device Advisor per verificare se i tuoi dispositivi possono connettersi e interagire in modo sicuro e affidabile con,. AWS IoT

- the section called “Tutorial interattivo”

  Per avviare il tutorial interattivo, dalla pagina Impara della AWS IoT console, nel riquadro Scopri come AWS IoT funziona, scegli Avvia il tutorial.

- Preparati a esplorare altri tutorial

  Questo avvio rapido ti offre solo un esempio di AWS IoT. Se desideri AWS IoT approfondire e conoscere le funzionalità che la rendono una potente piattaforma di soluzioni IoT, inizia a preparare la tua piattaforma di sviluppo entroAWS IoT Core Scopritelo in tutorial pratici.

Verifica la connettività con l'endpoint di dati del tuo dispositivo

In questo argomento viene descritto come testare la connessione di un dispositivo con l'endpoint dei dati del dispositivo del tuo account, ossia l'endpoint che i tuoi dispositivi IoT utilizzano per connettersi a AWS IoT.

Esegui queste procedure sul dispositivo che desideri testare o utilizza una sessione di terminale SSH collegata al dispositivo che intendi testare.

Test della connettività di un dispositivo con l'endpoint dati del dispositivo.

- Individuazione dell'endpoint dati del dispositivo
- Test rapido della connessione
- Test della connessione all'endpoint e alla porta dati del dispositivo con l'applicazione
- Test della connessione all'endpoint e alla porta dati del dispositivo

Individuazione dell'endpoint dati del dispositivo

Questa procedura spiega come trovare l'endpoint dei dati del dispositivo nella AWS IoT console per testare la connessione al dispositivo IoT.

Individuazione dell'endpoint dati del dispositivo

1. Nella AWS IoT console, nella sezione Connect, vai a Configurazioni di dominio.
2. Nella pagina Configurazioni del dominio, vai al contenitore delle configurazioni del dominio e copia il nome del dominio. Il valore del tuo endpoint è unico per te Account AWS ed è simile a questo esempio:. `a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com`
3. Memorizza l'endpoint dati del dispositivo per poterlo utilizzare nelle procedure seguenti.

Test rapido della connessione

Questa procedura verifica la connettività generale con l'endpoint dati del dispositivo, ma non verifica la porta specifica che verrà utilizzata dai dispositivi. Questo test utilizza un programma comune e di solito è sufficiente per sapere se i dispositivi sono in grado di connettersi a AWS IoT.

Se desideri testare la connettività con la porta specifica utilizzata dai dispositivi, salta questa procedura e passa a Test della connessione all'endpoint e alla porta dati del dispositivo con l'applicazione.

Test rapido dell'endpoint dati del dispositivo

1. In un terminale o nella finestra della riga di comando del dispositivo, sostituisci l'endpoint dati del dispositivo di esempio (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) con l'endpoint dati del dispositivo del tuo account, quindi inserisci questo comando.

   Linux

   ```
   ping -c 5 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

   Windows

   ```
   ping -n 5 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

2. Se `ping` mostra un output simile al seguente, si è connesso correttamente all'endpoint dati del dispositivo. Sebbene non abbia comunicato AWS IoT direttamente con, ha trovato il server ed è probabile che AWS IoT sia disponibile tramite questo endpoint.

   ```
   PING a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (xx.xx.xxx.xxx) 56(84) bytes of
    data.
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=1 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=2 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=3 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=4 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=5 ttl=231 time=127 ms
   ```

   Se il risultato ti soddisfa, puoi interrompere il test qui.

   Se desideri testare la connettività con la porta specifica utilizzata da AWS IoT, vai a Test della connessione all'endpoint e alla porta dati del dispositivo con l'applicazione.

3. Se `ping` non restituisce un output positivo, controlla il valore dell'endpoint per assicurarti che si tratti di quello corretto e verifica la connessione del dispositivo a Internet.

Test della connessione all'endpoint e alla porta dati del dispositivo con l'applicazione

È possibile eseguire un test di connettività più approfondito utilizzando `nmap`. Questa procedura verifica se `nmap` è installato sul dispositivo.

Verificare la presenza di **`nmap`** sul dispositivo

1. In un terminale o nella finestra della riga di comando sul dispositivo che desideri testare, inserisci questo comando per vedere se `nmap` è installato.

   ```
   nmap --version
   ```

2. Se viene visualizzato un output simile al seguente, `nmap` è installato e puoi continuare con the section called “Test della connessione all'endpoint e alla porta dati del dispositivo”.

   ```
   Nmap version 6.40 ( http://nmap.org )
   Platform: x86_64-koji-linux-gnu
   Compiled with: nmap-liblua-5.2.2 openssl-1.0.2k libpcre-8.32 libpcap-1.5.3 nmap-
   libdnet-1.12 ipv6
   Compiled without:
   Available nsock engines: epoll poll select
   ```

3. Se non visualizzi una risposta simile a quella mostrata nel passaggio precedente, è necessario installare `nmap` sul dispositivo. Scegli la procedura in base al sistema operativo del dispositivo.

Linux

Questa procedura richiede l'autorizzazione all'installazione del software sul computer.

Installazione di nmap su un computer Linux

1. In un terminale o nella finestra della riga di comando del dispositivo, inserisci il comando corrispondente alla versione di Linux in esecuzione.

   a. Debian o Ubuntu:

      ```
      sudo apt install nmap
      ```

   b. CentOS o RHEL:

      ```
      sudo yum install nmap
      ```

2. Testa l'installazione con questo comando:

```
nmap --version
```

3. Se viene visualizzato un output simile al seguente, nmap è installato e puoi continuare con the section called “Test della connessione all'endpoint e alla porta dati del dispositivo”.

```
Nmap version 6.40 ( http://nmap.org )
Platform: x86_64-koji-linux-gnu
Compiled with: nmap-liblua-5.2.2 openssl-1.0.2k libpcre-8.32 libpcap-1.5.3 nmap-
libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
```

macOS

Questa procedura richiede l'autorizzazione all'installazione del software sul computer.

Installazione di nmap su un computer macOS

1. In un browser, apri https://nmap.org/download#macosx ed esegui il download dell'ultima versione stabile del programma di installazione.

   Quando richiesto, seleziona Apri con. DiskImageInstaller
2. Nella finestra di installazione, sposta il pacchetto nella cartella Applicazioni.
3. Nel Finder, individua il pacchetto `nmap-xxxx-mpkg` nella cartella Applications (Applicazioni). Esegui il comando Ctrl-click sul pacchetto e seleziona Open (Apri) per aprire il pacchetto.
4. Esamina la finestra di dialogo sulla sicurezza. Se sei pronto per l'installazione di nmap, scegli Apri per installare nmap.
5. Nel Terminal, testa l'installazione con questo comando.

```
nmap --version
```

6. Se viene visualizzato un output simile al seguente, nmap è installato e puoi continuare con the section called “Test della connessione all'endpoint e alla porta dati del dispositivo”.

```
Nmap version 7.92 ( https://nmap.org )
Platform: x86_64-apple-darwin17.7.0
```

```
Compiled with: nmap-liblua-5.3.5 openssl-1.1.1k nmap-libssh2-1.9.0 libz-1.2.11
 nmap-libpcre-7.6 nmap-libpcap-1.9.1 nmap-libdnet-1.12 ipv6 Compiled without:
Available nsock engines: kqueue poll select
```

Windows

Questa procedura richiede l'autorizzazione all'installazione del software sul computer.

Installazione di nmap su un computer Windows

1. In un browser, apri https://nmap.org/download#windows ed esegui il download dell'ultima versione stabile del programma di installazione.

   Se richiesto, scegli Salva file. Dopo aver scaricato il file, aprilo dalla cartella dei download.
2. Al termine del download del file di installazione, apri il file nmap-xxxx-setup.exe che hai scaricato per installare l'applicazione.
3. Accetta le impostazioni predefinite durante l'installazione del programma.

   Per questo test non è necessaria l'applicazione Npcap. Se non desideri installarla, puoi deselezionare tale opzione.
4. Nel Command, testa l'installazione con questo comando.

   ```
   nmap --version
   ```
5. Se viene visualizzato un output simile al seguente, nmap è installato e puoi continuare con the section called “Test della connessione all'endpoint e alla porta dati del dispositivo”.

   ```
   Nmap version 7.92 ( https://nmap.org )
   Platform: i686-pc-windows-windows
   Compiled with: nmap-liblua-5.3.5 openssl-1.1.1k nmap-libssh2-1.9.0 nmap-
   libz-1.2.11 nmap-libpcre-7.6 Npcap-1.50 nmap-libdnet-1.12 ipv6
   Compiled without:
   Available nsock engines: iocp poll select
   ```

Test della connessione all'endpoint e alla porta dati del dispositivo

Questa procedura verifica la connessione del dispositivo IoT all'endpoint dati del dispositivo utilizzando la porta selezionata.

Test dell'endpoint e della porta dati del dispositivo

1. In un terminale o nella finestra della riga di comando del dispositivo, sostituisci l'endpoint dati del dispositivo di esempio (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) con l'endpoint dati del dispositivo del tuo account, quindi inserisci questo comando.

   ```
   nmap -p 8443 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

2. Se nmap mostra un output simile al seguente, nmap si è connesso correttamente all'endpoint dati del dispositivo e alla porta selezionata.

   ```
   Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-18 16:23 Pacific Standard Time
   Nmap scan report for a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
    (xx.xxx.147.160)
   Host is up (0.036s latency).
   Other addresses for a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (not scanned):
    xx.xxx.134.144 xx.xxx.55.139 xx.xxx.110.235 xx.xxx.174.233 xx.xxx.74.65
    xx.xxx.122.179 xx.xxx.127.126
   rDNS record for xx.xxx.147.160: ec2-EXAMPLE-160.eu-west-1.compute.amazonaws.com

   PORT     STATE SERVICE
   8443/tcp open  https-alt
   MAC Address: 00:11:22:33:44:55 (Cimsys)

   Nmap done: 1 IP address (1 host up) scanned in 0.91 seconds
   ```

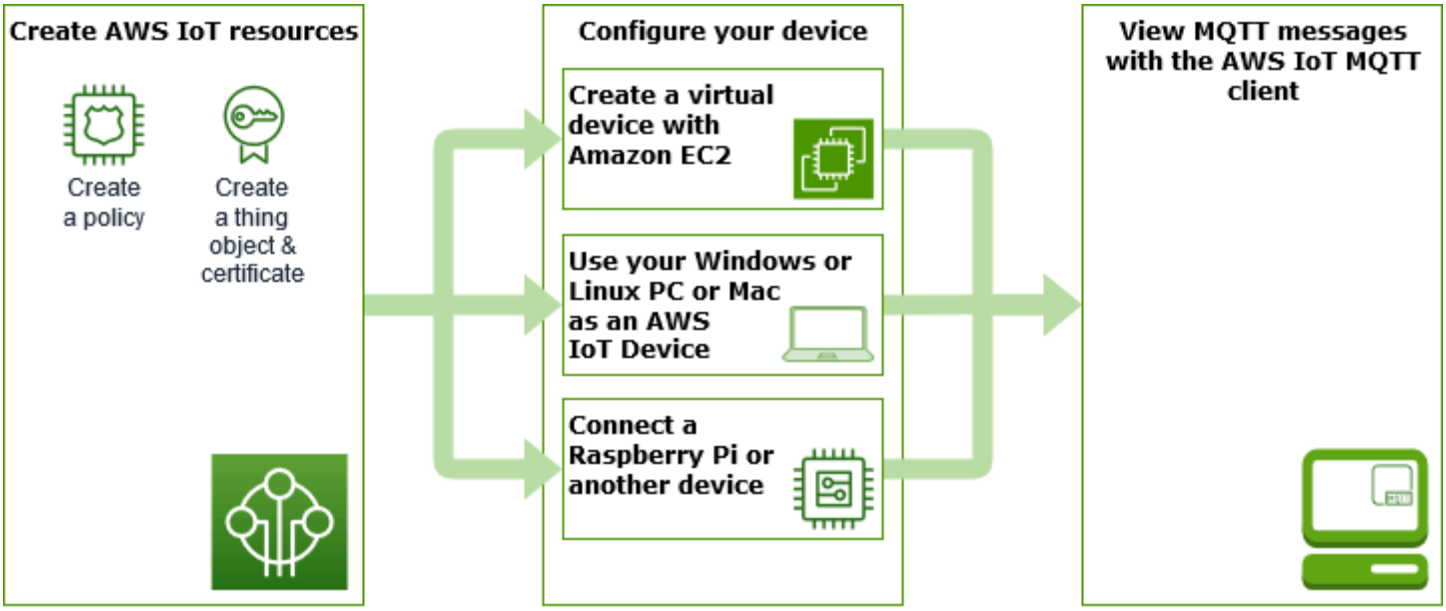
3. Se nmap non restituisce un output positivo, controlla il valore dell'endpoint per assicurarti che sia l'endpoint corretto e verifica la connessione del dispositivo a Internet.

Puoi testare altre porte sull'endpoint dati del dispositivo, ad esempio la porta 443, la porta HTTPS principale, sostituendo la porta utilizzata nel passaggio 1, **8443**, con la porta che desideri testare.

AWS IoT Core Scopritelo in tutorial pratici

In questo tutorial, installerai il software e creerai le AWS IoT risorse necessarie per connettere un dispositivo AWS IoT Core in modo che possa inviare e ricevere messaggi MQTT con AWS IoT Core. Vedrete i messaggi nel client MQTT nella AWS IoT console.

Puoi aspettarti di impiegare 20-30 minuti per questo tutorial. Se utilizzi un dispositivo IoT o un Raspberry Pi, questa esercitazione potrebbe richiedere più tempo se, ad esempio, è necessario installare il sistema operativo e configurare il dispositivo.



Questo tutorial è ideale per gli sviluppatori che vogliono iniziare e continuare a esplorare funzionalità più avanzate, come il motore delle regole e le ombre. AWS IoT Core Questo tutorial ti prepara a continuare a conoscere AWS IoT Core e a capire come interagisce con altri AWS servizi, spiegando i passaggi in modo più dettagliato rispetto al tutorial di avvio rapido. Se cerchi solo un'esperienza Hello World veloce, prova il Prova il tutorial sulla connessione AWS IoT Core rapida.

Dopo aver configurato la tua AWS IoT console Account AWS e la tua console, seguirai questi passaggi per vedere come connettere un dispositivo e come inviargli messaggi. AWS IoT Core

Fasi successive

- Scegli quale opzione di dispositivo è la migliore per te
- Se non intendi creare un dispositivo virtuale con Amazon EC2, scegli the section called “Crea AWS IoT risorse”
- the section called “Configurazione del dispositivo”
- the section called “Visualizza i messaggi MQTT con il AWS IoT client MQTT”

Per ulteriori informazioni su AWS IoT Core, vedi Cos'è AWS IoT Core?

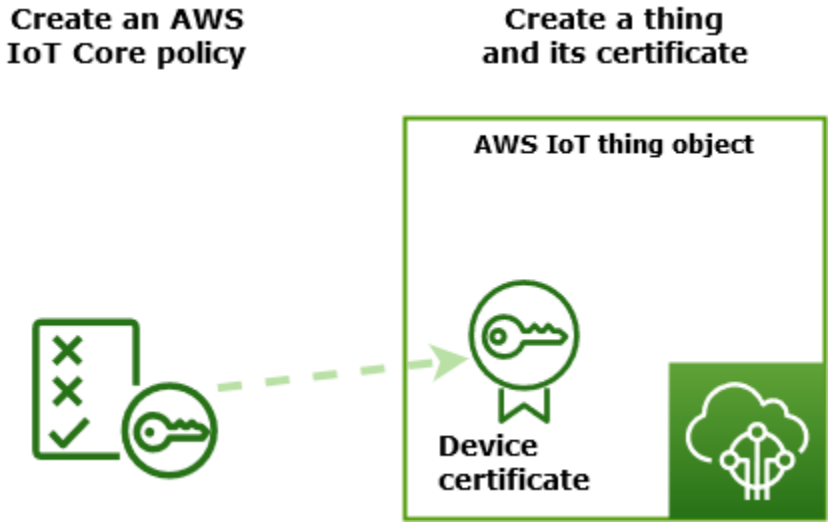
Quale opzione di dispositivo è la migliore per te?

Se non sai quale opzione scegliere, utilizza il seguente elenco dei vantaggi e degli svantaggi di ciascuna opzione per decidere quale è più adatta a te.

Opzione	Questa potrebbe essere una buona opzione se:	Questa potrebbe non essere una buona opzione se:
the section called “Crea un dispositivo virtuale con Amazon EC2”	• Non hai il tuo dispositivo da testare. • Non desideri installare alcun software sul tuo sistema. • Desideri testare su un sistema operativo Linux OS.	• Non hai familiarità con l'uso di un'interfaccia a riga di comando. • Non desideri sostenere ulteriori addebiti AWS . • Non desideri testare su un sistema operativo Linux OS.
the section called “Usa il tuo PC o Mac Windows o Linux come dispositivo AWS IoT”	• Non desideri sostenere ulteriori addebiti AWS . • Non desideri configurare alcun dispositivo aggiuntivo.	• Non desideri installare alcun software nel personal computer. • Desideri una piattaforma di test più rappresentativa.
the section called “Connettere un Raspberry Pi o altro dispositivo”	• Vuoi eseguire il test AWS IoT con un dispositivo reale. • Hai già un dispositivo con cui testare. • Hai esperienza nell'integrazione dell'hardware nei sistemi.	• Non desideri acquistare o configurare un dispositivo solo per provarlo. • Vuoi fare il test AWS IoT nel modo più semplice possibile , per ora.

Crea AWS IoT risorse

In questo tutorial, creerai le AWS IoT risorse necessarie a un dispositivo per connettersi AWS IoT Core e scambiare messaggi.



1. Crea un documento di AWS IoT policy che autorizzi il tuo dispositivo a interagire con AWS IoT i servizi.
2. Crea un oggetto AWS IoT e il relativo certificato del dispositivo X.509, quindi allega il documento relativo alla policy. L'oggetto oggetto è la rappresentazione virtuale del dispositivo nel AWS IoT registro. Il certificato autentica il dispositivo e il documento di policy autorizza AWS IoT Core il dispositivo a interagire con. AWS IoT

> **Note**
>
> Se hai intenzione di the section called “Crea un dispositivo virtuale con Amazon EC2”, puoi saltare questa pagina e continuare con the section called “Configurazione del dispositivo”. Creerai queste risorse una volta creata la tua cosa virtuale.

Questo tutorial utilizza la AWS IoT console per creare le risorse. AWS IoT Se il dispositivo supporta un browser Web, potrebbe essere più semplice eseguire questa procedura sul browser Web del dispositivo perché sarà possibile scaricare i file del certificato direttamente sul dispositivo. Se esegui questa procedura su un altro computer, sarà necessario copiare i file del certificato sul dispositivo prima che possano essere utilizzati dall'app di esempio.

Crea una AWS IoT politica

I dispositivi utilizzano un certificato X.509 con cui effettuare l'autenticazione. AWS IoT Core Al certificato sono associate AWS IoT delle politiche. Queste policy determinano quali operazioni AWS

IoT , ad esempio la sottoscrizione o la pubblicazione in argomenti MQTT, il dispositivo è autorizzato a eseguire. Il dispositivo presenta il certificato quando si connette e invia messaggi a AWS IoT Core.

Segui questi passaggi per creare una policy che consenta al dispositivo di eseguire le operazioni AWS IoT necessarie per eseguire il programma di esempio. È necessario creare la policy AWS IoT prima di poterla collegare al certificato del dispositivo che creerai in seguito.

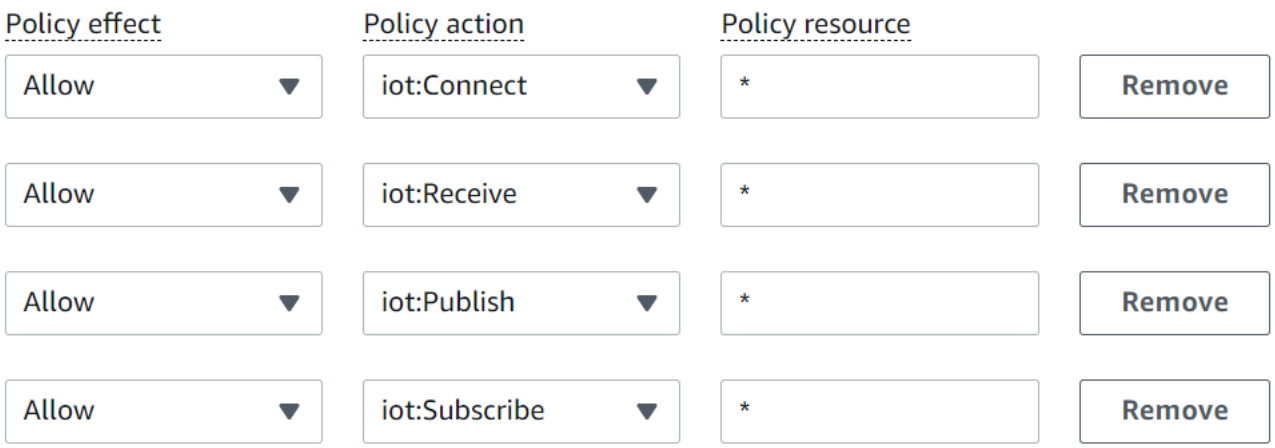
Per creare una AWS IoT politica

1. Nella console AWS IoT, nel menu di sinistra, scegli Sicurezza, quindi scegli Policy.
2. Nella pagina You don't have a policy yet (Al momento non sono disponibili policy), scegli Create a policy (Crea una policy).

   Se il tuo account dispone di policy esistenti, scegli Crea policy.
3. Nella pagina Create a policy (Crea una policy):

   1. Nella sezione Policy properties (Proprietà della policy), nel campo Policy name (Nome policy), inserisci un nome per la policy (ad esempio, **`My_Iot_Policy`**). Non utilizzare dati personali identificabili nei nomi delle policy.
   2. Nella sezione Policy document (Documento di policy), crea le istruzioni della policy che concedono o negano alle risorse l'accesso alle operazioni AWS IoT Core . Per creare un'istruzione della policy che consenta a tutti i client di eseguire **`iot:Connect`**, procedi nel seguente modo:
      - Nel campo Policy effect (Effetto della policy), scegli Allow (Permetti). Questo consente a tutti i client che dispongono di questa policy collegata al certificato di eseguire le operazioni elencate nel campo Policy action (Operazione della policy).
      - Nel campo Policy action (Operazione della policy), scegli un'operazione della policy come **`iot:Connect`**. Le operazioni della policy sono le operazioni per le quali il dispositivo ha bisogno dell'autorizzazione quando esegue il programma di esempio dall'SDK di dispositivo.
      - Nel campo Policy resource (Risorse della policy), inserisci un Amazon Resource Name (ARN) della risorsa o un **`*`**. Un **`*`** per selezionare qualsiasi client (dispositivo).

      Per creare le istruzioni della policy per **`iot:Receive`**, **`iot:Publish`** e **`iot:Subscribe`**,scegli Add new statement (Aggiungi una nuova istruzione) e ripeti i passaggi.



> **Note**
>
> In questo avvio rapido, il carattere jolly (*) viene utilizzato per semplicità. Per una maggiore sicurezza, devi limitare i client (dispositivi) che possono connettersi e pubblicare messaggi specificando un ARN client anziché il carattere jolly come risorsa. Il cliente ARNs segue questo formato:`arn:aws:iot:`*`your-region`*`:`*`your-aws-account`*`:client/`*`my-client-id`*.
>
> Tuttavia, prima di poter assegnare l'ARN a una policy devi creare la risorsa (ad esempio un dispositivo client o una shadow dell'oggetto). Per ulteriori informazioni, consulta la sezione Risorse per l'operazione AWS IoT Core.

4. Dopo avere inserito le informazioni per la policy, scegli Create (Crea).

Per ulteriori informazioni, consulta Come AWS IoT funziona con IAM.

Crea un oggetto

I dispositivi collegati a AWS IoT Core sono rappresentati da oggetti nel AWS IoT registro. Un oggetto rappresenta un dispositivo specifico o un'entità logica. Può trattarsi di un dispositivo fisico o un sensore (ad esempio una lampadina o un interruttore su un muro). Può anche essere un'entità logica, ad esempio un'istanza di un'applicazione o un'entità fisica a cui non si connette AWS IoT, ma è correlata ad altri dispositivi che lo fanno (ad esempio, un'auto dotata di sensori del motore o un pannello di controllo).

Per creare qualcosa nella AWS IoT console

1. Nella console AWS IoT, nel menu di sinistra scegli Tutti i dispositivi e quindi Oggetti.

2. Sulla pagina Things (Oggetti), scegli Create things (Creazione di oggetti).
3. Nella pagina Create things (Crea oggetti), scegli Create a single thing (Crea un singolo oggetto) e poi Next (Successivo).
4. Sulla pagina Specify thing properties (Specifica le proprietà degli oggetti), per Thing name (Nome oggetto) immetti un nome per l'oggetto, ad esempio **`MyIotThing`**.

   Scegli i nomi degli oggetti con attenzione, perché non possono essere modificati in seguito.

   Per cambiare il nome di un oggetto, devi creare un nuovo oggetto, dargli il nuovo nome e quindi eliminare il vecchio oggetto.

   > **Note**
   >
   > Non utilizzare dati personali identificabili nei nomi degli oggetti. Il nome dell'oggetto può essere visualizzato nelle comunicazioni e nei report non crittografati.

5. Mantieni vuoti gli altri campi di questa pagina. Scegli Next (Successivo).
6. Sulla pagina Configure device certificate - optional (Configura il certificato del dispositivo - facoltativo), scegli Auto-generate a new certificate (recommended) (Generazione automatica di un nuovo certificato) (scelta consigliata). Scegli Next (Successivo).
7. Sulla pagina Attach policies to certificate - optional (Collega policy al certificato - facoltativo) seleziona la policy creata nella sezione precedente. In quella sezione, la policy è stata denominata **`My_Iot_Policy`**. Scegli Create thing (Crea oggetto).
8. Sulla pagina Download certificates and keys (Scarica certificati e chiavi):

   1. Scarica tutti i file di certificato e chiave e salvali per un secondo momento. Dovrai installare questi file sul tuo dispositivo.

      Quando salvi i file di certificato, assegna loro i nomi della tabella seguente. Questi sono i nomi di file utilizzati negli esempi successivi.

      Nomi dei file dei certificati

File	Percorso del file
Chiave privata	`private.pem.key`
Chiavi pubbliche	(non utilizzato in questi esempi)

File	Percorso del file
Certificato del dispositivo	`device.pem.crt`
Un certificato emesso da una CA root	`Amazon-root-CA-1.pem`

2. Per scaricare il file della CA root per questi file, scegli il collegamento Download (Scarica) del file del certificato CA root corrispondente al tipo di endpoint di dati e suite di crittografia in uso. In questo tutorial, scegli Download (Scarica) a destra di RSA 2048 bit key: Amazon Root CA 1 (Chiave RSA a 2048 bit: autorità di certificazione root Amazon 1) e scarica il file Chiave RSA a 2048 bit: autorità di certificazione root Amazon 1.

   > **Important**
   >
   > È necessario salvare i file del certificato prima di lasciare questa pagina. Dopo aver lasciato questa pagina nella console, non avrai più accesso ai file di certificato.
   > Se ti sei è dimenticato di scaricare i file di certificato creati in questa fase, esci dalla schermata della console, accedi all'elenco di oggetti nella console, elimina l'oggetto creato e quindi riavvia la procedura dall'inizio.

3. Seleziona Fatto.

Dopo aver completato questa procedura, dovresti vedere il nuovo oggetto nell'elenco degli oggetti.

Configurazione del dispositivo

Questa sezione descrive come configurare il dispositivo per la connessione ad AWS IoT Core. Se vuoi iniziare AWS IoT Core ma non hai ancora un dispositivo, puoi creare un dispositivo virtuale usando Amazon EC2 oppure puoi usare il tuo PC Windows o Mac come dispositivo IoT.

Seleziona l'opzione di dispositivo migliore da provare AWS IoT Core. Certo, puoi provarli tutti, ma provane solo uno alla volta. Se non sei certo di quale opzione del dispositivo è più adatta a te, leggi come scegliere quale opzione del dispositivo è la migliore e poi torna a questa pagina.

Opzioni del dispositivo

- Crea un dispositivo virtuale con Amazon EC2
- Usa il tuo PC o Mac Windows o Linux come dispositivo AWS IoT
- Connettere un Raspberry Pi o altro dispositivo

Crea un dispositivo virtuale con Amazon EC2

In questo tutorial, creerai un'istanza Amazon EC2 da utilizzare come dispositivo virtuale nel cloud.

Per completare questo tutorial, hai bisogno di un Account AWS. Se non disponi dell'account, effettua la procedura descritta in Configurare Account AWS, prima di continuare.

In questo tutorial, apprenderai a:

- Configurare un'istanza Amazon EC2
- Installa Git, Node.js e configura AWS CLI
- Crea AWS IoT risorse per il tuo dispositivo virtuale
- Installa l'SDK AWS IoT del dispositivo per JavaScript
- Esecuzione dell'applicazione di esempio
- Visualizza i messaggi dell'app di esempio nella AWS IoT console

Configurare un'istanza Amazon EC2

Nei passaggi seguenti viene illustrato come creare un'istanza Amazon EC2 che fungerà da dispositivo virtuale al posto di un dispositivo fisico.

Se è la prima volta che crei un'istanza Amazon EC2, potresti trovare più utili le istruzioni contenute nella Guida introduttiva alle istanze Amazon EC2 Linux.

Per avviare un'istanza

1. Apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Dal menu della console a sinistra, espandi la sezione Instances (Istanze) e scegli Istanze (Istanze). Dalla dashboard Istanze (Istanze), scegli Launch instances (Avvia istanze) sulla destra per visualizzare un elenco di configurazioni di base.
3. Nella sezione Name and tags (Nome e tag), inserisci un nome per l'istanza e, facoltativamente, aggiungi tag.
4. Nella sezione Immagini di applicazioni e sistema operativo (Amazon Machine Image), scegli un modello AMI per l'istanza, ad esempio Amazon Linux 2 AMI (HVM). Nota che queste AMI sono contrassegnate dalla dicitura "Free tier eligible" (Idonea per il piano gratuito).
5. Nella sezione Instance type (Tipo di istanza), è possibile selezionare la configurazione hardware per l'istanza. Selezionare il tipo `t2.micro`, ovvero l'opzione selezionata per impostazione di default. Questo tipo di istanza è idoneo per il piano gratuito.

6. Nella sezione Key pair (login) (Coppia di chiavi (accesso)), scegli il nome di una coppia di chiavi dall'elenco a discesa o scegli Create a new key pair (Crea una nuova coppia di chiavi) per crearne una nuova. Quando crei una nuova coppia di chiavi, assicurati di scaricare il file della chiave privata e di salvarlo in un luogo sicuro, perché questa è l'unica possibilità che hai per scaricarlo e salvarlo. Dovrai fornire il nome della coppia di chiavi quando avvii un'istanza e la chiave privata corrispondente ogni volta che ti connetti all'istanza.

   > **Warning**
   >
   > Non selezionare l'opzione Proceed without a key pair (Procedi senza una coppia di chiavi). Se l'istanza viene avviata senza una coppia di chiavi, non sarà possibile connettersi a essa.

7. Nella sezione Network settings (Impostazioni di rete) e nella sezione Configure storage (Configura archiviazione), è possibile mantenere le impostazioni predefinite. Al termine, scegli Launch Instances (Avvia istanze).
8. Una pagina di conferma indicherà che l'istanza si sta avviando. Scegliere View Instances (Visualizza istanze) per chiudere la pagina di conferma e tornare alla console.
9. Nella schermata Instances (Istanze), è possibile visualizzare lo stato dell'avvio. L'avvio di un'istanza richiede pochi minuti. Quando avvii un'istanza, il suo stato iniziale è `pending`. Dopo aver avviato l'istanza, il relativo stato cambia in `running` e l'istanza riceve un nome DNS pubblico. (Se la colonna Public DNS (IPv4) è nascosta, scegli Mostra/Nascondi colonne (l'icona a forma di ingranaggio) nell'angolo in alto a destra della pagina, quindi seleziona Public DNS ().) IPv4
10. Possono essere necessari alcuni minuti affinché l'istanza sia pronta e sia possibile connettervisi. Controllare che l'istanza abbia superato i controlli relativi allo stato. È possibile visualizzare queste informazioni nella colonna Status Checks (Verifiche dello stato).

    Dopo che la nuova istanza ha superato i controlli di stato, continua con la procedura successiva e connettiti ad essa.

Per collegarsi all'istanza

È possibile connettersi a un'istanza tramiteil client basato su browser selezionando l'istanza dalla console Amazon EC2 e scegliendo di effettuare la connessione tramite Amazon EC2 Instance Connect. Instance Connect gestisce le autorizzazioni e fornisce una connessione valida.

1. Apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nel menu a sinistra, seleziona Instances (Istanze).
3. Selezionare l'istanza, quindi scegliere Collegarsi.
4. Scegli Amazon EC2 Instance Connect, Connect (Collega).

Ora dovresti disporre di una finestra Amazon EC2 Instance Connect che effettua l'accesso alla nuova istanza Amazon EC2.

Installa Git, Node.js e configura AWS CLI

In questa sezione, installerai Git e Node.js, nell'istanza Linux.

Per installare Git

1. Nella finestra Amazon EC2 Instance Connect, esegui l'aggiornamento dell'istanza utilizzando il comando seguente.

   ```
   sudo yum update -y
   ```

2. Nella finestra Amazon EC2 Instance Connect, installa Git utilizzando il comando seguente.

   ```
   sudo yum install git -y
   ```

3. Per verificare se Git è stato installato e la versione corrente di Git, esegui il seguente comando:

   ```
   git --version
   ```

Per installare Node.js

1. Nella finestra Amazon EC2 Instance Connect, installa il gestore delle versioni del nodo (nvm) utilizzando il seguente comando.

   ```
   curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
   ```

   Verrà utilizzato nvm per installare Node.js perché è in grado di installare più versioni di Node.js e consente di passare dall'una all'altra.

2. Nella finestra Amazon EC2 Instance Connect, attiva nvm utilizzando questo comando.

```
. ~/.nvm/nvm.sh
```

3. Nella finestra Amazon EC2 Instance Connect, utilizza nvm per installare la versione più recente di Node.js utilizzando questo comando.

```
nvm install 16
```

> **Note**
>
> Installa l'ultima versione LTS di Node.js.

Installando Node.js, viene installato anche il gestore di pacchetti nodo (npm), quindi è possibile installare moduli aggiuntivi in base alle esigenze.

4. Nella finestra Amazon EC2 Instance Connect verifica che Node.js sia installato e funzioni correttamente utilizzando questo comando.

```
node -e "console.log('Running Node.js ' + process.version)"
```

Questo tutorial richiede Node 10.0 o versioni successive. Per ulteriori informazioni, consulta Tutorial: Setting Up Node.js on an Amazon EC2 Instance (Tutorial: configurazione di Node.js su un'istanza Amazon EC2).

Per configurare AWS CLI

L'istanza Amazon EC2 viene precaricata con AWS CLI. Tuttavia, devi completare il tuo AWS CLI profilo. Per ulteriori informazioni su come configurare la tua CLI, consulta Configurazione di AWS CLI.

1. L'esempio seguente illustra i valori di esempio. Sostituisci questi valori con i tuoi valori. Questi valori sono disponibili nella console AWS nelle informazioni dell'account in Security credentials (Credenziali di sicurezza).

Nella finestra Amazon EC2 Instance Connect, immetti il comando:

```
aws configure
```

Quindi inserisci i valori dal tuo account alle richieste visualizzate.

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

2. Puoi testare la tua AWS CLI configurazione con questo comando:

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

   Se il tuo AWS CLI è configurato correttamente, il comando dovrebbe restituire un indirizzo di endpoint inviato dal tuo Account AWS.

Crea AWS IoT risorse per il tuo dispositivo virtuale

Questa sezione descrive come utilizzare AWS CLI per creare l'oggetto e i relativi file di certificato direttamente sul dispositivo virtuale. Questo viene fatto direttamente sul dispositivo per evitare le potenziali complicazioni che potrebbero derivare dalla copia sul dispositivo da un altro computer. In questa sezione, creerai le seguenti risorse per il tuo dispositivo virtuale:

- Un oggetto in cui rappresentare il dispositivo virtuale AWS IoT.
- Un certificato per autenticare il dispositivo virtuale.
- Un documento di policy per autorizzare il dispositivo virtuale a connettersi ad AWS IoT e pubblicare, ricevere e sottoscrivere messaggi.

Per creare un AWS IoT oggetto nell'istanza Linux

I dispositivi collegati a AWS IoT sono rappresentati da oggetti Thing nel AWS IoT registro. Un oggetto rappresenta un dispositivo specifico o un'entità logica. In questo caso, il tuo oggetto rappresenta il dispositivo virtuale, questa istanza Amazon EC2.

1. Nella finestra Amazon EC2 Instance Connect, esegui il comando seguente per creare il tuo oggetto.

   ```
   aws iot create-thing --thing-name "MyIotThing"
   ```

2. La risposta JSON deve essere simile alla seguente:

   ```
   {
   ```

```
    "thingArn": "arn:aws:iot:your-region:your-aws-account:thing/MyIotThing",
    "thingName": "MyIotThing",
    "thingId": "6cf922a8-d8ea-4136-f3401EXAMPLE"
}
```

Per creare e allegare AWS IoT chiavi e certificati nella tua istanza Linux

Il comando create-keys-and-certificate crea certificati client firmati dall'autorità di certificazione root Amazon. Questo certificato viene utilizzato per autenticare l'identità del dispositivo virtuale.

1. Nell finestra Amazon EC2 Instance Connect crea una directory in cui archiviare i file del certificato e delle chiavi.

   ```
   mkdir ~/certs
   ```

2. Nella finestra Amazon EC2 Instance Connect, scarica una copia del certificato dell'autorità di certificazione (CA) Amazon utilizzando questo comando.

   ```
   curl -o ~/certs/Amazon-root-CA-1.pem \
       https://www.amazontrust.com/repository/AmazonRootCA1.pem
   ```

3. Nella finestra Amazon EC2 Instance Connect, esegui il comando seguente per creare la tua chiave privata e pubblica e i file del certificato X.509. Questo comando inoltre registra e attiva il certificato con. AWS IoT

   ```
   aws iot create-keys-and-certificate \
       --set-as-active \
       --certificate-pem-outfile "~/certs/device.pem.crt" \
       --public-key-outfile "~/certs/public.pem.key" \
       --private-key-outfile "~/certs/private.pem.key"
   ```

   La risposta avrà il seguente aspetto. Salva il `certificateArn` in modo da poterlo utilizzare nei comandi successivi. Ne avrai bisogno per collegare il certificato al tuo oggetto e per allegare la policy al certificato in un passaggio successivo.

   ```
   {
       "certificateArn": "arn:aws:iot:us-
   west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
       "certificateId":
    "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
   ```

```
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC
 KEY-----\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h
\nMMEXAMPLEuuN/dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/
gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQWE
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB\n-----
END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omitted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

4. Nella finestra di Amazon EC2 Instance Connect, collega l'oggetto oggetto al certificato appena creato utilizzando il comando seguente e la *certificateArn* risposta del comando precedente.

```
aws iot attach-thing-principal \
    --thing-name "MyIotThing" \
    --principal "certificateArn"
```

Se va a buon fine, questo comando non restituisce alcun output.

Creazione e collegamento di una policy

1. Nella finestra Amazon EC2 Instance Connect, crea il file della policy copiando e incollando questo documento di policy in un file denominato **`~/policy.json`**.

   Se non disponi di un editor Linux preferito, puoi aprire nano, utilizzando questo comando.

   ```
   nano ~/policy.json
   ```

   Incolla il documento di policy per `policy.json` in esso. Salva il file ed esci dall'editor di testo nano (Ctrl-X).

   Contenuto del documento di policy per `policy.json`.

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "iot:Publish",
                   "iot:Subscribe",
                   "iot:Receive",
                   "iot:Connect"
               ],
               "Resource": [
                   "*"
               ]
           }
       ]
   }
   ```

2. Nella finestra Amazon EC2 Instance Connect crea la policy utilizzando il comando seguente.

   ```
   aws iot create-policy \
       --policy-name "MyIotThingPolicy" \
       --policy-document "file://~/policy.json"
   ```

   Output:

```
{
    "policyName": "MyIotThingPolicy",
    "policyArn": "arn:aws:iot:us-east-1:your-aws-account:policy/
MyIotThingPolicy",
    "policyDocument": "{
        \"Version\": \"2012-10-17\",
        \"Statement\": [
            {
                \"Effect\": \"Allow\",
                \"Action\": [
                    \"iot:Publish\",
                    \"iot:Receive\",
                    \"iot:Subscribe\",
                    \"iot:Connect\"
                ],
                \"Resource\": [
                    \"*\"
                ]
        }
        ]
    }",
    "policyVersionId": "1"
}
```

3. Nella finestra Amazon EC2 Instance Connect, collega la policy al certificato del dispositivo virtuale utilizzando il comando seguente.

```
aws iot attach-policy \
    --policy-name "MyIotThingPolicy" \
    --target "certificateArn"
```

Se va a buon fine, questo comando non restituisce alcun output.

Installa l'SDK AWS IoT del dispositivo per JavaScript

In questa sezione installerai AWS IoT Device SDK for JavaScript, che contiene il codice con cui le applicazioni possono comunicare AWS IoT e i programmi di esempio. Per ulteriori informazioni, consulta AWS IoT Device SDK for JavaScript GitHub repository.

Per installare AWS IoT Device SDK for JavaScript sulla tua istanza Linux

1. Nella finestra Amazon EC2 Instance Connect, clona il AWS IoT Device SDK for JavaScript repository nella `aws-iot-device-sdk-js-v2` directory della tua home directory usando questo comando.

```
cd ~
git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
```

2. Naviga nella directory `aws-iot-device-sdk-js-v2` creata nella fase precedente.

```
cd aws-iot-device-sdk-js-v2
```

3. Utilizzare npm per installare l'SDK.

```
npm install
```

Esecuzione dell'applicazione di esempio

I comandi delle sezioni successive presuppongono che i file di chiave e certificato siano memorizzati sul dispositivo virtuale, come illustrato in questa tabella.

Nomi dei file dei certificati

File	Percorso del file
Chiave privata	`~/certs/private.pem.key`
Certificato del dispositivo	`~/certs/device.pem.crt`
Un certificato emesso da una CA root	`~/certs/Amazon-root-CA-1.pem`

In questa sezione, installerai ed eseguirai l'app di `pub-sub.js` esempio che si trova nella `aws-iot-device-sdk-js-v2/samples/node` directory del Device SDK per. AWS IoT JavaScript Questa app mostra come un dispositivo, la tua istanza Amazon EC2, utilizza la libreria MQTT per pubblicare ed effettuare la sottoscrizione ai messaggi MQTT. L'app di esempio `pub-sub.js` esegue la sottoscrizione a un argomento `topic_1`, pubblica 10 messaggi in tale argomento e visualizza i messaggi ricevuti dal broker di messaggi.

Per installare ed eseguire l'app di esempio

1. Nella finestra Amazon EC2 Instance Connect, accedi alla directory `aws-iot-device-sdk-js-v2/samples/node/pub_sub` che l'SDK ha creato e installa l'app di esempio utilizzando questi comandi.

   ```
   cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
   npm install
   ```

2. Nella finestra di Amazon EC2 Instance Connect, accedi *`your-iot-endpoint`* AWS IoT utilizzando questo comando.

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

3. Nella finestra Amazon EC2 Instance Connect, inserisci *`your-iot-endpoint`* come indicato ed esegui questo comando.

   ```
   node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
   certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
   ```

L'applicazione di esempio:

1. Si AWS IoT Core connette al tuo account.
2. Sottoscrive l'argomento del messaggio topic_1 e visualizza i messaggi ricevuti su tale argomento.
3. Pubblica 10 messaggi sull’argomento topic_1.
4. L'output sia simile a quello riportato di seguito:

```
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":1}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":2}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":3}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":4}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":5}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":6}
```

```
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":7}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":8}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":9}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":10}
```

In caso di problemi nell'esecuzione dell'app di esempio, consulta the section called “Risolvi i problemi con l'applicazione di esempio”.

È inoltre possibile aggiungere il parametro `--verbosity debug` alla riga di comando in modo che l'app di esempio visualizzi messaggi dettagliati su ciò che sta facendo. Tali informazioni potrebbero fornirti l'aiuto necessario per risolvere il problema.

Visualizza i messaggi dell'app di esempio nella AWS IoT console

È possibile visualizzare i messaggi dell'app di esempio durante il passaggio attraverso il broker di messaggi utilizzando il client di test MQTT nella console AWS IoT .

Per visualizzare i messaggi MQTT pubblicati dall'app di esempio

1. Verificare Visualizza i messaggi MQTT con il AWS IoT client MQTT. In questo modo si impara come utilizzare il client di test MQTT nella console AWS IoT per visualizzare i messaggi MQTT durante il passaggio attraverso il broker di messaggi.
2. Apri il client di test MQTT nella console AWS IoT .
3. In Subscribe to a topic (Sottoscrizione a un argomento), effettua la sottoscrizione all'argomento, topic_1.
4. Nella finestra Amazon EC2 Instance Connect, esegui nuovamente l'app di esempio e guarda i messaggi nel client d test MQTT nella console AWS IoT .

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
```

Per ulteriori informazioni su MQTT e su come AWS IoT Core supporta il protocollo, vedere MQTT.

Usa il tuo PC o Mac Windows o Linux come dispositivo AWS IoT

In questo tutorial, configurerai un personal computer da utilizzare con AWS IoT. Queste istruzioni supportano Windows, Linux PCs e Mac. Per eseguire questa operazione, è necessario installare alcuni software sul computer. Se non desideri installare software sul computer, puoi provare Crea un dispositivo virtuale con Amazon EC2 che installa tutto il software in una macchina virtuale.

In questo tutorial, apprenderai a:

- Configurare il tuo personal computer
- Installa Git, Python e AWS IoT Device SDK per Python
- Configurare la policy ed eseguire l'applicazione di esempio
- Visualizza i messaggi dell'app di esempio nella AWS IoT console
- Esecuzione dell'esempio Sottoscrizione condivisa in Python

Configurare il tuo personal computer

Per completare questo tutorial, è necessario un PC Windows o Linux o un Mac con connessione a Internet.

Prima di continuare con la fase successiva, assicurati di poter aprire una finestra della riga di comando nel computer. Utilizza cmd.exe su un PC Windows. Su un PC Linux o un Mac, usa Terminal.

Installa Git, Python e AWS IoT Device SDK per Python

In questa sezione installerai Python e AWS IoT Device SDK for Python sul tuo computer.

Installa e usa la versione più recente di Git e di Python

Questa procedura spiega come installare l'ultima versione di Git e Python sul tuo personal computer.

Per scaricare e installare Git e Python sul tuo computer

1. Verificare che sia installato Git nel computer. Nella riga di comando, immetti questo comando.

   ```
   git --version
   ```

   Se il comando visualizza la versione Git, Git è installato e puoi continuare con la fase successiva.

Se il comando visualizza un errore, apri https://git-scm.com/download e installa Git per il tuo computer.

2. Verifica che Python non sia già installato. Inserisci questo comando nella riga di comando.

```
python -V
```

> **Note**
>
> Se questo comando dà un errore: `Python was not found`, potrebbe essere perché il tuo sistema operativo chiama l'eseguibile Python v3.x come `Python3`. In tal caso, sostituisci tutte le istanze di `python` con `python3` e continua il resto di questo tutorial.

Se il comando visualizza la versione Python, Python è già installato. Questo tutorial richiede Python v3.7 o versioni successive.

3. Se Python è installato, puoi ignorare i passaggi rimanenti in questa sezione. Altrimenti, continua.
4. Apri https://www.python.org/downloads/ e scarica il programma di installazione per il tuo computer.
5. Se il download non ha avviato automaticamente l'installazione, esegui il programma scaricato per installare Python.
6. Verificare l'installazione di Python.

```
python -V
```

Conferma che il comando visualizzi la versione di Python. Se la versione di Python non viene visualizzata, prova a scaricare e installare nuovamente Python.

Installa l'SDK AWS IoT del dispositivo per Python

Per installare AWS IoT Device SDK for Python sul tuo computer

1. Installa la v2 del AWS IoT Device SDK for Python.

```
python3 -m pip install awsiotsdk
```

2. Clona il repository AWS IoT Device SDK for Python nella directory aws-iot-device-sdk -python-v2 della tua home directory. Questa procedura si riferisce alla directory di base per i file come installi. *home*

   La posizione effettiva della *home* directory dipende dal sistema operativo in uso.

   Linux/macOS

   In macOS e Linux, la *home* directory è. ~

   ```
   cd ~
   git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
   ```

   Windows

   In Windows, puoi trovare il percorso della *home* directory eseguendo questo comando nella cmd finestra.

   ```
   echo %USERPROFILE%
   cd %USERPROFILE%
   git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
   ```

   > **Note**
   >
   > Se utilizzi Windows PowerShell anziché 1cmd.exe, utilizza il seguente comando.
   >
   > ```
   > echo $home
   > ```

Per ulteriori informazioni, consultate il repository AWS IoT Device SDK for GitHub Python.

Preparazione dell'esecuzione delle applicazioni di esempio

Per preparare il sistema all'esecuzione dell'applicazione di esempio

- Creazione della directory `certs`. Nella directory `certs`, copia i file della chiave privata, del certificato del dispositivo e del certificato CA root salvati quando è stato creato e registrato

l'oggetto in the section called “Crea AWS IoT risorse”. I nomi di file di ogni file nella directory di destinazione devono corrispondere a quelli della tabella.

I comandi della sezione successiva presuppongono che i file di chiave e certificato siano memorizzati sul dispositivo, come illustrato in questa tabella.

Linux/macOS

Esegui questo comando per creare la sottodirectory `certs` che verrà utilizzata quando si eseguono le applicazioni di esempio.

```
mkdir ~/certs
```

Nella nuova sottodirectory, copia i file nei percorsi dei file di destinazione indicati nella tabella riportata di seguito.

Nomi dei file dei certificati

File	Percorso del file
Chiave privata	`~/certs/private.pem.key`
Certificato del dispositivo	`~/certs/device.pem.crt`
Un certificato emesso da una CA root	`~/certs/Amazon-root-CA-1.pem`

Esegui questo comando per elencare i file nella directory `certs` e confrontarli con quelli elencati nella tabella.

```
ls -l ~/certs
```

Windows

Esegui questo comando per creare la sottodirectory `certs` che verrà utilizzata quando si eseguono le applicazioni di esempio.

```
mkdir %USERPROFILE%\certs
```

Nella nuova sottodirectory, copia i file nei percorsi dei file di destinazione indicati nella tabella riportata di seguito.

Nomi dei file dei certificati

File	Percorso del file
Chiave privata	`%USERPROFILE%\certs\private.pem.key`
Certificato del dispositivo	`%USERPROFILE%\certs\device.pem.crt`
Un certificato emesso da una CA root	`%USERPROFILE%\certs\Amazon-root-CA-1.pem`

Esegui questo comando per elencare i file nella directory `certs` e confrontarli con quelli elencati nella tabella.

```
dir %USERPROFILE%\certs
```

Configurare la policy ed eseguire l'applicazione di esempio

In questa sezione imposterai la policy ed eseguirai lo script di esempio `pubsub.py` trovato nella directory `aws-iot-device-sdk-python-v2/samples` di SDK per dispositivi AWS IoT per Python. Questo script mostra come il dispositivo utilizza la libreria MQTT per pubblicare e effettuare la sottoscrizione ai messaggi MQTT.

L'app di esempio `pubsub.py` sottoscrive un argomento `test/topic`, pubblica 10 messaggi in tale argomento e visualizza i messaggi ricevuti dal broker messaggi.

Per eseguire l'applicazione di esempio `pubsub.py` sono necessarie le seguenti informazioni:

Valori dei parametri dell'applicazione

Parametro	Dove trovare il valore
your-iot-endpoint	1. Nella console AWS IoT, nel menu a sinistra, seleziona Settings (Impostazioni). 2. Sulla pagina Settings (Impostazioni) l'endpoint viene visualizzato nella sezione Device data endpoint (Endpoint dei dati del dispositivo).

Il *your-iot-endpoint* valore ha il formato:*endpoint_id*`-ats.iot.`*region*`.amazonaws.com`, ad esempio,. `a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com`

Prima di eseguire lo script, assicurati che le policy relative alle informazioni forniscano le autorizzazioni per lo script di esempio per connettersi, sottoscrivere, pubblicare e ricevere.

Per trovare e rivedere il documento della policy per una risorsa oggetto

1. Nella console AWS IoT, nell'elenco Things (Oggetti), trova la risorsa oggetto che rappresenta il tuo dispositivo.
2. Scegli il link di Name (Nome) della risorsa oggetto che rappresenta il tuo dispositivo per aprire la pagina Thing details (Dettagli dell'oggetto).
3. Nella pagina Thing details (Dettagli dell'oggetto), nella scheda Certificates (Certificati), scegli il certificato allegato alla risorsa oggetto. Dovrebbe esserci un solo certificato nell'elenco. Se ci sono più certificati, scegli il certificato i cui sono installati i file sul tuo dispositivo e che verrà utilizzato per connetterti ad AWS IoT Core.

   Nella pagina dei dettagli del Certificate (Certificato), nella scheda Policies (Policy), scegli la policy allegata al certificato. Dovrebbe essercene solo uno. Se ce n'è più di uno, ripeti il passaggio successivo per assicurarti che almeno una policy conceda l'accesso richiesto.
4. Nella pagina di panoramica Policy, trova l'editor JSON e scegli Edit policy document (Modifica documento della policy) per rivedere e modificare il documento della policy secondo necessità.
5. Nell'esempio seguente viene visualizzata la policy JSON. Nell'`"Resource"`elemento, sostituisci *region:account* con il tuo Regione AWS e Account AWS in ciascuno dei `Resource` valori.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/test/topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topicfilter/test/topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:client/test-*"
            ]
        }
    ]
}
```

Linux/macOS

Per eseguire lo script di esempio su Linux/macOS

1. Nella finestra a riga di comando passa alla directory `~/aws-iot-device-sdk-python-v2/samples/node/pub_sub` che l'SDK ha creato utilizzando questi comandi.

```
cd ~/aws-iot-device-sdk-python-v2/samples
```

2. Nella finestra della riga di comando, sostituisci *your-iot-endpoint* come indicato ed esegui questo comando.

```
python3 pubsub.py --endpoint your-iot-endpoint --ca_file ~/certs/Amazon-root-
CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/private.pem.key
```

Windows

Per eseguire l'applicazione di esempio su un PC Windows

1. Nella finestra a riga di comando passa alla directory `%USERPROFILE%\aws-iot-device-sdk-python-v2\samples` che l'SDK ha creato e installa l'app di esempio utilizzando questi comandi.

```
cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
```

2. Nella finestra della riga di comando, sostituisci *your-iot-endpoint* come indicato ed esegui questo comando.

```
python3 pubsub.py --endpoint your-iot-endpoint --ca_file %USERPROFILE%
\certs\Amazon-root-CA-1.pem --cert %USERPROFILE%\certs\device.pem.crt --key
 %USERPROFILE%\certs\private.pem.key
```

Lo script di esempio:

1. Si connette al AWS IoT Core file per il tuo account.
2. Sottoscrive l'argomento del messaggio test/argomento e visualizza i messaggi ricevuti su tale argomento.
3. Pubblica 10 messaggi sull'argomento test/argomento.
4. L'output sia simile a quello riportato di seguito:

```
Connected!
Subscribing to topic 'test/topic'...
Subscribed with QoS.AT_LEAST_ONCE
```

```
Sending 10 message(s)
Publishing message to topic 'test/topic': Hello World! [1]
Received message from topic 'test/topic': b'"Hello World! [1]"'
Publishing message to topic 'test/topic': Hello World! [2]
Received message from topic 'test/topic': b'"Hello World! [2]"'
Publishing message to topic 'test/topic': Hello World! [3]
Received message from topic 'test/topic': b'"Hello World! [3]"'
Publishing message to topic 'test/topic': Hello World! [4]
Received message from topic 'test/topic': b'"Hello World! [4]"'
Publishing message to topic 'test/topic': Hello World! [5]
Received message from topic 'test/topic': b'"Hello World! [5]"'
Publishing message to topic 'test/topic': Hello World! [6]
Received message from topic 'test/topic': b'"Hello World! [6]"'
Publishing message to topic 'test/topic': Hello World! [7]
Received message from topic 'test/topic': b'"Hello World! [7]"'
Publishing message to topic 'test/topic': Hello World! [8]
Received message from topic 'test/topic': b'"Hello World! [8]"'
Publishing message to topic 'test/topic': Hello World! [9]
Received message from topic 'test/topic': b'"Hello World! [9]"'
Publishing message to topic 'test/topic': Hello World! [10]
Received message from topic 'test/topic': b'"Hello World! [10]"'
10 message(s) received.
Disconnecting...
Disconnected!
```

In caso di problemi nell'esecuzione dell'app di esempio, consulta the section called “Risolvi i problemi con l'applicazione di esempio”.

È inoltre possibile aggiungere il parametro `--verbosity Debug` alla riga di comando in modo che l'app di esempio visualizzi messaggi dettagliati su ciò che sta facendo. Tali informazioni potrebbero aiutarti a correggere il problema.

Visualizza i messaggi dell'app di esempio nella AWS IoT console

È possibile visualizzare i messaggi dell'app di esempio durante il passaggio attraverso il broker di messaggi utilizzando il client di test MQTT nella console AWS IoT .

Per visualizzare i messaggi MQTT pubblicati dall'app di esempio

1. Verificare Visualizza i messaggi MQTT con il AWS IoT client MQTT. In questo modo si impara come utilizzare il client di test MQTT nella console AWS IoT per visualizzare i messaggi MQTT durante il passaggio attraverso il broker di messaggi.

2. Apri il client di test MQTT nella console AWS IoT .
3. In Subscribe to a topic (Sottoscrizione a un argomento), effettua la sottoscrizione all'argomento, test/topic.
4. Nella finestra a riga di comando, esegui di nuovo l'app di esempio e guarda i messaggi nel Client MQTT nella console AWS IoT .

   Linux/macOS

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 pubsub.py --topic test/topic --ca_file ~/certs/Amazon-root-CA-1.pem --
   cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
   endpoint
   ```

   Windows

   ```
   cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
   python3 pubsub.py --topic test/topic --ca_file %USERPROFILE%\certs\Amazon-root-
   CA-1.pem --cert %USERPROFILE%\certs\device.pem.crt --key %USERPROFILE%\certs
   \private.pem.key --endpoint your-iot-endpoint
   ```

Per ulteriori informazioni su MQTT e su come AWS IoT Core supporta il protocollo, vedere MQTT.

Esecuzione dell'esempio Sottoscrizione condivisa in Python

AWS IoT Core supporta sottoscrizioni condivise sia per MQTT 3 che per MQTT 5. Sottoscrizioni condivise consentono a più client di condividere una sottoscrizione a un argomento e solo un client riceverà i messaggi pubblicati su tale argomento utilizzando una distribuzione casuale. Per utilizzare Sottoscrizioni condivise, i client effettuano la sottoscrizione al filtro di argomenti di una sottoscrizione condivisa: `$share/{ShareName}/{TopicFilter}`.

Per configurare la policy ed eseguire l'esempio Sottoscrizione condivisa

1. Per eseguire l'esempio Sottoscrizione condivisa, è necessario impostare la policy dell'oggetto come documentato in MQTT 5 Shared Subscription.
2. Per eseguire l'esempio Sottoscrizione condivisa, esegui i comandi seguenti.

Linux/macOS

Per eseguire lo script di esempio su Linux/macOS

1. Nella finestra a riga di comando passa alla directory `~/aws-iot-device-sdk-python-v2/samples` che l'SDK ha creato utilizzando questi comandi.

```
cd ~/aws-iot-device-sdk-python-v2/samples
```

2. Nella finestra della riga di comando, sostituisci *`your-iot-endpoint`* come indicato ed esegui questo comando.

```
python3 mqtt5_shared_subscription.py --endpoint your-iot-endpoint --ca_file
 ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
private.pem.key --group_identifier consumer
```

Windows

Per eseguire l'applicazione di esempio su un PC Windows

1. Nella finestra a riga di comando passa alla directory `%USERPROFILE%\aws-iot-device-sdk-python-v2\samples` che l'SDK ha creato e installa l'app di esempio utilizzando questi comandi.

```
cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
```

2. Nella finestra della riga di comando, sostituisci *`your-iot-endpoint`* come indicato ed esegui questo comando.

```
python3 mqtt5_shared_subscription.py --endpoint your-iot-endpoint --ca_file
 %USERPROFILE%\certs\Amazon-root-CA-1.pem --cert %USERPROFILE%\certs
\device.pem.crt --key %USERPROFILE%\certs\private.pem.key --group_identifier
 consumer
```

> **Note**
>
> Facoltativamente, puoi specificare un identificatore gruppo in base alle tue esigenze quando esegui l'esempio (ad esempio, `--group_identifier consumer`). Se non viene specificato, `python-sample` è l'identificatore gruppo predefinito.

3. L'aspetto dell'output nella riga di comando può essere simile al seguente:

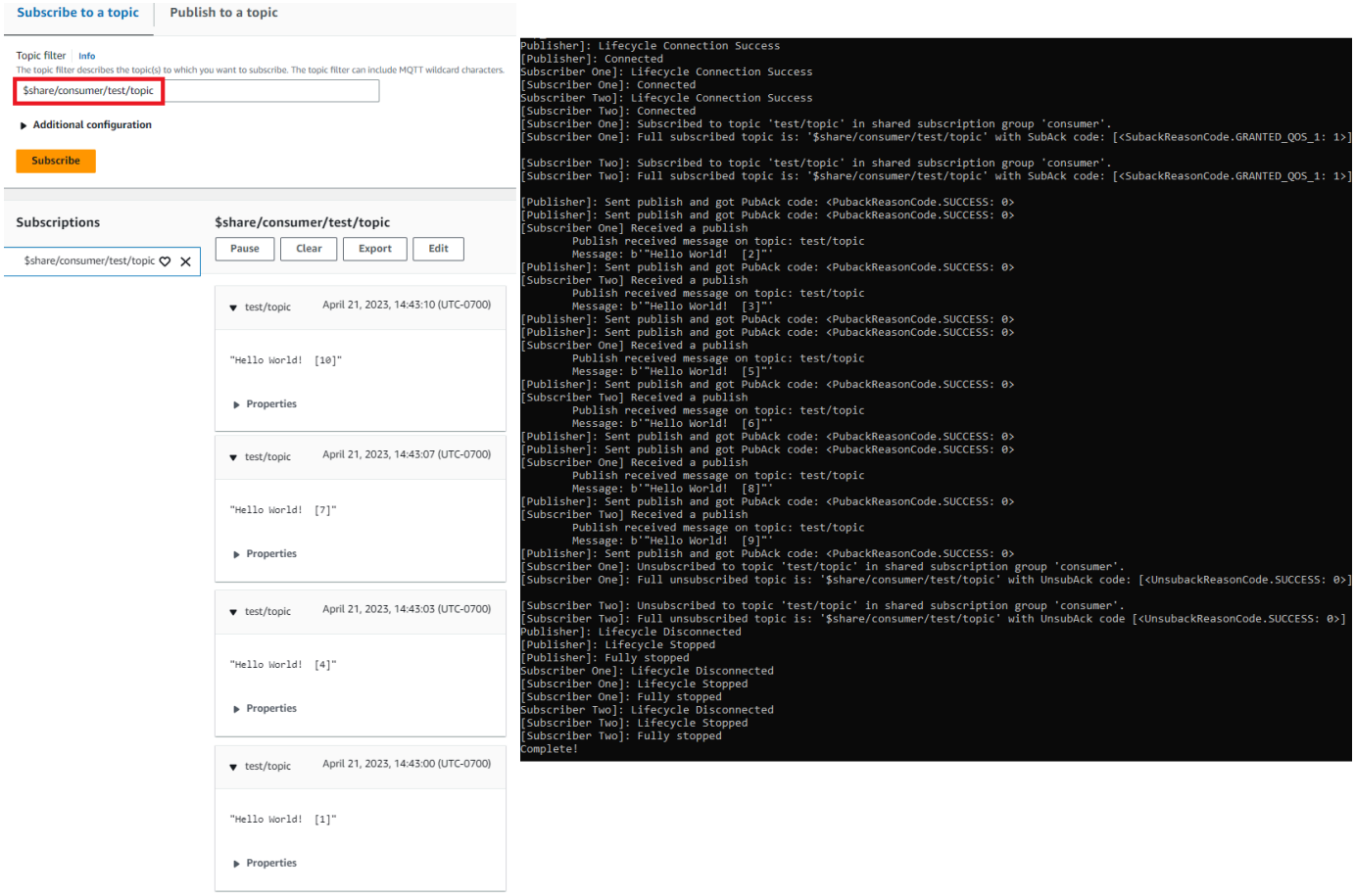
```
Publisher]: Lifecycle Connection Success
[Publisher]: Connected
Subscriber One]: Lifecycle Connection Success
[Subscriber One]: Connected
Subscriber Two]: Lifecycle Connection Success
[Subscriber Two]: Connected
[Subscriber One]: Subscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber One]: Full subscribed topic is: '$share/consumer/test/topic' with
 SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Subscriber Two]: Subscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber Two]: Full subscribed topic is: '$share/consumer/test/topic' with
 SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [1]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [2]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [3]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [4]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
```

```
        Message: b'"Hello World!  [5]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [6]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [7]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [8]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [9]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [10]"'
[Subscriber One]: Unsubscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber One]: Full unsubscribed topic is: '$share/consumer/test/topic' with
 UnsubAck code: [<UnsubackReasonCode.SUCCESS: 0>]
[Subscriber Two]: Unsubscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber Two]: Full unsubscribed topic is: '$share/consumer/test/topic' with
 UnsubAck code [<UnsubackReasonCode.SUCCESS: 0>]
Publisher]: Lifecycle Disconnected
[Publisher]: Lifecycle Stopped
[Publisher]: Fully stopped
Subscriber One]: Lifecycle Disconnected
[Subscriber One]: Lifecycle Stopped
[Subscriber One]: Fully stopped
Subscriber Two]: Lifecycle Disconnected
[Subscriber Two]: Lifecycle Stopped
[Subscriber Two]: Fully stopped
Complete!
```

4. Apri Client di test MQTT nella console AWS IoT . In Sottoscrizione a un argomento, effettua la sottoscrizione all'argomento della Sottoscrizione condivisa, ad esempio: `$share/consumer/test/topic`. Puoi specificare un identificatore gruppo in base alle tue esigenze quando esegui

l'esempio (ad esempio, `--group_identifier consumer`). Se non specifichi un identificatore gruppo, il valore predefinito è `python-sample`. Per ulteriori informazioni, consultare esempio Python MQTT 5 Shared Subscription e Sottoscrizioni condivise nella Guida per gli sviluppatori di AWS IoT Core .

Nella finestra a riga di comando, esegui nuovamente l'app di esempio e guarda la distribuzione dei messaggi nel Client di test MQTT della Console AWS IoT e la riga di comando.



Connettere un Raspberry Pi o altro dispositivo

In questa sezione, configureremo un Raspberry Pi da utilizzare con. AWS IoT Se hai un altro dispositivo che desideri collegare, le istruzioni per Raspberry Pi includono riferimenti che possono aiutarti ad adattare queste istruzioni al tuo dispositivo.

Normalmente questo richiede circa 20 minuti, ma può richiedere più tempo se si hanno molti aggiornamenti del software di sistema da installare.

Nel corso di questo tutorial, apprenderai che:

- Configurare il dispositivo
- Installa gli strumenti e le librerie necessari per AWS IoT Device SDK
- Installa AWS IoT Device SDK
- Installazione ed esecuzione dell'applicazione di esempio
- Visualizza i messaggi dell'app di esempio nella AWS IoT console

> **Important**
>
> Adattare queste istruzioni ad altri dispositivi e sistemi operativi può essere difficile. Dovrai comprendere abbastanza bene il tuo dispositivo da poter interpretare queste istruzioni e applicarle al tuo dispositivo.
> Se riscontri difficoltà durante la configurazione del dispositivo per AWS IoT, potresti provare una delle altre opzioni del dispositivo come alternativa, ad esempio o. Crea un dispositivo virtuale con Amazon EC2 Usa il tuo PC o Mac Windows o Linux come dispositivo AWS IoT

Configurare il dispositivo

L'obiettivo di questo passaggio è raccogliere ciò che è necessario per configurare il dispositivo in modo che possa avviare il sistema operativo (OS), connettersi a Internet e consentire di interagire con esso tramite un'interfaccia a riga di comando.

Per completare questo tutorial, è necessario quanto segue:

- Un Account AWS. Se non disponi dell'account, effettua la procedura descritta in Configurare Account AWS prima di continuare.
- Un modello Raspberry Pi 3 B o più recente. Questo potrebbe funzionare su versioni precedenti di Raspberry Pi, ma non sono state testate.
- Raspberry Pi OS (32 bit) o versione successiva. Ti consigliamo di utilizzare sempre la versione più recente di Raspberry Pi OS. Le versioni precedenti del sistema operativo potrebbero funzionare, ma non sono state testate.

  Per eseguire questo esempio, non è necessario installare il desktop con l'interfaccia grafica utente (GUI); tuttavia, se sei nuovo di Raspberry Pi e l'hardware Raspberry Pi lo supporta, l'utilizzo del desktop con la GUI potrebbe essere più semplice.
- Una WiFi connessione Ethernet o.

- Tastiera, mouse, monitor, cavi, alimentatori e altro hardware richiesto dal dispositivo.

> **Important**
>
> Prima di continuare con la fase successiva, è necessario che il sistema operativo sia installato, configurato e in esecuzione. Il dispositivo deve essere connesso a Internet e deve essere possibile accedere al dispositivo utilizzando la sua interfaccia da riga di comando. L'accesso della riga di comando può avvenire tramite tastiera, mouse e monitor collegati direttamente o utilizzando un'interfaccia remota del terminale SSH.

Se sul Raspberry Pi è in esecuzione un sistema operativo che dispone di un'interfaccia utente grafica (GUI), apri una finestra di terminale sul dispositivo ed esegui le seguenti istruzioni in tale finestra. In caso contrario, se ci si connette al dispositivo utilizzando un terminale remoto, ad esempio PuTTY, apri un terminale remoto sul dispositivo e utilizzalo.

Installa gli strumenti e le librerie necessari per AWS IoT Device SDK

Prima di installare il AWS IoT Device SDK e il codice di esempio, assicurati che il sistema sia aggiornato e disponga degli strumenti e delle librerie necessari per installare. SDKs

1. Aggiorna il sistema operativo e installa le librerie richieste

   Prima di installare un AWS IoT Device SDK, esegui questi comandi in una finestra di terminale del dispositivo per aggiornare il sistema operativo e installare le librerie richieste.

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get upgrade
   ```

   ```
   sudo apt-get install cmake
   ```

   ```
   sudo apt-get install libssl-dev
   ```

2. Installa Git

   Se nel sistema operativo del tuo dispositivo non è installato Git, devi installarlo per installare AWS IoT Device SDK for JavaScript.

a. Verifica se Git è già installato eseguendo questo comando.

```
git --version
```

b. Se il comando precedente restituisce la versione Git, Git è già installato ed è possibile passare alla fase 3.

c. Se viene visualizzato un errore quando esegui il comando git, installa Git eseguendo questo comando.

```
sudo apt-get install git
```

d. Prova di nuovo per vedere se Git è installato eseguendo questo comando.

```
git --version
```

e. Se Git è installato, passa alla sezione successiva. In caso contrario, risolvi e correggi l'errore prima di continuare. Hai bisogno di Git per installare l'SDK AWS IoT del dispositivo. JavaScript

Installa AWS IoT Device SDK

Installa l'SDK AWS IoT del dispositivo.

Python

In questa sezione installerai Python, i suoi strumenti di sviluppo e AWS IoT Device SDK for Python sul tuo dispositivo. Queste istruzioni sono per un Raspberry Pi con il sistema operativo Raspberry Pi più recente. Se disponi di un altro dispositivo o utilizzi un altro sistema operativo, potrebbe essere necessario adattare queste istruzioni per il dispositivo.

1. Installa Python e i suoi strumenti di sviluppo

   Il AWS IoT Device SDK for Python richiede l'installazione di Python v3.5 o versione successiva sul tuo Raspberry Pi.

   In una finestra di terminale del dispositivo, esegui questi comandi.

   1. Esegui questo comando per determinare la versione di Python installata sul dispositivo.

```
python3 --version
```

Se Python è installato, mostrerà la sua versione.

2. Se la versione visualizzata è `Python 3.5` o superiore, puoi passare alla fase 2.
3. Se la versione visualizzata è inferiore a `Python 3.5`, puoi installare la versione corretta eseguendo questo comando.

```
sudo apt install python3
```

4. Esegui questo comando per confermare che la versione corretta di Python è ora installata.

```
python3 --version
```

2. Test per pip3

In una finestra di terminale del dispositivo, esegui questi comandi.

1. Esegui questo comando per vedere se pip3 è installato.

```
pip3 --version
```

2. Se il comando restituisce un numero di versione, pip3 è installato e puoi passare alla fase 3.
3. Se il comando precedente restituisce un errore, esegui questo comando per installare pip3.

```
sudo apt install python3-pip
```

4. Esegui questo comando per vedere se pip3 è installato.

```
pip3 --version
```

3. Installa l'attuale AWS IoT Device SDK per Python

Installa AWS IoT Device SDK for Python e scarica le app di esempio sul tuo dispositivo.

Sul dispositivo, esegui questi comandi.

```
cd ~
```

```
python3 -m pip install awsiotsdk
```

```
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

JavaScript

In questa sezione installerai Node.js, il gestore di pacchetti npm e AWS IoT Device SDK for JavaScript sul tuo dispositivo. Queste istruzioni sono per un Raspberry Pi che esegue il sistema operativo Raspberry Pi. Se disponi di un altro dispositivo o utilizzi un altro sistema operativo, potrebbe essere necessario adattare queste istruzioni per il dispositivo.

1. Installare la versione più recente di Node.js

   Il AWS IoT Device SDK for JavaScript richiede l'installazione di Node.js e del gestore di pacchetti npm sul tuo Raspberry Pi.

   a. Scarica la versione più recente del repository Node immettendo questo comando.

   ```
   cd ~
   curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
   ```

   b. Installa Node e npm.

   ```
   sudo apt-get install -y nodejs
   ```

   c. Verifica l'installazione di Node.

   ```
   node -v
   ```

   Verifica che il comando visualizzi la versione di Node. Questo tutorial richiede Node 10.0 o versioni successive. Se la versione di Node non viene visualizzata, prova a scaricare di nuovo il repository di Node.

   d. Verifica l'installazione di npm.

   ```
   npm -v
   ```

   Conferma che il comando visualizzi la versione di npm. Se la versione npm non viene visualizzata, provare a installare nuovamente Node e npm.

e. Riavviare il dispositivo.

```
sudo shutdown -r 0
```

Continua dopo il riavvio del dispositivo.

2. Installa il Device SDK per AWS IoT JavaScript

Installa il AWS IoT Device SDK for JavaScript sul tuo Raspberry Pi.

a. Clona il AWS IoT Device SDK for JavaScript repository nella directory della `aws-iot-device-sdk-js-v2` tua directory. *home* Sul Raspberry Pi, la *home* directory è`~/`, che viene utilizzata come *home* directory nei seguenti comandi. Se il dispositivo utilizza un percorso diverso per la *home* directory, è necessario sostituirlo `~/` con il percorso corretto per il dispositivo nei seguenti comandi.

Questi comandi creano la directory `~/aws-iot-device-sdk-js-v2` e copiano il codice SDK in esso.

```
cd ~
git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
```

b. Passa alla directory `aws-iot-device-sdk-js-v2` creata nella fase precedente ed esegui `npm install` per installare l'SDK. Il comando `npm install` chiama la compilazione della libreria `aws-crt`, operazione che può richiedere alcuni minuti.

```
cd ~/aws-iot-device-sdk-js-v2
npm install
```

Installazione ed esecuzione dell'applicazione di esempio

In questa sezione, installerai ed eseguirai l'app di `pubsub` esempio disponibile nel AWS IoT Device SDK. Questa app mostra come il tuo dispositivo utilizza la libreria MQTT per pubblicare ed effettuare la sottoscrizione ai messaggi MQTT. L'app di esempio sottoscrive un argomento `topic_1`, pubblica 10 messaggi in tale argomento e visualizza i messaggi ricevuti dal broker di messaggi.

Installazione dei file di certificato

L'app di esempio richiede i file di certificato che autenticano il dispositivo da installare sul dispositivo.

Per installare i file di certificato del dispositivo per l'app di esempio

1. Crea una `certs` sottodirectory nella tua *home* directory eseguendo questi comandi.

   ```
   cd ~
   mkdir certs
   ```

2. Copia nella directory `~/certs` il certificato, la chiave privata e il certificato CA root creato precedentemente in the section called “Crea AWS IoT risorse”.

   La modalità di copia dei file di certificato sul dispositivo dipende dal dispositivo e dal sistema operativo e non è descritto qui. Tuttavia, se il dispositivo supporta un'interfaccia utente grafica (GUI) e dispone di un browser Web, è possibile eseguire la procedura descritta in the section called “Crea AWS IoT risorse” dal browser web del dispositivo per scaricare i file risultanti direttamente sul dispositivo.

   I comandi della sezione successiva presuppongono che i file di chiave e certificato siano memorizzati sul dispositivo, come illustrato in questa tabella.

   Nomi dei file dei certificati

File	Percorso del file
Un certificato emesso da una CA root	`~/certs/Amazon-root-CA-1.pem`
Certificato del dispositivo	`~/certs/device.pem.crt`
Chiave privata	`~/certs/private.pem.key`

Per eseguire l'app di esempio sono necessarie le seguenti informazioni:

Valori dei parametri dell'applicazione

Parametro	Dove trovare il valore
`your-iot-endpoint`	Nella console AWS IoT, scegli All devices (Tutti i dispositivi) e quindi Things (Oggetti). Nella pagina Impostazioni del AWS IoT menu. L'endpoint viene visualizzato nella sezione

Parametro	Dove trovare il valore
	Device data endpoint (Endpoint dei dati del dispositivo).

Il *your-iot-endpoint* valore ha un formato di:*endpoint_id*-ats.iot.*region*.amazonaws.com, ad esempio,a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com.

Python

Per installare ed eseguire l'app di esempio

1. Passa alla directory delle app di esempio.

```
cd ~/aws-iot-device-sdk-python-v2/samples
```

2. Nella finestra della riga di comando, sostituisci *your-iot-endpoint* come indicato ed esegui questo comando.

```
python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

3. Osserva che l'applicazione di esempio:

    1. Si connette al AWS IoT servizio del tuo account.
    2. Sottoscrive l'argomento del messaggio topic_1 e visualizza i messaggi ricevuti su tale argomento.
    3. Pubblica 10 messaggi sull'argomento topic_1.
    4. L'output sia simile a quello riportato di seguito:

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to topic 'topic_1'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
```

```
Publishing message to topic 'topic_1': Hello World! [1]
Received message from topic 'topic_1': b'Hello World! [1]'
Publishing message to topic 'topic_1': Hello World! [2]
Received message from topic 'topic_1': b'Hello World! [2]'
Publishing message to topic 'topic_1': Hello World! [3]
Received message from topic 'topic_1': b'Hello World! [3]'
Publishing message to topic 'topic_1': Hello World! [4]
Received message from topic 'topic_1': b'Hello World! [4]'
Publishing message to topic 'topic_1': Hello World! [5]
Received message from topic 'topic_1': b'Hello World! [5]'
Publishing message to topic 'topic_1': Hello World! [6]
Received message from topic 'topic_1': b'Hello World! [6]'
Publishing message to topic 'topic_1': Hello World! [7]
Received message from topic 'topic_1': b'Hello World! [7]'
Publishing message to topic 'topic_1': Hello World! [8]
Received message from topic 'topic_1': b'Hello World! [8]'
Publishing message to topic 'topic_1': Hello World! [9]
Received message from topic 'topic_1': b'Hello World! [9]'
Publishing message to topic 'topic_1': Hello World! [10]
Received message from topic 'topic_1': b'Hello World! [10]'
10 message(s) received.
Disconnecting...
Disconnected!
```

In caso di problemi nell'esecuzione dell'app di esempio, consulta the section called “Risolvi i problemi con l'applicazione di esempio”.

È inoltre possibile aggiungere il parametro `--verbosity Debug` alla riga di comando in modo che l'app di esempio visualizzi messaggi dettagliati su ciò che sta facendo. Tali informazioni potrebbero fornirti l'aiuto necessario per risolvere il problema.

JavaScript

Per installare ed eseguire l'app di esempio

1. Nella finestra a riga di comando passa alla directory `~/aws-iot-device-sdk-js-v2/samples/node/pub_sub` che l'SDK ha creato e installa l'app di esempio utilizzando questi comandi. Il comando `npm install` chiama la compilazione della libreria `aws-crt`, operazione che può richiedere alcuni minuti.

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
```

```
npm install
```

2. Nella finestra della riga di comando, sostituisci *your-iot-endpoint* come indicato ed esegui questo comando.

```
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

3. Osserva che l'applicazione di esempio:

   1. Si connette al AWS IoT servizio del tuo account.
   2. Sottoscrive l'argomento del messaggio topic_1 e visualizza i messaggi ricevuti su tale argomento.
   3. Pubblica 10 messaggi sull'argomento topic_1.
   4. L'output sia simile a quello riportato di seguito:

```
Publish received on topic topic_1
{"message":"Hello world!","sequence":1}
Publish received on topic topic_1
{"message":"Hello world!","sequence":2}
Publish received on topic topic_1
{"message":"Hello world!","sequence":3}
Publish received on topic topic_1
{"message":"Hello world!","sequence":4}
Publish received on topic topic_1
{"message":"Hello world!","sequence":5}
Publish received on topic topic_1
{"message":"Hello world!","sequence":6}
Publish received on topic topic_1
{"message":"Hello world!","sequence":7}
Publish received on topic topic_1
{"message":"Hello world!","sequence":8}
Publish received on topic topic_1
{"message":"Hello world!","sequence":9}
Publish received on topic topic_1
{"message":"Hello world!","sequence":10}
```

In caso di problemi nell'esecuzione dell'app di esempio, consulta the section called “Risolvi i problemi con l'applicazione di esempio”.

È inoltre possibile aggiungere il parametro `--verbosity Debug` alla riga di comando in modo che l'app di esempio visualizzi messaggi dettagliati su ciò che sta facendo. Tali informazioni potrebbero fornirti l'aiuto necessario per risolvere il problema.

Visualizza i messaggi dell'app di esempio nella AWS IoT console

È possibile visualizzare i messaggi dell'app di esempio durante il passaggio attraverso il broker di messaggi utilizzando il client di test MQTT nella console AWS IoT .

Per visualizzare i messaggi MQTT pubblicati dall'app di esempio

1. Verificare Visualizza i messaggi MQTT con il AWS IoT client MQTT. In questo modo si impara come utilizzare il client di test MQTT nella console AWS IoT per visualizzare i messaggi MQTT durante il passaggio attraverso il broker di messaggi.
2. Apri il client di test MQTT nella console AWS IoT .
3. Effettua la sottoscrizione all'argomento topic_1.
4. Nella finestra a riga di comando, esegui di nuovo l'app di esempio e guarda i messaggi nel Client MQTT nella console AWS IoT .

   Python

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
   cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
   endpoint
   ```

   JavaScript

   ```
   cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
   node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
   cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
   endpoint
   ```

Risolvi i problemi con l'applicazione di esempio

Se si verifica un errore quando tenti di eseguire l'app di esempio, ecco alcune cose da verificare.

Controlla il certificato

Se il certificato non è attivo, non AWS IoT accetterà alcun tentativo di connessione che lo utilizzi per l'autorizzazione. Quando crei il tuo certificato, potrebbe capitarti di trascurare il bottone Activate (Attiva). Fortunatamente, puoi attivare il certificato dalla console AWS IoT.

Per controllare l'attivazione del certificato

1. Nella console AWS IoT, nel menu a sinistra, seleziona Secure (Sicurezza), quindi scegli Certificates (Certificati).
2. Nell'elenco dei certificati, individua il certificato creato per l'esercizio e verificane lo stato nella colonna Status (Stato).

   Se non ti ricordi il nome del certificato, verifica se sono presenti Inactive (Inattivo) per vedere se potrebbero essere quelli che stai usando.

   Scegli il certificato nell'elenco per aprire la relativa pagina dei dettagli. Nella pagina dettagli, puoi vedere la Create date (Creare data) per informazioni sull'identificazione del certificato.
3. Per attivare un certificato inattivo nella pagina dei dettagli del certificato, seleziona Actions (Operazioni) e poi Activate (Attiva).

Se hai trovato il certificato corretto e il relativo certificato è attivo, ma si verificano ancora problemi nell'esecuzione dell'app di esempio, verifica la sua policy come descritto nel passaggio successivo.

È inoltre possibile provare a creare un nuovo oggetto e un nuovo certificato seguendo la procedura descritta in the section called “Crea un oggetto”. Se crei un nuovo oggetto, dovrai assegnarli un nuovo nome e scaricare i nuovi file del certificato sul tuo dispositivo.

Controlla la policy collegata al certificato

Le politiche autorizzano le azioni in AWS IoT. Se il certificato utilizzato per la connessione ad AWS IoT non dispone di una policy o non ha una policy che gli consenta di connettersi, la connessione verrà rifiutata, anche se il certificato è attivo.

Per controllare i criteri collegati a un certificato

1. Individua il certificato come descritto nell'elemento precedente e apri la relativa pagina dei dettagli.

2. Nel menu a sinistra della pagina dei dettagli del certificato, scegli Policies (Policy) per visualizzare le policy collegate al certificato.
3. Se al certificato non sono associate policy, aggiungine una scegliendo il menu Actions (Operazioni), quindi scegli Attach policy (Collega policy).

   Scegli la policy che hai creato in precedenza in the section called “Crea AWS IoT risorse”.
4. Se è associata una policy, scegli il riquadro policy per aprire la pagina dei dettagli.

   Nella pagina dei dettagli, rivedi Policy document (Documento di policy) per assicurarti che contenga le stesse informazioni di quello creato in the section called “Crea una AWS IoT politica”.

Controlla la riga di comando

Assicurati di aver usato la riga di comando corretta per il tuo sistema. I comandi utilizzati sui sistemi Linux e macOS sono spesso diversi da quelli utilizzati nei sistemi Windows.

Controlla l'indirizzo dell'endpoint

Esamina il comando immesso e ricontrolla l'indirizzo dell'endpoint nel comando con quello nella console AWS IoT.

Controlla i nomi dei file del certificato

Confronta i nomi dei file nel comando inserito con i nomi dei file dei certificati nella directory `certs`.

Alcuni sistemi potrebbero richiedere che i nomi dei file siano tra virgolette per funzionare correttamente.

Controlla l'installazione dell'SDK

Assicurati che l'installazione dell'SDK sia completa e corretta.

In caso di dubbio, reinstalla l'SDK sul dispositivo. Nella maggior parte dei casi, è sufficiente trovare la sezione del tutorial intitolata Installa l'SDK del AWS IoT dispositivo ***`SDK language`*** e seguire nuovamente la procedura.

Se utilizzi AWS IoT Device SDK per JavaScript, ricordati di installare le app di esempio prima di provare a eseguirle. L'installazione dell'SDK non installa automaticamente le app di esempio. Le app di esempio devono essere installate manualmente dopo l'installazione dell'SDK.

Visualizza i messaggi MQTT con il AWS IoT client MQTT

Questa sezione descrive come utilizzare il client di test AWS IoT MQTT nella AWS IoT console per guardare i messaggi MQTT inviati e ricevuti da. AWS IoT L'esempio utilizzato in questa sezione si riferisce agli esempi utilizzati inGuida introduttiva ai AWS IoT Core tutorial; tuttavia, è possibile sostituire quello `topicName` utilizzato negli esempi con qualsiasi nome di argomento o filtro di argomento utilizzato dalla soluzione IoT.

I dispositivi pubblicano messaggi MQTT identificati da argomenti a cui comunicare il loro stato e AWS IoT pubblicano messaggi MQTT per informare i dispositivi e le app di modifiche ed eventi. AWS IoTÈ possibile utilizzare il client MQTT per sottoscrivere questi argomenti e controllare i messaggi mentre si verificano. È inoltre possibile utilizzare il client di test MQTT per pubblicare messaggi MQTT su dispositivi e servizi sottoscritti nel proprio. Account AWS

Indice

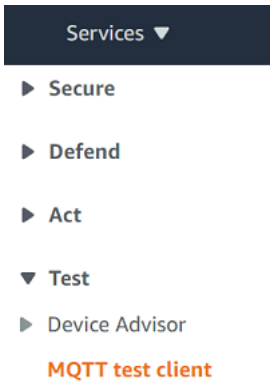
- Visualizzazione dei messaggi MQTT nel client MQTT
- Pubblicazione di messaggi MQTT dal client MQTT
- Test delle sottoscrizioni condivise nel client MQTT

Visualizzazione dei messaggi MQTT nel client MQTT

La procedura seguente spiega come sottoscrivere un argomento MQTT specifico su cui il dispositivo pubblica messaggi e come visualizzarli nella console.AWS IoT

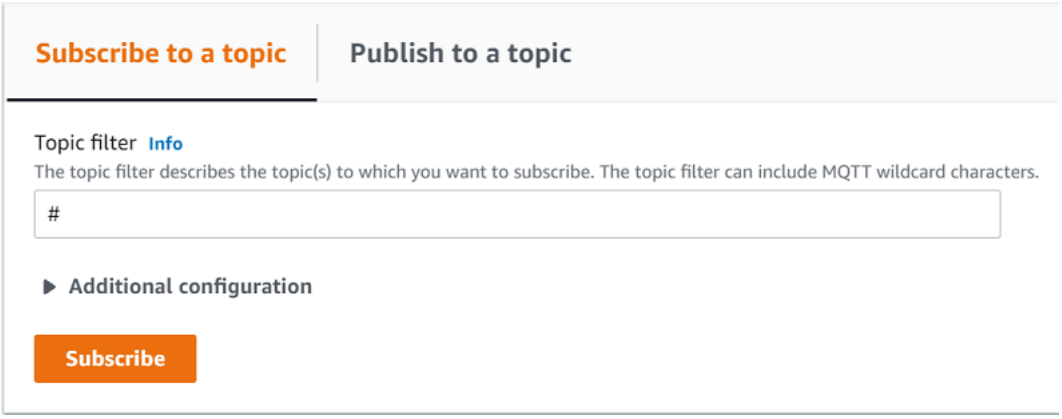
Per visualizzare i messaggi MQTT nel client di test MQTT

1. Nella console AWS IoT e, nel menu a sinistra, scegli Test quindi scegli MQTT test client (Client MQTT di test).

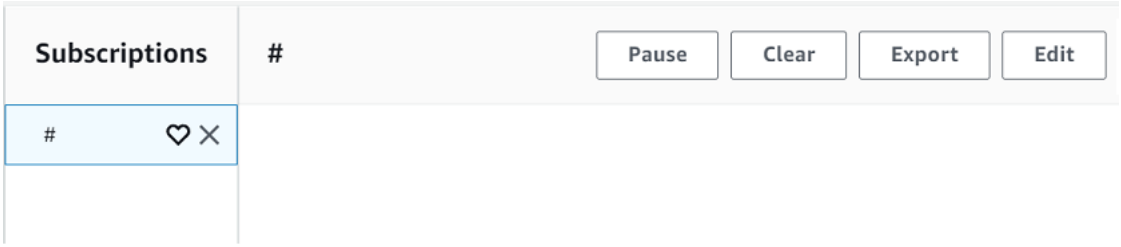


2. Nella scheda Sottoscrivi a un argomento, inserisci *topicName* per iscriverti all'argomento su cui viene pubblicato il tuo dispositivo. Per l'app di esempio introduttiva, effettua la sottoscrizione a **#**, che sottoscrive tutti gli argomenti dei messaggi.

   Continuando con l'esempio introduttivo, nella scheda Subscribe to a topic (Sottoscrizione a un argomento), nel campo Topic filter (Filtro di argomenti), immetti **#**, quindi scegli Subscribe (Sottoscrizione).



   La pagina di log dei messaggi dell'argomento # si apre e **#** viene visualizzato nell'elenco Subscriptions (Abbonamenti). Se il dispositivo in cui hai the section called “Configurazione del dispositivo” configurato esegue il programma di esempio, dovresti vedere i messaggi a cui invia i messaggi AWS IoT nel registro #. Le voci del registro dei messaggi verranno visualizzate sotto la sezione Pubblica quando i messaggi con l'argomento sottoscritto vengono ricevuti da AWS IoT.



3. Sulla pagina log dei messaggi #, puoi pubblicare messaggi in un argomento, ma sarà necessario specificare il nome dell'argomento. Non è possibile pubblicare nell'argomento #.

   I messaggi pubblicati per gli argomenti sottoscritti vengono visualizzati nel registro messaggi quando vengono ricevuti, con il messaggio più recente all'inizio.

Risoluzione dei problemi relativi ai messaggi MQTT

Usa il filtro per argomenti jolly

Se i messaggi non vengono visualizzati come previsto, prova a sottoscrivere un argomento con caratteri jolly come descritto in Filtri per i nomi degli argomenti. Il filtro argomento jolly multi-livello MQTT è il segno hash o cancelletto (#) e può essere utilizzato come filtro per argomenti nel campo Subscription topic (Argomento sottoscrizione).

Sottoscrizione alla filtro argomenti # sottoscrive ogni argomento ricevuto dal broker di messaggi. È possibile restringere il filtro sostituendo gli elementi del percorso del filtro dell'argomento con un carattere jolly a più livelli # o il carattere jolly a livello singolo '+'.

Quando utilizzare i caratteri jolly in un filtro argomento

- Il carattere jolly multilivello deve essere l'ultimo carattere nel filtro argomento.
- Il percorso del filtro argomento può avere un solo carattere jolly a livello singolo per livello di argomento.

Ad esempio:

Filtro di argomenti	Visualizza i messaggi con
#	Nome di qualsiasi argomento
`topic_1/#`	Un nome di argomento che inizia con `topic_1/`
`topic_1/level_2/#`	Un nome di argomento che inizia con `topic_1/level_2/`
`topic_1/+/level_3`	Un nome di argomento che inizia con `topic_1/`, termina con `/level_3` e ha un elemento di qualsiasi valore in mezzo.

Per ulteriori informazioni sui filtri argomento, consulta Filtri per i nomi degli argomenti.

Controlla la presenza di errori di nome dell'argomento

I nomi degli argomenti e i filtri degli argomenti MQTT distinguono tra maiuscole e minuscole. Ad esempio, se il dispositivo pubblica messaggi su `Topic_1` (con la lettera maiuscola T) invece di `topic_1`, l'argomento a cui hai effettuato la sottoscrizione e i relativi messaggi non verranno

visualizzato nel client di test MQTT. La sottoscrizione filtro di argomento del carattere jolly, tuttavia, mostrerebbe che il dispositivo sta pubblicando messaggi e si potrebbe vedere che utilizzava un nome di argomento che non era quello che ti aspettavi.

Pubblicazione di messaggi MQTT dal client MQTT

Per pubblicare un messaggio in un argomento MQTT

1. Nella pagina del client di test MQTT, nella scheda Pubblica su un argomento, nel campo Nome argomento, inserisci il *topicName* messaggio. Per questo esempio, utilizzare **my/topic**.

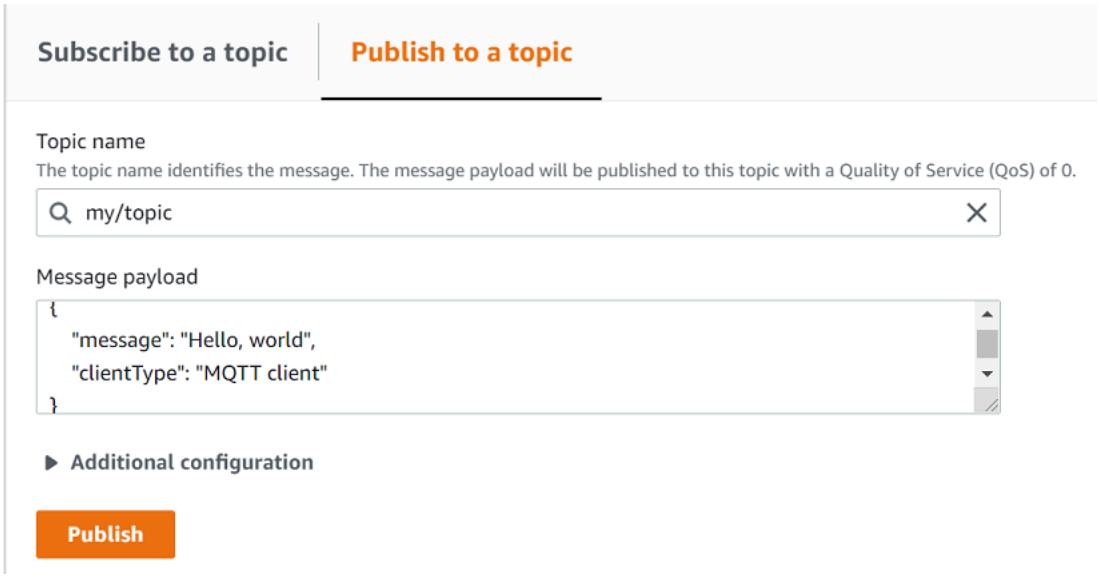
   > **Note**
   >
   > Non utilizzare informazioni di identificazione personale nei nomi degli argomenti, a prescindere che vengano utilizzati nel client di test MQTT o nell'implementazione del sistema. I nomi dell'argomento possono essere visualizzati nelle comunicazioni e nei report non crittografati.

2. Nella finestra relativa al payload del messaggio digita il codice JSON seguente:

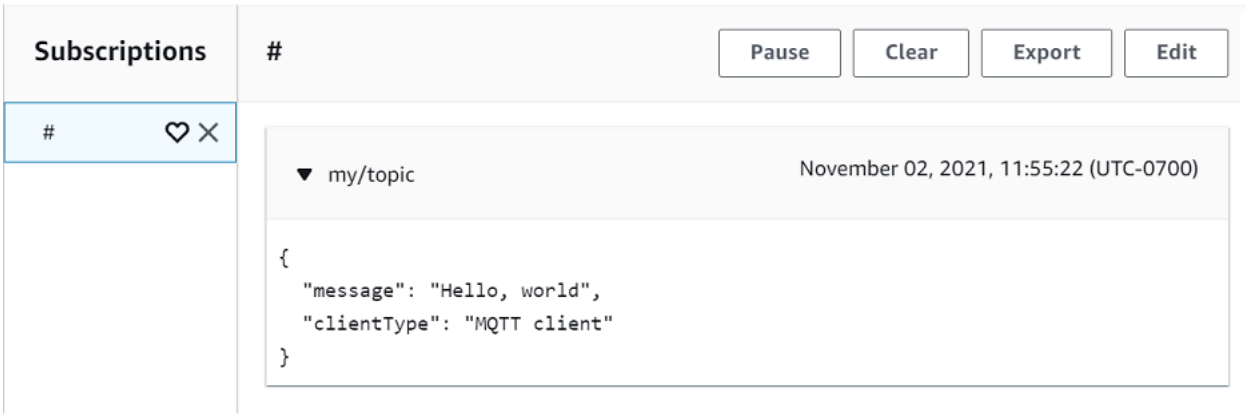
   ```
   {
       "message": "Hello, world",
       "clientType": "MQTT test client"
   }
   ```

3. Scegli Publish (Pubblica) per pubblicare il messaggio in AWS IoT.

   > **Note**
   >
   > Assicurati di essere iscritto a my/topic prima di pubblicare il messaggio.



4. Scegli my/topic nell'elenco Subscription (Sottoscrizione) per visualizzare il messaggio. Il messaggio deve essere visualizzato nel client di test MQTT sotto la finestra di payload del messaggio di pubblicazione.



È possibile pubblicare messaggi MQTT su altri argomenti modificando il *topicName* campo Nome argomento e scegliendo il pulsante Pubblica.

> **Important**
>
> Quando crei più abbonamenti con argomenti sovrapposti (ad esempio, sonda1/temperatura e sonda1/#), è possibile che un singolo messaggio pubblicato su un argomento che corrisponde a entrambi gli abbonamenti venga recapitato più volte, una volta per ogni abbonamento sovrapposto.

Test delle sottoscrizioni condivise nel client MQTT

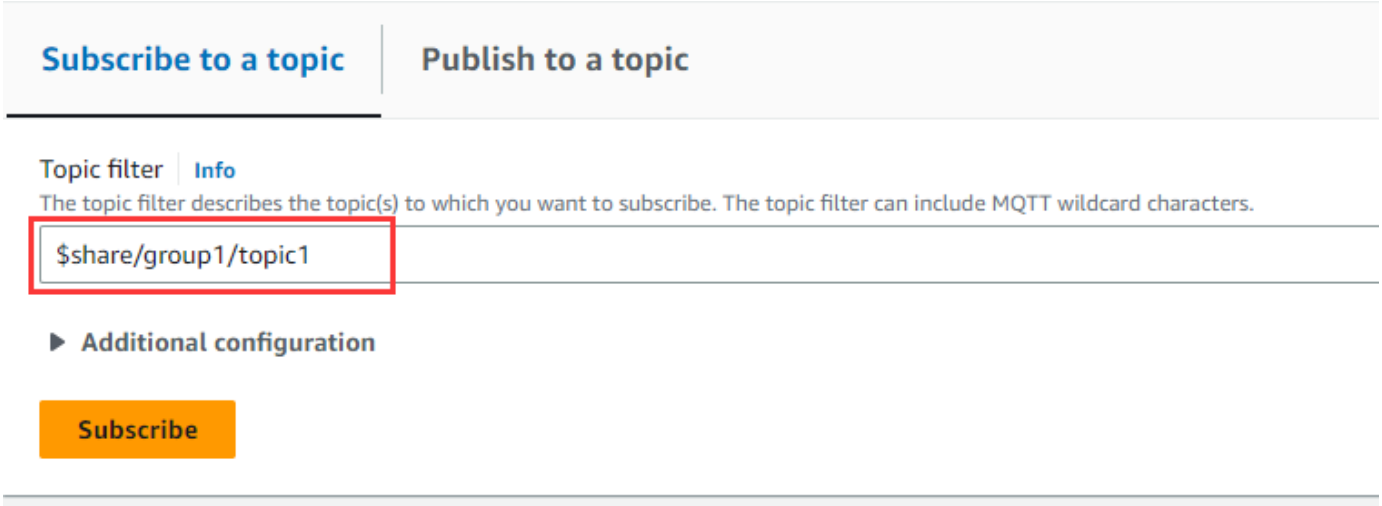
Questa sezione descrive come utilizzare il client AWS IoT MQTT nella AWS IoT console per guardare i messaggi MQTT inviati e ricevuti AWS IoT utilizzando sottoscrizioni condivise. ???consente a più client di condividere un abbonamento a un argomento con un solo client che riceve i messaggi pubblicati su quell'argomento utilizzando una distribuzione casuale. Per simulare più client MQTT (in questo esempio, due client MQTT) che condividono lo stesso abbonamento, aprite il client AWS IoT MQTT nella AWS IoT console da più browser Web. L'esempio utilizzato in questa sezione non fa riferimento agli esempi utilizzati in Guida introduttiva ai AWS IoT Core tutorial. Per ulteriori informazioni, consultare Sottoscrizioni condivise.

Per condividere una sottoscrizione a un argomento MQTT

1. Nella Console AWS IoT, nel pannello di navigazione, scegli Test, quindi seleziona Client MQTT di test.
2. Nella scheda Sottoscrivi a un argomento, inserisci *topicName* per iscriverti all'argomento su cui viene pubblicato il tuo dispositivo. Per utilizzare sottoscrizioni condivise, effettua la sottoscrizione al filtro di argomenti di una sottoscrizione condivisa come illustrato di seguito:

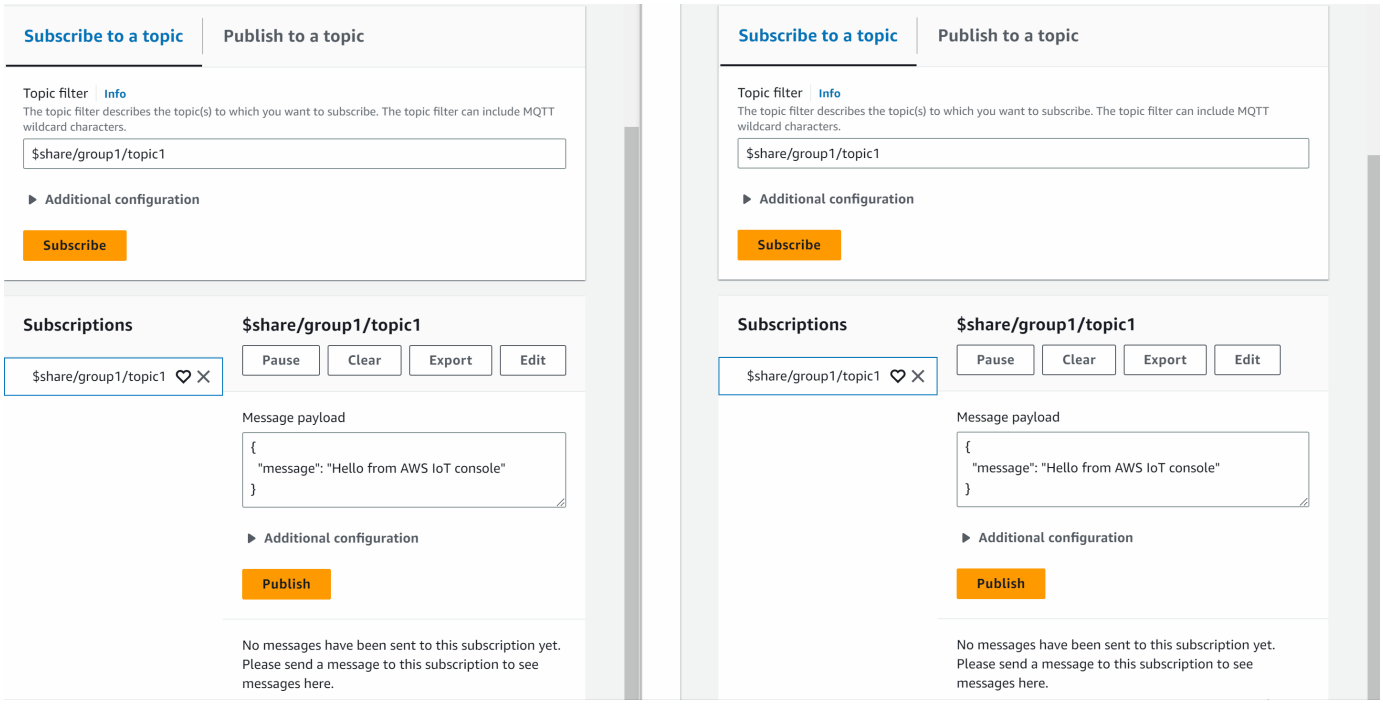
   ```
   $share/{ShareName}/{TopicFilter}
   ```

   Un filtro di argomenti di esempio può essere **$share/group1/topic1**, che effettua la sottoscrizione all'argomento del messaggio **topic1**.

3. Apri un altro browser Web e ripeti i passaggi 1 e 2. Questa procedura simula due diversi client MQTT che condividono la stessa sottoscrizione **$share/group1/topic1**.

4. Scegli un client MQTT, nella scheda Pubblica su un argomento, nel campo Nome argomento, inserisci il *topicName* messaggio. Per questo esempio, utilizzare **topic1**. Prova a pubblicare il messaggio alcune volte. Dall'elenco Sottoscrizioni di entrambi i client MQTT, devi essere in grado di vedere che i client ricevono il messaggio utilizzando una distribuzione casuale. In questo esempio, lo stesso messaggio "Hello from AWS IoT console" viene pubblicato tre volte. Il client MQTT a sinistra ha ricevuto il messaggio due volte e il client MQTT a destra ha ricevuto il messaggio una volta.



AWS IoT tutorial

I AWS IoT tutorial sono suddivisi in due percorsi di apprendimento per supportare due obiettivi diversi. Scegli il percorso di apprendimento migliore per il tuo obiettivo.

- Vuoi creare un' proof-of-conceptidea per testare o dimostrare un' AWS IoT idea di soluzione

  Per dimostrare le attività e le applicazioni IoT comuni che utilizzano AWS IoT Device Client sui tuoi dispositivi, segui il percorso di the section called “Creazione di demo con AWS IoT Device Client” apprendimento. Il AWS IoT Device Client fornisce un software per dispositivi con cui è possibile applicare le proprie risorse cloud per dimostrare una end-to-end soluzione con uno sviluppo minimo.

  Per informazioni sul AWS IoT Device Client, consulta AWS IoT Device Client.
- Vuoi imparare come creare software di produzione per implementare la tua soluzione

  Per creare la tua soluzione software che soddisfi i tuoi requisiti specifici utilizzando un AWS IoT Device SDK, segui il percorso di the section called “Creazione di soluzioni con il AWS IoT dispositivo SDKs” apprendimento.

  Per informazioni sul AWS IoT dispositivo disponibile SDKs, consulta???. Per informazioni su AWS SDKs, vedere Strumenti su cui costruire AWS.

AWS IoT opzioni del percorso di apprendimento del tutorial

- Creazione di demo con AWS IoT Device Client
- Creazione di soluzioni con il AWS IoT dispositivo SDKs

Creazione di demo con AWS IoT Device Client

I tutorial di questo percorso di apprendimento illustrano i passaggi necessari per sviluppare un software dimostrativo utilizzando Device Client. AWS IoT Il AWS IoT Device Client fornisce un software che funziona sul tuo dispositivo IoT per testare e dimostrare aspetti di una soluzione IoT basata su AWS IoT.

L'obiettivo di questi tutorial è facilitare l'esplorazione e la sperimentazione, in modo che possiate avere la certezza di AWS IoT supportare la vostra soluzione prima di sviluppare il software del dispositivo.

Cosa imparerai in questi tutorial:

- Come preparare un Raspberry Pi da utilizzare come dispositivo IoT con AWS IoT
- Come dimostrare AWS IoT le funzionalità utilizzando il AWS IoT Device Client sul dispositivo

In questo percorso di apprendimento, installerai il AWS IoT Device Client sul tuo Raspberry Pi e creerai le AWS IoT risorse nel cloud per dimostrare idee di soluzioni IoT. Mentre i tutorial di questo percorso di apprendimento mostrano le caratteristiche utilizzando un Raspberry Pi, spiegano gli obiettivi e le procedure per aiutarvi ad adattarli ad altri dispositivi.

Prerequisiti per creare demo con Device Client AWS IoT

Questa sezione descrive di cosa hai bisogno prima di iniziare i tutorial in questo percorso di apprendimento.

Per completare i tutorial in questo percorso di apprendimento, è necessario:

- Un Account AWS

  Puoi usare quello esistente Account AWS, se ne hai uno, ma potresti dover aggiungere ruoli o autorizzazioni aggiuntivi per utilizzare le AWS IoT funzionalità utilizzate da questi tutorial.

  Se devi crearne uno nuovo Account AWS, vedi. the section called “Configurare Account AWS”
- Un dispositivo Raspberry Pi o IoT compatibile

  Le esercitazioni utilizzano un Raspberry Pi perché è disponibile in diversi fattori di forma, è onnipresente ed è un dispositivo dimostrativo relativamente economico. Le esercitazioni sono state testate sul Un Raspberry Pi 3 Modello B+, il Raspberry Pi 4 Modello B e su un'istanza Amazon EC2 che esegue Ubuntu Server 20.04 LTS (HVM). Per utilizzare AWS CLI ed eseguire i comandi, ti consigliamo di utilizzare la versione più recente del sistema operativo Raspberry Pi (sistema operativo Raspberry Pi (64 bit) o OS Lite). Le versioni precedenti del sistema operativo potrebbero funzionare, ma non le abbiamo testate.

> **Note**
>
> I tutorial spiegano gli obiettivi di ogni fase per aiutarti ad adattarli all'hardware IoT su cui non li abbiamo provati; tuttavia, non descrivono specificamente come adattarli ad altri dispositivi.

- Dimestichezza con il sistema operativo del dispositivo IoT

  I passaggi di questi tutorial presuppongono che tu abbia familiarità con l'uso di comandi e operazioni Linux di base dall'interfaccia della riga di comando supportata da un Raspberry Pi. Se non hai familiarità con queste operazioni, concediti più tempo per completare i tutorial.

  Per completare questi tutorial, dovresti già saper:

  - Eseguire in modo sicuro le operazioni di base del dispositivo come assemblaggio e collegamento di componenti, collegamento del dispositivo alle fonti di alimentazione richieste e installazione e rimozione di schede di memoria.
  - Caricare e scaricare software e file di sistema sul dispositivo. Se il dispositivo non utilizza un dispositivo di archiviazione rimovibile, ad esempio una scheda microSD, dovrai sapere come connetterti al dispositivo e caricare e scaricare software e file di sistema sul dispositivo.
  - Connettere il tuo dispositivo alle reti su cui intendi utilizzarlo.
  - Connetterti al dispositivo da un altro computer utilizzando un terminale SSH o un programma simile.
  - Utilizzare un'interfaccia a riga di comando per creare, copiare, spostare, rinominare e impostare le autorizzazioni di file e directory sul dispositivo.
  - Installare nuovi programmi sul dispositivo.
  - Trasferire file da e verso il tuo dispositivo utilizzando strumenti come FTP o SCP.
- Ambiente di sviluppo e test per la tua soluzione IoT

  I tutorial descrivono il software e l'hardware richiesti; tuttavia, i tutorial presuppongono che sarai in grado di eseguire operazioni che potrebbero non essere descritte esplicitamente. Esempi di questo tipo di hardware e operazioni includono:

  - Un computer host locale su cui scaricare e archiviare file

    Per il Raspberry Pi, di solito si tratta di un personal computer o laptop in grado di leggere e scrivere su schede di memoria microSD. Il computer host locale deve:

    - Essere connesso a Internet.
    - Avere la AWS CLI installata e configurata.
    - Disponi di un browser web che supporti la AWS console.
  - Un modo per connettere il computer host locale al dispositivo per comunicare con esso, immettere comandi e trasferire file

    Sul Raspberry Pi, spesso questo viene fatto utilizzando SSH e SCP dal computer host locale.

- Connettersi al dispositivo IoT utilizzando un monitor e una tastiera

  Possono essere utili, ma non sono necessari per completare i tutorial.

- Un modo in cui il computer host locale e i dispositivi IoT possono connettersi a Internet

  Potrebbe trattarsi di una connessione di rete cablata o wireless a un router o gateway connesso a Internet. L'host locale deve anche essere in grado di connettersi al Raspberry Pi. Ciò potrebbe richiedere che si trovino sulla stessa rete locale. I tutorial non possono mostrarti come applicare queste impostazioni per la configurazione del tuo dispositivo o dispositivo specifico, ma mostrano come testare questa connettività.

- Accesso al router della rete locale per visualizzare i dispositivi collegati

  Per completare i tutorial in questo percorso di apprendimento, dovrai essere in grado di trovare l'indirizzo IP del tuo dispositivo IoT.

  Su una rete locale, questo può essere fatto accedendo all'interfaccia dell'amministratore del router di rete a cui i dispositivi si connettono. Se è possibile assegnare un indirizzo IP fisso per il dispositivo nel router, è possibile semplificare la riconnessione dopo ogni riavvio del dispositivo.

  Se hai una tastiera e un monitor collegati al dispositivo, ifconfig può visualizzare l'indirizzo IP del dispositivo.

  Se non puoi applicare nessuna di queste opzioni, dovrai trovare un modo per identificare l'indirizzo IP del dispositivo dopo ogni riavvio.

Dopo aver ottenuto tutto il materiale, prosegui con the section called “Preparazione all'utilizzo di IoT Device Client”.

Tutorial in questo percorso di apprendimento

- Tutorial: Preparazione dei dispositivi per il AWS IoT Device Client
- Tutorial: installazione e configurazione del AWS IoT Device Client
- Tutorial: Dimostrare la comunicazione dei messaggi MQTT con il AWS IoT Device Client
- Tutorial: Dimostra le azioni (lavori) remote con il AWS IoT Device Client
- Tutorial: pulizia dopo l'esecuzione dei tutorial AWS IoT Device Client

Tutorial: Preparazione dei dispositivi per il AWS IoT Device Client

Questo tutorial ti guida attraverso l'inizializzazione del tuo Raspberry Pi per prepararlo per i tutorial successivi di questo percorso di apprendimento.

L'obiettivo di questo tutorial è installare la versione corrente del sistema operativo del dispositivo e assicurarsi di poter comunicare con il dispositivo nel contesto dell'ambiente di sviluppo.

Prerequisiti

Prima di iniziare questo tutorial, assicurati di avere gli elementi elencati in the section called “Prerequisiti per creare demo con Device Client AWS IoT” disponibili e pronti per l'uso.

Il tutorial dura circa 90 minuti.

In questo tutorial, apprenderai a:

- Installa e aggiorna il sistema operativo del tuo dispositivo.
- Installa e verifica qualsiasi software aggiuntivo necessario per eseguire i tutorial.
- Verifica la connettività del tuo dispositivo e installa i certificati richiesti.

Dopo aver completato questo tutorial, il tutorial successivo prepara il dispositivo per le demo che utilizzano AWS IoT Device Client.

Procedure all'interno del tutorial

- Installa e aggiorna il sistema operativo del dispositivo
- Installa e verifica il software richiesto sul tuo dispositivo
- Testa il tuo dispositivo e salva il certificato Amazon CA

Installa e aggiorna il sistema operativo del dispositivo

Le procedure contenute in questa sezione descrivono come inizializzare la scheda microSD utilizzata da Raspberry Pi per l'unità di sistema. La scheda microSD di Raspberry Pi contiene il software del sistema operativo (OS) e lo spazio per l'archiviazione dei file applicativi. Se non utilizzi un Raspberry Pi, segui le istruzioni del dispositivo per installare e aggiornare il software del sistema operativo del dispositivo.

Dopo aver completato questa sezione, dovresti essere in grado di avviare il dispositivo IoT e connetterti al programma terminale sul computer host locale.

Equipaggiamento necessario:

- Ambiente locale di sviluppo e test
- Un Raspberry Pi o dispositivo IoT, in grado di connettersi a Internet
- Una scheda di memoria microSD con una capacità di almeno 8 GB o spazio di archiviazione sufficiente per il sistema operativo e il software richiesto.

> **Note**
>
> Quando si seleziona una scheda microSD per questi esercizi, scegline una che sia abbastanza capiente ma più piccola possibile.
> Con una scheda SD piccola sarà più veloce eseguire il backup e l'aggiornamento. Sul Raspberry Pi, non avrai bisogno di più di una scheda microSD da 8 GB per questi tutorial. Se hai bisogno di più spazio per l'applicazione specifica, i file immagine più piccoli salvati in questi tutorial possono ridimensionare il file system su una scheda più grande per utilizzare tutto lo spazio supportato della scheda scelta.

Equipaggiamento facoltativo:

- Una tastiera USB collegata al Raspberry Pi
- Monitor HDMI e cavo per connettere il monitor al Raspberry Pi

Procedure in questa sezione:

- Caricare il sistema operativo del dispositivo sulla scheda microSD
- Avvia il tuo dispositivo IoT con il nuovo sistema operativo
- Collega il dispositivo al computer host locale

Caricare il sistema operativo del dispositivo sulla scheda microSD

Questa procedura utilizza il computer host locale per caricare il sistema operativo del dispositivo su una scheda microSD.

> **Note**
>
> Se il dispositivo non utilizza un supporto di memorizzazione rimovibile per il sistema operativo, installa il sistema operativo utilizzando la procedura per quel dispositivo e prosegui con the section called “Avvia il tuo dispositivo IoT”.

Per installare il sistema operativo sul tuo Raspberry Pi

1. Sul computer host locale, scarica e decomprimi l'immagine del sistema operativo Raspberry Pi che si desidera utilizzare. Le versioni più recenti sono disponibili su https://www.raspberrypi.com/software/operating-systems/

   Scegliere una versione di Raspberry Pi OS

   Questo tutorial utilizza la versione Raspberry Pi OS Lite perché è la versione più piccola che supporta questi tutorial in questo percorso di apprendimento. Questa versione del sistema operativo Raspberry Pi ha solo un'interfaccia a riga di comando e non dispone di un'interfaccia utente grafica. Una versione dell'ultimo sistema operativo Raspberry Pi con un'interfaccia utente grafica funzionerà anche con questi tutorial; tuttavia, le procedure descritte in questo percorso di apprendimento utilizzano solo l'interfaccia della riga di comando per eseguire operazioni su Raspberry Pi.
2. Inserisci la microSD nel computer host locale.
3. Utilizzando uno strumento di imaging per schede SD, scrivi il file immagine del sistema operativo decompresso sulla scheda microSD.
4. Dopo aver scritto l'immagine del sistema operativo Raspberry Pi sulla microSD:

   a. Apri la partizione BOOT sulla microSD in una finestra della riga di comando o in una finestra Esplora file.

   b. Nella partizione BOOT della microSD, nella directory principale, creare un file vuoto denominato `ssh` senza estensione di file e senza contenuto. Questo ordina al Raspberry Pi di abilitare le comunicazioni SSH al primo avvio.
5. Estrarre la scheda microSD e rimuoverla in modo sicuro dal computer host locale.

La tua scheda microSD è pronta a the section called “Avvia il tuo dispositivo IoT”.

Avvia il tuo dispositivo IoT con il nuovo sistema operativo

Questa procedura installa la scheda microSD ed esegue il primo avvio del Raspberry Pi utilizzando il sistema operativo scaricato.

Per avviare il dispositivo IoT con il nuovo sistema operativo

1. Con l'alimentazione scollegata dal dispositivo, inserire la scheda microSD dal passaggio precedente, the section called “Carica il sistema operativo”, nel Raspberry Pi.
2. Connetti il dispositivo alla rete cablata.
3. Questi tutorial interagiranno con il tuo Raspberry Pi dal tuo computer host locale utilizzando un terminale SSH.

   Se si desidera interagire direttamente con il dispositivo, è possibile:

   a. Connettere un monitor HDMI per guardare i messaggi della console di Raspberry Pi prima di poter connettere la finestra terminale del computer host locale al tuo Raspberry Pi.
   b. Connettere una tastiera USB ad essa se si desidera interagire direttamente con il Raspberry Pi.
4. Connettere il Raspberry Pi all'alimentazione e attendere circa un minuto prima che si inizializzi.

   Se hai un monitor connesso al tuo Raspberry Pi, puoi guardare il processo di avvio da qui.
5. Scopri l'indirizzo IP del tuo dispositivo:

   - Se è stato connesso un monitor HDMI a Raspberry Pi, l'indirizzo IP viene visualizzato nei messaggi visualizzati sul monitor
   - Se si ha accesso al router a cui si connette Raspberry Pi, è possibile visualizzarne l'indirizzo nell'interfaccia amministratore del router.

Una volta ottenuto l'indirizzo IP di Raspberry Pi, sei pronto per the section called “Connect il computer host”.

Collega il dispositivo al computer host locale

Questa procedura utilizza il programma terminale sul computer host locale per connettersi al Raspberry Pi e modificarne la password predefinita.

Per connettere il computer host locale al dispositivo

1. Sul computer host locale, apri il programma del terminale SSH:

   - Windows: `PuTTY`
   - Sistema operativo Linux/mac: `Terminal`

   > **Note**
   >
   > PuTTY non è installato automaticamente su Windows. Se non è presente nel computer, potrebbe essere necessario scaricarlo e installarlo.

2. Connetti il programma terminale all'indirizzo IP di Raspberry Pi e accedi utilizzando le credenziali di default.

   ```
   username: pi
   password: raspberry
   ```

3. Dopo aver effettuato l'accesso al tuo Raspberry Pi, cambia la password per l'utente `pi`.

   ```
   passwd
   ```

   Segui le istruzioni per modificare la password.

   ```
   Changing password for pi.
   Current password: raspberry
   New password: YourNewPassword
   Retype new password: YourNewPassword
   passwd: password updated successfully
   ```

Dopo aver visualizzato il prompt della riga di comando di Raspberry Pi nella finestra terminale e aver modificato la password, sei pronto a proseguire con the section called “Installa e verifica il software richiesto”.

Installa e verifica il software richiesto sul tuo dispositivo

Le procedure in questa sezione continuano dalla sezione precedente per aggiornare il sistema operativo del Raspberry Pi e installare sul Raspberry Pi il software che verrà utilizzato nella sezione successiva per creare e installare il AWS IoT Device Client.

Dopo aver completato questa sezione, il tuo Raspberry Pi avrà un sistema up-to-date operativo, il software richiesto dai tutorial di questo percorso di apprendimento, e sarà configurato in base alla tua posizione.

Equipaggiamento necessario:

- Il tuo ambiente locale di sviluppo e di test dalla sezione precedente
- Il Raspberry Pi che hai utilizzato nella sezione precedente
- La scheda di memoria microSD della sezione precedente

> **Note**
>
> Raspberry Pi Model 3+ e Raspberry Pi Model 4 possono eseguire tutti i comandi descritti in questo percorso di apprendimento. Se il dispositivo IoT non è in grado di compilare o eseguire il software AWS Command Line Interface, potrebbe essere necessario installare i compilatori richiesti sul computer host locale per creare il software e quindi trasferirlo sul dispositivo IoT. Per ulteriori informazioni su come installare e costruire il software per il dispositivo, consulta la documentazione relativa al software del dispositivo.

Procedure in questa sezione:

- Aggiorna il software del sistema operativo
- Installare le applicazioni e le librerie richieste
- (Facoltativo) Salvare l'immagine della scheda microSD

Aggiorna il software del sistema operativo

Questa procedura aggiorna il software del sistema operativo.

Per aggiornare il software del sistema operativo su Raspberry Pi

Eseguire questi passaggi nella finestra terminale del computer host locale.

1. Inserisci questi comandi per aggiornare il software di sistema sul Raspberry Pi.

```
sudo apt-get -y update
sudo apt-get -y upgrade
sudo apt-get -y autoremove
```

2. Aggiornare le impostazioni locali e del fuso orario di Raspberry Pi (facoltativo).

   Inserisci questo comando per aggiornare le impostazioni locali e il fuso orario del dispositivo.

```
sudo raspi-config
```

   a. Per impostare le impostazioni locali del dispositivo:

      i. Nella schermata dello Raspberry Pi Software Configuration Tool (raspi-config) (Strumento di configurazione del software Raspberry Pi (raspi-config)), seleziona l'opzione 5.

         5 Localisation Options Configure language and regional settings

         Utilizzare il tasto Tab per passare a <Seleziona>, quindi fai clic su space bar.

      ii. Nel menu delle opzioni di localizzazione, seleziona l'opzione L1.

         L1 Locale Configure language and regional settings

         Utilizzare il tasto Tab per passare a <Seleziona>, quindi fai clic su space bar.

      iii. Nell'elenco delle opzioni locali, seleziona le impostazioni locali che desideri installare sul Raspberry Pi utilizzando i tasti freccia per scorrere e space bar per contrassegnare le opzioni desiderate.

         Negli Stati Uniti, **en_US.UTF-8** è una scelta suggerita.

      iv. Dopo aver selezionato le impostazioni locali per il tuo dispositivo, usa il tasto Tab per selezionare <OK>, quindi fai clic su space bar per visualizzare la pagina di conferma di Configuring locales (Configurazione di impostazioni locali).

   b. Per impostare il fuso orario del dispositivo:

      i. Nella schermata raspi-config, seleziona l'opzione 5.

         5 Localisation Options Configure language and regional settings

Utilizzare il tasto Tab per passare a <Seleziona>, quindi fai clic su space bar.

ii. Nel menu delle opzioni di localizzazione, utilizzare il tasto freccia per seleziona l'opzione L2:

`L2 time zone Configure time zone`

Utilizzare il tasto Tab per passare a <Seleziona>, quindi fai clic su space bar.

iii. Nel menu di Configuring tzdata (Configurazione di tzdata), seleziona la tua area geografica dall'elenco.

Utilizza la chiave Tab per passare a <OK>, quindi fai clic su space bar.

iv. Nell'elenco delle città, usa i tasti freccia per seleziona una città nel tuo fuso orario.

Per impostare il fuso orario, utilizza il tasto Tab per passare a <OK>, quindi fai clic su space bar.

c. Quando hai finito di aggiornare le impostazioni, utilizza il tasto Tab per passare a <Termina>, quindi premere space bar per chiudere l'app raspi-config.

3. Inserisci questo comando per riavviare il Raspberry Pi.

```
sudo shutdown -r 0
```

4. Attendi il riavvio del Raspberry Pi.
5. Dopo il riavvio di Raspberry Pi, riconnetti la finestra terminale sul computer host locale al Raspberry Pi.

Ora il software di sistema Raspberry Pi è configurato e sei pronto a proseguire con the section called “Installa applicazioni e librerie”.

Installare le applicazioni e le librerie richieste

Questa procedura installa il software e le librerie dell'applicazione utilizzati dai tutorial successivi.

Se si utilizza un Raspberry Pi o se è possibile compilare il software richiesto sul dispositivo IoT, eseguire questi passaggi nella finestra terminale del computer host locale. Se è necessario compilare software per il dispositivo IoT sul computer host locale, consultare la documentazione software del dispositivo IoT per informazioni su come eseguire questi passaggi sul dispositivo.

Per installare il software e le librerie dell'applicazione sul Raspberry Pi

1. Inserisci questo comando per installare il software e le librerie dell'applicazione.

```
sudo apt-get -y install build-essential libssl-dev cmake unzip git python3-pip
```

2. Inserisci questi comandi per confermare che è stata installata la versione corretta del software.

```
gcc --version
cmake --version
openssl version
git --version
```

3. Verificare che siano installate queste versioni del software applicativo:

   - `gcc`: 9.3.0 o versioni successive
   - `cmake`: 3.10.x o versioni successive
   - `OpenSSL`: 1.1.1 o versioni successive
   - `git`: 2.20.1 o versioni successive

Se il tuo Raspberry Pi ha versioni accettabili del software dell'applicazione richiesto, sei pronto a proseguire con the section called “(Opzionale) Salva immagine microSD”.

(Facoltativo) Salvare l'immagine della scheda microSD

Durante i tutorial di questo percorso di apprendimento, verranno eseguite queste procedure per salvare una copia dell'immagine della scheda microSD di Raspberry Pi in un file sul computer host locale. Sebbene siano attività suggerite, non sono obbligatorie. Salvando l'immagine della scheda microSD nel percorso suggerito, è possibile saltare le procedure che precedono il punto di salvataggio in questo percorso di apprendimento; ciò consente di risparmiare tempo nel caso in cui fosse necessario ripetere uno dei passaggi. Se l'immagine della scheda microSD non viene salvata periodicamente, potrebbe essere necessario ricominciare dall'inizio i tutorial nel percorso di apprendimento se la scheda microSD è danneggiata o se si configura accidentalmente un'app o le relative impostazioni in modo errato.

A questo punto, la scheda microSD di Raspberry Pi ha un sistema operativo aggiornato e il software dell'applicazione di base caricato. Puoi riguadagnare il tempo che hai impiegato per completare i passaggi precedenti salvando ora il contenuto della scheda microSD in un file. Avere l'immagine corrente dell'immagine della scheda microSD del dispositivo consente di iniziare da questo punto

per continuare o riprovare un tutorial o una procedura senza la necessità di installare e aggiornare il software da zero.

Per salvare l'immagine della scheda microSD in un file

1. Inserisci questo comando per chiudere il Raspberry Pi.

   ```
   sudo shutdown -h 0
   ```

2. Dopo che il Raspberry Pi si è spento completamente, scollega l'alimentatore.
3. Rimuovi la scheda microSD dal Raspberry Pi.
4. Sul computer host locale:

   a. Inserisci la scheda microSD.

   b. Utilizzando lo strumento di creazione di immagini della scheda SD, salva l'immagine della scheda microSD in un file.

   c. Dopo aver salvato l'immagine della scheda microSD, espelli la scheda dal computer host locale.
5. Con l'alimentazione scollegata dal Raspberry Pi, inserisci la scheda microSD nel Raspberry Pi.
6. Collega Raspberry Pi all'alimentazione.
7. Dopo aver atteso circa un minuto, sul computer host locale, riconnettere la finestra terminale sul computer host locale connesso al Raspberry Pi, quindi accedi a Raspberry Pi.

Testa il tuo dispositivo e salva il certificato Amazon CA

Le procedure descritte in questa sezione riprendono quelle della sezione precedente per l'installazione AWS Command Line Interface e il certificato dell'Autorità di certificazione utilizzato per l'autenticazione delle connessioni. AWS IoT Core

Dopo aver completato questa sezione, saprai che il tuo Raspberry Pi dispone del software di sistema necessario per installare il AWS IoT Device Client e che dispone di una connessione Internet funzionante.

Equipaggiamento necessario:

- Il tuo ambiente locale di sviluppo e di test dalla sezione precedente
- Il Raspberry Pi che hai utilizzato nella sezione precedente

- La scheda di memoria microSD della sezione precedente

Procedure in questa sezione:

- Installa il AWS Command Line Interface
- Configura le tue Account AWS credenziali
- Esegui il download del certificato Amazon Root CA
- (Facoltativo) Salvare l'immagine della scheda microSD

Installa il AWS Command Line Interface

Questa procedura lo installa AWS CLI sul tuo Raspberry Pi.

Se si utilizza un Raspberry Pi o se è possibile compilare software sul dispositivo IoT, esegui questi passaggi nella finestra terminale del computer host locale. Se è necessario compilare software per il dispositivo IoT sul computer host locale, consulta la documentazione software del dispositivo IoT per informazioni sulle librerie necessarie.

Per installarlo AWS CLI sul tuo Raspberry Pi

1. Eseguire i seguenti comandi per scaricare e installare AWS CLI.

```
export PATH=$PATH:~/.local/bin # configures the path to include the directory with
 the AWS CLI
git clone https://github.com/aws/aws-cli.git # download the AWS CLI code from
 GitHub
cd aws-cli && git checkout v2 # go to the directory with the repo and checkout
 version 2
pip3 install -r requirements.txt # install the prerequisite software
```

2. Esegui questo comando per installare il AWS CLI. Il completamento del processo può richiedere fino a 15 minuti.

```
pip3 install . # install the AWS CLI
```

3. Esegui questo comando per confermare che AWS CLI è stata installata la versione corretta di.

```
aws --version
```

   La versione di AWS CLI dovrebbe essere 2.2 o successiva.

Se è AWS CLI visualizzata la versione corrente, sei pronto per continuarethe section called "Configura le credenziali dell'account".

Configura le tue Account AWS credenziali

In questa procedura, otterrai Account AWS le credenziali e le aggiungerai per utilizzarle sul tuo Raspberry Pi.

Per aggiungere le Account AWS credenziali al dispositivo

1. Ottieni un ID chiave di accesso e una chiave di accesso segreta da te Account AWS per autenticarli AWS CLI sul tuo dispositivo.

   Se non conosci AWS IAM, https://aws.amazon.com/premiumsupport/knowledge-center/create-access-key/descrive il processo da eseguire nella AWS console per creare credenziali AWS IAM da utilizzare sul tuo dispositivo.
2. Nella finestra terminale del computer host locale connesso al tuo Raspberry Pi. e con le credenziali ID chiave di accesso e Chiave di accesso segreta per il tuo dispositivo:

   a. Esegui l'app di configurazione con questo comando: AWS

   ```
   aws configure
   ```

   b. Inserisci le credenziali e le informazioni di configurazione quando richiesto:

   ```
   AWS Access Key ID: your Access Key ID
   AWS Secret Access Key: your Secret Access Key
   Default region name: your Regione AWS code
   Default output format: json
   ```
3. Esegui questo comando per testare l'accesso del tuo dispositivo al tuo Account AWS AWS IoT Core endpoint.

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

   Dovrebbe restituire il tuo endpoint Account AWS di AWS IoT dati specifico, come questo esempio:

   ```
   {
       "endpointAddress": "a3EXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   ```

```
}
```

Se vedi il tuo endpoint di AWS IoT dati Account AWS specifico, il tuo Raspberry Pi dispone della connettività e delle autorizzazioni per continuare a farlo. the section called “Esegui il download del certificato Amazon Root CA”

> **Important**
>
> Le tue Account AWS credenziali sono ora memorizzate sulla scheda microSD del tuo Raspberry Pi. Oltre a AWS semplificare le interazioni future con il software che creerai in questi tutorial, queste verranno anche salvate e duplicate in tutte le immagini della scheda microSD che creerai dopo questo passaggio per impostazione predefinita.
> Per proteggere la sicurezza delle tue Account AWS credenziali, prima di salvare altre immagini di schede microSD, valuta la possibilità di cancellare le credenziali `aws configure` eseguendo nuovamente l'operazione e inserendo caratteri casuali per l'ID della chiave di accesso e la chiave di accesso segreta per evitare che le tue credenziali vengano compromesse. Account AWS
> Se scopri di aver salvato le tue Account AWS credenziali inavvertitamente, puoi disattivarle nella console IAM. AWS

Esegui il download del certificato Amazon Root CA

Questa procedura scarica e salva una copia di un certificato Amazon Root Certification Authority (CA). Il download di questo certificato lo salva per l'uso nei tutorial successivi e verifica anche la connettività del dispositivo con i servizi AWS .

Per scaricare e salvare il certificato Amazon Root CA

1. Esegui il seguente comando per creare una directory per il certificato.

   ```
   mkdir ~/certs
   ```

2. Esegui questo comando per scaricare il certificato Amazon Root CA.

   ```
   curl -o ~/certs/AmazonRootCA1.pem https://www.amazontrust.com/repository/
   AmazonRootCA1.pem
   ```

3. Esegui questi comandi per impostare l'accesso alla directory del certificato e al relativo file.

```
chmod 745 ~
chmod 700 ~/certs
chmod 644 ~/certs/AmazonRootCA1.pem
```

4. Esegui questo comando per visualizzare il file del certificato emesso da una CA nella nuova directory.

   ```
   ls -l ~/certs
   ```

   Dovresti vedere una voce come questa. La data e l'ora saranno diverse; tuttavia, le dimensioni del file e tutte le altre informazioni dovrebbero essere le stesse mostrate qui.

   ```
   -rw-r--r-- 1 pi pi 1188 Oct 28 13:02 AmazonRootCA1.pem
   ```

   Se la dimensione del file non è 1188, controllare i parametri di comando curl. Potresti aver scaricato un file errato.

(Facoltativo) Salvare l'immagine della scheda microSD

A questo punto, la scheda microSD di Raspberry Pi ha un sistema operativo aggiornato e il software dell'applicazione di base caricato.

Per salvare l'immagine della scheda microSD in un file

1. Nella finestra terminale del computer host locale, cancella le credenziali AWS .

   a. Esegui l'app di AWS configurazione con questo comando:

      ```
      aws configure
      ```

   b. Sostituisci le credenziali quando richiesto. Puoi lasciare il Nome della regione di default e il Formato di output di default così come sono facendo clic su Invio.

      ```
      AWS Access Key ID [****************YT2H]: XYXYXYXYX
      AWS Secret Access Key [****************9plH]: XYXYXYXYX
      Default region name [us-west-2]:
      Default output format [json]:
      ```

2. Inserisci questo comando per chiudere il Raspberry Pi.

```
sudo shutdown -h 0
```

3. Dopo che il Raspberry Pi si spegne completamente, rimuovere l'alimentatore.
4. Rimuovere la scheda microSD dal dispositivo.
5. Sul computer host locale:
   a. Inserisci la scheda microSD.
   b. Utilizzando lo strumento di creazione di immagini della scheda SD, salva l'immagine della scheda microSD in un file.
   c. Dopo aver salvato l'immagine della scheda microSD, espelli la scheda dal computer host locale.
6. Con l'alimentazione scollegata dal Raspberry Pi, inserisci la scheda microSD nel Raspberry Pi.
7. Applica l'alimentazione al dispositivo.
8. Dopo circa un minuto, sul computer host locale, riavvia la sessione della finestra terminale e accedi al dispositivo.

   Non reinserire ancora Account AWS le tue credenziali.

Dopo aver riavviato e effettuato l'accesso al tuo Raspberry Pi, sei pronto a passare a the section called “Installazione e configurazione del client per dispositivi IoT”.

Tutorial: installazione e configurazione del AWS IoT Device Client

Questo tutorial ti guida attraverso l'installazione e la configurazione del AWS IoT Device Client e la creazione di AWS IoT risorse che utilizzerai in questa e in altre demo.

Per iniziare il tutorial:

- Prepara il tuo computer host locale e Raspberry Pi seguendo il il tutorial precedente.

Questo tutorial dura circa 90 minuti.

Al termine di questo argomento:

- Il tuo dispositivo IoT sarà pronto per l'uso in altre demo di AWS IoT Device Client.
- Avrai effettuato il provisioning del tuo dispositivo IoT. AWS IoT Core

- Avrai scaricato e installato il AWS IoT Device Client sul tuo dispositivo.
- Avrai salvato un'immagine della scheda microSD del tuo dispositivo che può essere utilizzata nei tutorial successivi.

Equipaggiamento necessario:

- Il tuo ambiente locale di sviluppo e di test dalla sezione precedente
- Il Raspberry Pi che hai utilizzato nella sezione precedente
- La scheda di memoria microSD del Raspberry Pi utilizzata nella sezione precedente

Procedure all'interno del tutorial

- Scarica e salva il AWS IoT Device Client
- Esegui il provisioning del tuo Raspberry Pi in AWS IoT
- Configura il AWS IoT Device Client per testare la connettività

Scarica e salva il AWS IoT Device Client

Le procedure descritte in questa sezione scaricano il AWS IoT Device Client, lo compilano e lo installano sul tuo Raspberry Pi. Dopo aver testato l'installazione, è possibile salvare l'immagine della scheda microSD di Raspberry Pi da utilizzare in seguito quando si desidera riprovare i tutorial.

Procedure in questa sezione:

- Scarica e crea il Device Client AWS IoT
- Crea le directory utilizzate dal tutorial
- (Facoltativo) Salvare l'immagine della scheda microSD

Scarica e crea il Device Client AWS IoT

Questa procedura installa il AWS IoT Device Client sul tuo Raspberry Pi.

Esegui questi comandi nella finestra terminale del computer host locale connesso al Raspberry Pi.

Per installare il AWS IoT Device Client sul tuo Raspberry Pi

1. Inserisci questi comandi per scaricare e creare il AWS IoT Device Client sul tuo Raspberry Pi.

```
cd ~
git clone https://github.com/awslabs/aws-iot-device-client aws-iot-device-client
mkdir ~/aws-iot-device-client/build && cd ~/aws-iot-device-client/build
cmake ../
```

2. Esegui questo comando per creare il AWS IoT Device Client. Il completamento del processo può richiedere fino a 15 minuti.

   ```
   cmake --build . --target aws-iot-device-client
   ```

   I messaggi di avviso visualizzati durante la compilazione del AWS IoT Device Client possono essere ignorati.

   Questi tutorial sono stati testati con il AWS IoT Device Client integratogcc, versione (Raspbian 10.2.1-6+rpi1) 10.2.1 20210110 sulla versione del sistema operativo Raspberry Pi (bullseye) del 30 ottobre 2021gcc, versione (Raspbian 8.3.0-6+rpi1) 8.3.0 sulla versione del sistema operativo Raspberry Pi (buster) del 7 maggio 2021.

3. Dopo che il AWS IoT Device Client ha terminato la creazione, testalo eseguendo questo comando.

   ```
   ./aws-iot-device-client --help
   ```

Se vedi la guida della riga di comando per AWS IoT Device Client, significa che il AWS IoT Device Client è stato creato correttamente ed è pronto per l'uso.

Crea le directory utilizzate dal tutorial

Questa procedura crea le directory sul Raspberry Pi che verranno utilizzate per memorizzare i file utilizzati dai tutorial in questo percorso di apprendimento.

Per creare le directory utilizzate dai tutorial in questo percorso di apprendimento:

1. Esegui questi comandi per creare le directory richieste.

   ```
   mkdir ~/dc-configs
   mkdir ~/policies
   mkdir ~/messages
   mkdir ~/certs/testconn
   ```

```
mkdir ~/certs/pubsub
mkdir ~/certs/jobs
```

2. Esegui questi comandi per impostare le autorizzazioni sulle nuove directory.

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 700 ~/certs/pubsub
chmod 700 ~/certs/jobs
```

Dopo aver creato queste directory e impostato l'autorizzazione, prosegui con the section called “(Facoltativo) Salvare l'immagine della scheda microSD”.

(Facoltativo) Salvare l'immagine della scheda microSD

A questo punto, la scheda microSD del Raspberry Pi ha un sistema operativo aggiornato, il software applicativo di base e il Device Client. AWS IoT

Se vuoi tornare a provare nuovamente questi esercizi e tutorial, puoi saltare le procedure precedenti scrivendo l'immagine della scheda microSD salvata con questa procedura su una nuova scheda microSD e continuare i tutorial da the section called “Fornisci il tuo Raspberry Pi”.

Per salvare l'immagine della scheda microSD in un file:

Nella finestra terminale del computer host locale connesso al Raspberry Pi:

1. Conferma che le tue Account AWS credenziali non sono state archiviate.

   a. Esegui l'app di AWS configurazione con questo comando:

   ```
   aws configure
   ```

   b. Se le credenziali sono state archiviate (se sono visualizzate nel prompt), inserisci la stringa **XYXYXYXYX** quando viene richiesto come illustrato qui. Lascia vuoti i campi Nome della regione di default e Formato di output di default.

   ```
   AWS Access Key ID [****************YXYX]: XYXYXYXYX
   AWS Secret Access Key [****************YXYX]: XYXYXYXYX
   Default region name:
   Default output format:
   ```

2. Inserisci questo comando per arrestare il Raspberry Pi.

```
sudo shutdown -h 0
```

3. Dopo che il Raspberry Pi si spegne completamente, rimuovere l'alimentatore.
4. Rimuovere la scheda microSD dal dispositivo.
5. Sul computer host locale:

   a. Inserisci la scheda microSD.

   b. Utilizzando lo strumento di creazione di immagini della scheda SD, salva l'immagine della scheda microSD in un file.

   c. Dopo aver salvato l'immagine della scheda microSD, espelli la scheda dal computer host locale.

Puoi continuare con questa scheda microSD in the section called “Fornisci il tuo Raspberry Pi”.

Esegui il provisioning del tuo Raspberry Pi in AWS IoT

Le procedure in questa sezione iniziano con l'immagine microSD salvata su cui è installato AWS CLI il Device Client AWS IoT e creano le risorse e i certificati del dispositivo che forniscono AWS IoT il tuo Raspberry Pi. AWS IoT

Installa la scheda microSD nel Raspberry Pi

Questa procedura installa la scheda microSD con il software necessario caricato e configurato nel Raspberry Pi e la configura in modo da poter continuare con Account AWS i tutorial di questo percorso di apprendimento.

Utilizzare una scheda microSD da the section called “(Facoltativo) Salvare l'immagine della scheda microSD” che dispone del software necessario per gli esercizi e le esercitazioni in questo percorso di apprendimento.

Per installare la scheda microSD nel Raspberry Pi

1. Con l'alimentazione scollegata dal Raspberry Pi, inserisci la scheda microSD nel Raspberry Pi.
2. Collega Raspberry Pi all'alimentazione.
3. Dopo circa un minuto, sul computer host locale, riavvia la sessione della finestra terminale e accedi al Raspberry Pi.

4. Sul computer host locale, nella finestra terminale e con le credenziali ID chiave di accesso e Chiave di accesso segreta per il Raspberry Pi:

   a. Esegui l'app di configurazione con questo comando: AWS

   ```
   aws configure
   ```

   b. Inserisci Account AWS le tue credenziali e le informazioni di configurazione quando richiesto:

   ```
   AWS Access Key ID [****************YXYX]: your Access Key ID
   AWS Secret Access Key [****************YXYX]: your Secret Access Key
   Default region name [us-west-2]: your Regione AWS code
   Default output format [json]: json
   ```

Dopo aver ripristinato Account AWS le credenziali, sei pronto per continuare. the section called “Effettua il provisioning del dispositivo in AWS IoT Core”

Effettua il provisioning del dispositivo in AWS IoT Core

Le procedure descritte in questa sezione creano AWS IoT le risorse per il rifornimento del tuo Raspberry Pi. AWS IoT Man mano che crei queste risorse, ti verrà chiesto di registrare varie informazioni. Queste informazioni vengono utilizzate dalla configurazione del AWS IoT Device Client nella procedura successiva.

Affinché il tuo Raspberry Pi funzioni AWS IoT, deve essere fornito. Il provisioning è il processo di creazione e configurazione AWS IoT delle risorse necessarie per supportare il tuo Raspberry Pi come dispositivo IoT.

Con il Raspberry Pi acceso e riavviato, connetti la finestra terminale del computer host locale al Raspberry Pi e completa queste procedure.

Procedure in questa sezione:

- Creare e scaricare i file di certificato del dispositivo
- Crea risorse AWS IoT

Creare e scaricare i file di certificato del dispositivo

Questa procedura crea i file di certificato del dispositivo per questa demo.

Per creare e scaricare i file di certificato del dispositivo per il tuo Raspberry Pi

1. Nella finestra terminale del computer host locale, immettere questi comandi per creare i file di certificato del dispositivo per il dispositivo.

```
mkdir ~/certs/testconn
aws iot create-keys-and-certificate \
--set-as-active \
--certificate-pem-outfile "~/certs/testconn/device.pem.crt" \
--public-key-outfile "~/certs/testconn/public.pem.key" \
--private-key-outfile "~/certs/testconn/private.pem.key"
```

   Questo comando restituisce una risposta simile alla seguente. Registra il valore *certificateArn* per utilizzarlo in seguito.

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
    "certificateId":
 "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
    "certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

2. Inserisci i seguenti comandi per impostare le autorizzazioni sulla directory dei certificati e sui relativi file.

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 644 ~/certs/testconn/*
chmod 600 ~/certs/testconn/private.pem.key
```

3. Esegui questo comando per rivedere le autorizzazioni sulle directory e sui file dei certificati.

```
ls -l ~/certs/testconn
```

L'output del comando dovrebbe essere lo stesso di quello che vedi qui, tranne per le date e gli orari del file.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

A questo punto, hai i file del certificato del dispositivo installati sul tuo Raspberry Pi e puoi proseguire con the section called “Crea risorse AWS IoT”.

Crea risorse AWS IoT

Questa procedura fornisce il dispositivo AWS IoT creando le risorse di cui ha bisogno per accedere a AWS IoT funzionalità e servizi.

Per effettuare il provisioning del dispositivo in AWS IoT

1. Nella finestra terminale del computer host locale, immettere il seguente comando per ottenere l'indirizzo dell'endpoint dati del dispositivo per l' Account AWS.

   ```
   aws iot describe-endpoint --endpoint-type IoT:Data-ATS
   ```

   Questo comando restituisce una risposta simile alla seguente. Registra il valore *endpointAddress* per utilizzarlo in seguito.

   ```
   {
       "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Inserisci questo comando per creare una AWS IoT risorsa per il tuo Raspberry Pi.

   ```
   aws iot create-thing --thing-name "DevCliTestThing"
   ```

   Se la tua risorsa AWS IoT thing è stata creata, il comando restituisce una risposta come questa.

   ```
   {
   ```

```
    "thingName": "DevCliTestThing",
    "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/DevCliTestThing",
    "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. Nella finestra terminale:

   a. Apri un editor di testo, ad esempio `nano`.

   b. Copia questo documento di policy JSON e incollalo nell'editor di testo aperto.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

> **Note**
>
> Questo documento di policy concede generosamente a tutte le risorse l'autorizzazione a connettersi, ricevere, pubblicare e sottoscrivere. Normalmente le policy consentono solo a risorse specifiche di eseguire azioni specifiche. Tuttavia, per il test iniziale della connettività del dispositivo, questa policy eccessivamente generale e permissiva viene utilizzata per ridurre al minimo la possibilità di un problema di accesso durante questo test. Nei tutorial successivi, verranno utilizzati documenti di policy più ristretti per dimostrare le migliori pratiche nella progettazione delle policy.

c. Salva il file nell'editor di testo come **~/policies/dev_cli_test_thing_policy.json**.

4. Esegui questo comando per utilizzare il documento di policy dei passaggi precedenti per creare una AWS IoT politica.

```
aws iot create-policy \
--policy-name "DevCliTestThingPolicy" \
--policy-document "file://~/policies/dev_cli_test_thing_policy.json"
```

Se la policy viene creata, verrà visualizzata una risposta simile alla seguente.

```
{
    "policyName": "DevCliTestThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/DevCliTestThingPolicy",
    "policyDocument": "{\n    \"Version\": \"2012-10-17\",       \n    \"Statement
\": [\n        {\n            \"Effect\": \"Allow\",\n            \"Action\": [\n
                \"iot:Publish\",\n                \"iot:Subscribe\",\n
    \"iot:Receive\",\n                \"iot:Connect\"\n            ],\n
 \"Resource\": [\n                \"*\"\n            ]\n        }\n    ]\n}\n",
    "policyVersionId": "1"
}
```

5. Eseguire questo comando per attribuire la policy al certificato del dispositivo. Sostituisci *certificateArn* con il valore `certificateArn` salvato in precedenza.

```
aws iot attach-policy \
--policy-name "DevCliTestThingPolicy" \
--target "certificateArn"
```

In caso di successo, questo comando non restituisce alcun risultato.

6. Esegui questo comando per allegare il certificato del dispositivo alla risorsa AWS IoT oggetto. Sostituisci *certificateArn* con il valore `certificateArn` salvato in precedenza.

```
aws iot attach-thing-principal \
--thing-name "DevCliTestThing" \
--principal "certificateArn"
```

In caso di successo, questo comando non restituisce alcun risultato.

Dopo aver effettuato correttamente il provisioning del dispositivo AWS IoT, sei pronto per continuarethe section called “Configura Device Client e verifica la connettività”.

Configura il AWS IoT Device Client per testare la connettività

Le procedure in questa sezione configurano il AWS IoT Device Client per pubblicare un messaggio MQTT dal tuo Raspberry Pi.

Procedure in questa sezione:

- Creare il file di configurazione
- Aprire il client di test MQTT
- AWS IoT Esegui Device Client

Creare il file di configurazione

Questa procedura crea il file di configurazione per testare il Device Client. AWS IoT

Per creare il file di configurazione per testare il AWS IoT Device Client

- Nella finestra terminale del computer host locale connesso al Raspberry Pi:

  a. Inserisci questi comandi per creare una directory per i file di configurazione e impostare l'autorizzazione sulla directory:

  ```
  mkdir ~/dc-configs
  chmod 745 ~/dc-configs
  ```

  b. Apri un editor di testo, ad esempio `nano`.

  c. Copia questo documento JSON e incollalo nell'editor di testo aperto.

  ```
  {
    "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
    "cert": "~/certs/testconn/device.pem.crt",
    "key": "~/certs/testconn/private.pem.key",
    "root-ca": "~/certs/AmazonRootCA1.pem",
    "thing-name": "DevCliTestThing",
    "logging": {
      "enable-sdk-logging": true,
      "level": "DEBUG",
      "type": "STDOUT",
  ```

```
    "file": ""
  },
  "jobs": {
    "enabled": false,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": ""
    }
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

d. Sostituisci il *endpoint* valore con Device Data Endpoint per il tuo dispositivo Account AWS che hai trovato in. the section called “Effettua il provisioning del dispositivo in AWS IoT Core”

e. Salva il file nell'editor di testo come **~/dc-configs/dc-testconn-config.json**.

f. Esegui questo comando per impostare le autorizzazioni sul nuovo file di configurazione.

```
chmod 644 ~/dc-configs/dc-testconn-config.json
```

Dopo aver salvato il file, è possibile proseguire con the section called “Aprire il client di test MQTT”.

Aprire il client di test MQTT

Questa procedura prepara il client di test MQTT nella AWS IoT console per sottoscrivere il messaggio MQTT che il AWS IoT Device Client pubblica durante l'esecuzione.

Per preparare il Client MQTT di test a sottoscrivere tutti i messaggi MQTT

1. Sul computer host locale, nella console AWS IoT, seleziona Client MQTT di test.
2. Nella scheda Subscribe to a topic (Sottoscrivi un argomento), in Topic filter (Filtro argomenti) inserisci **#** (un solo cancelletto) e seleziona Subscribe (Sottoscrivi) per sottoscrivere ogni argomento MQTT.
3. Sotto l'etichetta Subscriptions (Sottoscrizioni), conferma di vedere **#** (un solo segno di cancelletto).

Lascia aperta la finestra con il Client MQTT di test mentre prosegui con the section called “AWS IoT Esegui Device Client”.

AWS IoT Esegui Device Client

Questa procedura esegue il AWS IoT Device Client in modo da pubblicare un singolo messaggio MQTT che il client di test MQTT riceve e visualizza.

Per inviare un messaggio MQTT dal Device Client AWS IoT

1. Assicurati che sia la finestra terminale collegata al tuo Raspberry Pi che la finestra con il client MQTT di test siano visibili durante l'esecuzione di questa procedura.
2. Nella finestra del terminale, inserisci questi comandi per eseguire il AWS IoT Device Client utilizzando il file di configurazione creato in. the section called “Creare il file di configurazione”

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-testconn-config.json
```

Nella finestra del terminale, il AWS IoT Device Client visualizza i messaggi informativi e gli eventuali errori che si verificano durante l'esecuzione.

Se nella finestra del terminale non vengono visualizzati errori, esamina il client MQTT di test.

3. Nel client MQTT di test, nella finestra Subsciptions (Sottoscrizioni), viene visualizzato il messaggio Hello World! inviato all' argomento del messaggio `test/dc/pubtopic`.
4. Se il AWS IoT Device Client non mostra errori e viene visualizzato Hello World! inviato al `test/dc/pubtopic` messaggio nel client di test MQTT, hai dimostrato che la connessione è riuscita.
5. Nella finestra del terminale, digitate **^C** (Ctrl-C) per fermare il AWS IoT Device Client.

Dopo aver dimostrato che il AWS IoT Device Client funziona correttamente sul tuo Raspberry Pi e con cui puoi comunicare AWS IoT, puoi continuare con. the section called “Comunica con il client del dispositivo utilizzando MQTT”

Tutorial: Dimostrare la comunicazione dei messaggi MQTT con il AWS IoT Device Client

Questo tutorial dimostra come il AWS IoT Device Client può sottoscrivere e pubblicare messaggi MQTT, che sono comunemente usati nelle soluzioni IoT.

Per iniziare il tutorial:

- Configurare il computer host locale e Raspberry Pi configurati come nella la sezione precedente.

  Se hai salvato l'immagine della scheda microSD dopo aver installato AWS IoT il Device Client, puoi utilizzare una scheda microSD con quell'immagine con il tuo Raspberry Pi.
- Se hai già eseguito questa demo, prova ??? a eliminare tutte le AWS IoT risorse create nelle esecuzioni precedenti per evitare errori di risorse duplicate.

Questo tutorial dura circa 45 minuti.

Al termine di questo argomento:

- Avrai dimostrato diversi modi in cui il tuo dispositivo IoT può iscriversi ai messaggi MQTT AWS IoT e pubblicare messaggi MQTT su. AWS IoT

Equipaggiamento necessario:

- Il tuo ambiente locale di sviluppo e di test dalla sezione precedente
- Il Raspberry Pi che hai utilizzato nella sezione precedente
- La scheda di memoria microSD del Raspberry Pi utilizzata nella sezione precedente

Procedure all'interno del tutorial

- Prepara il Raspberry Pi per dimostrare la comunicazione tramite messaggi MQTT
- Dimostra la pubblicazione di messaggi con AWS IoT Device Client
- Dimostra l'iscrizione ai messaggi con Device Client AWS IoT

Prepara il Raspberry Pi per dimostrare la comunicazione tramite messaggi MQTT

Questa procedura crea le risorse in AWS IoT e nel Raspberry Pi per dimostrare la comunicazione dei messaggi MQTT utilizzando il AWS IoT Device Client.

Procedure in questa sezione:

- Creare i file di certificato per dimostrare la comunicazione MQTT
- Effettua il provisioning del dispositivo per dimostrare la comunicazione MQTT
- Configura il file di configurazione del AWS IoT Device Client e il client di test MQTT per dimostrare la comunicazione MQTT

Creare i file di certificato per dimostrare la comunicazione MQTT

Questa procedura crea i file di certificato del dispositivo per questa demo.

Per creare e scaricare i file di certificato del dispositivo per il tuo Raspberry Pi

1. Nella finestra terminale del computer host locale, immettere questi comandi per creare i file di certificato del dispositivo per il dispositivo.

```
mkdir ~/certs/pubsub
aws iot create-keys-and-certificate \
--set-as-active \
--certificate-pem-outfile "~/certs/pubsub/device.pem.crt" \
--public-key-outfile "~/certs/pubsub/public.pem.key" \
```

```
--private-key-outfile "~/certs/pubsub/private.pem.key"
```

Questo comando restituisce una risposta simile alla seguente. Salva il *certificateArn* per utilizzarlo in un secondo momento.

```
{
"certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificateId":
 "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
\n",
"keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
}
}
```

2. Inserisci i seguenti comandi per impostare le autorizzazioni sulla directory dei certificati e sui relativi file.

```
chmod 700 ~/certs/pubsub
chmod 644 ~/certs/pubsub/*
chmod 600 ~/certs/pubsub/private.pem.key
```

3. Esegui questo comando per rivedere le autorizzazioni sulle directory e sui file dei certificati.

```
ls -l ~/certs/pubsub
```

L'output del comando dovrebbe essere lo stesso di quello che vedi qui, tranne per le date e gli orari del file.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

4. Inserisci questi comandi per creare le directory per i file di log.

```
mkdir ~/.aws-iot-device-client
mkdir ~/.aws-iot-device-client/log
chmod 745 ~/.aws-iot-device-client/log
echo " " > ~/.aws-iot-device-client/log/aws-iot-device-client.log
echo " " > ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
chmod 600 ~/.aws-iot-device-client/log/*
```

Effettua il provisioning del dispositivo per dimostrare la comunicazione MQTT

Questa sezione crea le AWS IoT risorse che forniscono il tuo Raspberry Pi. AWS IoT

Per effettuare il provisioning del dispositivo in AWS IoT:

1. Nella finestra terminale del computer host locale, immettere il seguente comando per ottenere l'indirizzo dell'endpoint dati del dispositivo per l' Account AWS.

   ```
   aws iot describe-endpoint --endpoint-type IoT:Data-ATS
   ```

   Il valore dell'endpoint non è cambiato dal momento in cui hai eseguito questo comando per il tutorial precedente. Il comando viene eseguito nuovamente per trovare e incollare facilmente il valore dell'endpoint dei dati nel file di configurazione utilizzato in questo tutorial.

   Questo comando restituisce una risposta simile alla seguente. Registra il valore *endpointAddress* per utilizzarlo in seguito.

   ```
   {
   "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Inserisci questo comando per creare una nuova AWS IoT risorsa per il tuo Raspberry Pi.

   ```
   aws iot create-thing --thing-name "PubSubTestThing"
   ```

   Poiché una risorsa AWS IoT oggetto è una rappresentazione virtuale del tuo dispositivo nel cloud, possiamo creare più risorse di oggetti AWS IoT da utilizzare per scopi diversi. Possono essere utilizzati tutti dallo stesso dispositivo IoT fisico per rappresentare diversi aspetti del dispositivo.

Questi tutorial utilizzeranno solo una risorsa alla volta per rappresentare il Raspberry Pi. In questo modo, in questi tutorial, rappresentano le diverse demo in modo che, dopo aver creato le AWS IoT risorse per una demo, sia possibile tornare indietro e ripetere la demo utilizzando le risorse create appositamente per ciascuna di esse.

Se la risorsa AWS IoT oggetto è stata creata, il comando restituisce una risposta come questa.

```
{
"thingName": "PubSubTestThing",
"thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/PubSubTestThing",
"thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. Nella finestra terminale:

   a. Apri un editor di testo, ad esempio `nano`.

   b. Copia questo documento JSON e incollalo nell'editor di testo aperto.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:client/PubSubTestThing"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/pubtopic"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topicfilter/test/dc/
subtopic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/subtopic"
            ]
        }
    ]
}
```

c. Nell'editor, in ogni `Resource` sezione del documento di policy, sostituisci *us-west-2:57EXAMPLE833* con i tuoi Regione AWS, un carattere con i due punti (:) e il tuo numero di 12 cifre Account AWS .

d. Salva il file nell'editor di testo come **~/policies/pubsub_test_thing_policy.json**.

4. Esegui questo comando per utilizzare il documento di policy dei passaggi precedenti per creare una AWS IoT politica.

```
aws iot create-policy \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_test_thing_policy.json"
```

Se la policy viene creata, verrà visualizzata una risposta simile alla seguente.

```
{
                                    "policyName": "PubSubTestThingPolicy",
                                    "policyArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:policy/PubSubTestThingPolicy",
                                    "policyDocument": "{\n\"Version\":
 \"2012-10-17\",       \n\"Statement\": [\n{\n\"Effect\": \"Allow\",\n
\"Action\": [\n\"iot:Connect\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n},\n{\n\"Effect\": \"Allow
```

```
\",\n\"Action\": [\n\"iot:Publish\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Subscribe\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topicfilter/test/dc/subtopic\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Receive\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/*\"\n]\n}\n]\n}\n",
                                    "policyVersionId": "1"
                                    }
```

5. Eseguire questo comando per attribuire la policy al certificato del dispositivo. Sostituisci *certificateArn* con il valore `certificateArn` salvato in precedenza in questa sezione.

   ```
   aws iot attach-policy \
   --policy-name "PubSubTestThingPolicy" \
   --target "certificateArn"
   ```

   In caso di successo, questo comando non restituisce alcun risultato.

6. Esegui questo comando per attribuire il certificato del dispositivo alle risorse dell'oggetto AWS IoT . Sostituisci *certificateArn* con il valore `certificateArn` salvato in precedenza in questa sezione.

   ```
   aws iot attach-thing-principal \
   --thing-name "PubSubTestThing" \
   --principal "certificateArn"
   ```

   In caso di successo, questo comando non restituisce alcun risultato.

Dopo aver effettuato correttamente il provisioning del dispositivo AWS IoT, sei pronto per continuarethe section called “Configura il file di configurazione Device Client e il client MQTT”.

Configura il file di configurazione del AWS IoT Device Client e il client di test MQTT per dimostrare la comunicazione MQTT

Questa procedura crea un file di configurazione per testare il Device Client. AWS IoT

Per creare il file di configurazione per testare il AWS IoT Device Client

1. Nella finestra terminale del computer host locale connesso al Raspberry Pi:

   a. Apri un editor di testo, ad esempio `nano`.

b. Copia questo documento JSON e incollalo nell'editor di testo aperto.

```
{
  "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
  "cert": "~/certs/pubsub/device.pem.crt",
  "key": "~/certs/pubsub/private.pem.key",
  "root-ca": "~/certs/AmazonRootCA1.pem",
  "thing-name": "PubSubTestThing",
  "logging": {
    "enable-sdk-logging": true,
    "level": "DEBUG",
    "type": "STDOUT",
    "file": ""
  },
  "jobs": {
    "enabled": false,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
    }
  },
  "config-shadow": {
    "enabled": false
```

```
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

c. Sostituisci il *endpoint* valore con Device Data Endpoint per il tuo dispositivo Account AWS che hai trovato in. the section called “Effettua il provisioning del dispositivo in AWS IoT Core”

d. Salva il file nell'editor di testo come **~/dc-configs/dc-pubsub-config.json**.

e. Esegui questo comando per impostare le autorizzazioni sul nuovo file di configurazione.

```
chmod 644 ~/dc-configs/dc-pubsub-config.json
```

2. Per preparare il Client MQTT di test a sottoscrivere tutti i messaggi MQTT:

a. Sul computer host locale, nella console AWS IoT, seleziona Client MQTT di test.

b. Nella tabella Subscribe to a topic (Sottoscrivi un argomento), in Topic filter (Filtro argomenti), inserisci **#** (un solo segno di cancelletto) e scegli Subscribe (Sottoscrivi).

c. Sotto l'etichetta Subscriptions (Sottoscrizioni), conferma di vedere **#** (un solo segno di cancelletto).

Lascia la finestra con il Client MQTT di test aperta mentre continui questo tutorial.

Dopo aver salvato il file e configurato il client MQTT di test, sei pronto a proseguire con the section called “Pubblica messaggi con IoT Device Client”.

Dimostra la pubblicazione di messaggi con AWS IoT Device Client

Le procedure in questa sezione mostrano come il AWS IoT Device Client può inviare messaggi MQTT predefiniti e personalizzati.

Queste istruzioni di policy nella policy creata nella fase precedente per questi esercizi consentono al Raspberry Pi di eseguire queste azioni:

- **iot:Connect**

Consente la connessione al client denominato`PubSubTestThing`, il Raspberry Pi su cui è in esecuzione il AWS IoT Device Client.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Connect"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing"
  ]
}
```

- **`iot:Publish`**

  Fornire al Raspberry Pi l'autorizzazione a pubblicare messaggi con un argomento MQTT di `test/dc/pubtopic`.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Publish"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic"
  ]
}
```

  L'azione `iot:Publish` consente di pubblicare agli argomenti MQTT elencati nell'array di Risorsa. Il contenuto di tali messaggi non è controllato dall'istruzione di policy.

Pubblica il messaggio predefinito utilizzando il AWS IoT Device Client

Questa procedura esegue il AWS IoT Device Client in modo da pubblicare un singolo messaggio MQTT predefinito che il client di test MQTT riceve e visualizza.

Per inviare il messaggio MQTT predefinito dal Device Client AWS IoT

1. Assicurati che sia la finestra terminale collegata al computer host locale sia connessa al tuo Raspberry Pi e che la finestra con il client MQTT di test sia visibile durante l'esecuzione di questa procedura.
2. Nella finestra del terminale, inserisci questi comandi per eseguire il AWS IoT Device Client utilizzando il file di configurazione creato in. the section called “Creare il file di configurazione”

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-config.json
   ```

   Nella finestra del terminale, il AWS IoT Device Client visualizza i messaggi informativi e gli eventuali errori che si verificano durante l'esecuzione.

   Se nella finestra del terminale non vengono visualizzati errori, esamina il client MQTT di test.
3. In MQTT test client (Client MQTT di test), nella finestra Subscriptions (Sottoscrizioni) viene visualizzato il messaggio Hello World! inviato all'argomento del messaggio `test/dc/pubtopic`.
4. Se il AWS IoT Device Client non mostra errori e viene visualizzato Hello World! inviato al `test/dc/pubtopic` messaggio nel client di test MQTT, hai dimostrato che la connessione è riuscita.
5. Nella finestra del terminale, digitate **^C** (Ctrl-C) per fermare il AWS IoT Device Client.

Dopo aver dimostrato che AWS IoT Device Client ha pubblicato il messaggio MQTT predefinito, puoi continuare con. the section called “Pubblica un messaggio MQTT personalizzato”

Pubblica un messaggio personalizzato utilizzando AWS IoT Device Client

Le procedure di questa sezione creano un messaggio MQTT personalizzato e quindi eseguono AWS IoT Device Client in modo che pubblichi il messaggio MQTT personalizzato una volta per essere ricevuto e visualizzato dal client MQTT di test.

Crea un messaggio MQTT personalizzato per il Device Client AWS IoT

Esegui questi passaggi nella finestra terminale del computer host locale connesso al Raspberry Pi.

Per creare un messaggio personalizzato da pubblicare sul AWS IoT Device Client

1. Nella finestra terminale, apri un editor di testo, ad esempio `nano`.

2. Nell'editor di testo, copia e incolla il documento JSON riportato di seguito. Questo sarà il payload del messaggio MQTT pubblicato da AWS IoT Device Client.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

3. Salva il contenuto dell'editor di testo come **~/messages/sample-ws-message.json**.
4. Inserisci il comando seguente per impostare le autorizzazioni del file di messaggio appena creato.

```
chmod 600 ~/messages/*
```

Per creare un file di configurazione per il AWS IoT Device Client da utilizzare per inviare il messaggio personalizzato

1. Nella finestra del terminale, in un editor di testo comenano, apri il file di configurazione esistente di AWS IoT Device Client:. **~/dc-configs/dc-pubsub-config.json**
2. Modifica l'oggetto `samples` in questo modo. Nessun'altra parte di questo file deve essere modificata.

```
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "~/messages/sample-ws-message.json",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

3. Salva il contenuto dell'editor di testo come **~/dc-configs/dc-pubsub-custom-config.json**.
4. Esegui questo comando per impostare le autorizzazioni sul nuovo file di configurazione.

```
chmod 644 ~/dc-configs/dc-pubsub-custom-config.json
```

Pubblica il messaggio MQTT personalizzato utilizzando Device Client AWS IoT

Questa modifica influisce solo sui contenuti del payload del messaggio MQTT, quindi la policy attuale continuerà a funzionare. Tuttavia, se l'argomento MQTT (come definito dal valore `publish-topic` in `~/dc-configs/dc-pubsub-custom-config.json`) è stato cambiato, anche l'istruzione di policy `iot::Publish` dovrebbe essere modificata per consentire a Raspberry Pi di pubblicare sul nuovo argomento MQTT.

Per inviare il messaggio MQTT dal Device Client AWS IoT

1. Assicurati che sia la finestra terminale che la finestra con il client MQTT di test siano visibili durante l'esecuzione di questa procedura. Inoltre, assicurati che il tuo client MQTT di test abbia una sottoscrizione ancora attiva per il filtro di argomenti #. Se non lo è, effettua nuovamente la sottoscrizione al filtro argomento#.
2. Nella finestra terminale, immettere questi comandi per eseguire AWS IoT Device Client utilizzando il file di configurazione creato in the section called “Creare il file di configurazione”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
   ```

   Nella finestra del terminale, il AWS IoT Device Client visualizza i messaggi informativi e gli eventuali errori che si verificano durante l'esecuzione.

   Se nella finestra del terminale non vengono visualizzati errori, esaminare il client di test MQTT.
3. Nel client MQTT di test, nella finestra Subscriptions (Sottoscrizioni), viene visualizzato il payload del messaggio personalizzato inviato all'argomento del messaggio `test/dc/pubtopic`.
4. Se il AWS IoT Device Client non mostra errori e vedi il payload di messaggi personalizzati che hai pubblicato nel `test/dc/pubtopic` messaggio nel client di test MQTT, hai pubblicato un messaggio personalizzato con successo.
5. Nella finestra del terminale, digitate **^C** (Ctrl-C) per arrestare il Device Client. AWS IoT

Dopo aver dimostrato che AWS IoT Device Client ha pubblicato un payload di messaggi personalizzato, puoi continuare a farlo. the section called “Iscriviti ai messaggi con IoT Device Client”

Dimostra l'iscrizione ai messaggi con Device Client AWS IoT

In questa sezione verranno illustrati due tipi di sottoscrizione ai messaggi:

- Sottoscrizione ad argomento singolo
- Sottoscrizione ad argomento con caratteri jolly

Queste istruzioni di policy nella policy creata per questi esercizi consentono a Raspberry Pi di eseguire queste azioni:

- **`iot:Receive`**

  Fornisce al AWS IoT Device Client l'autorizzazione a ricevere argomenti MQTT che corrispondono a quelli nominati nell'`Resource`oggetto.

  ```
  {
    "Effect": "Allow",
    "Action": [
      "iot:Receive"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic"
    ]
  }
  ```

- **`iot:Subscribe`**

  Fornisce al AWS IoT Device Client l'autorizzazione a sottoscrivere i filtri degli argomenti MQTT che corrispondono a quelli nominati nell'`Resource`oggetto.

  ```
  {
    "Effect": "Allow",
    "Action": [
      "iot:Subscribe"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic"
    ]
  }
  ```

Iscriviti a un singolo argomento del messaggio MQTT

Questa procedura dimostra come AWS IoT Device Client può sottoscrivere e registrare i messaggi MQTT.

Nella finestra del terminale del computer host locale connesso al Raspberry Pi, elenca il contenuto di **`~/dc-configs/dc-pubsub-custom-config.json`** oppure aprire il file in un editor di testo per esaminarne il contenuto. Individua l'oggetto `samples`, che dovrebbe essere simile a questo.

```
"samples": {
  "pub-sub": {
    "enabled": true,
    "publish-topic": "test/dc/pubtopic",
    "publish-file": "~/messages/sample-ws-message.json",
    "subscribe-topic": "test/dc/subtopic",
    "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

Nota che il valore `subscribe-topic` è l'argomento MQTT che AWS IoT Device Client sottoscriverà quando viene eseguito. Il AWS IoT Device Client scrive i payload dei messaggi che riceve da questo abbonamento nel file indicato nel valore. `subscribe-file`

Per sottoscrivere un argomento relativo a un messaggio MQTT dal Device Client AWS IoT

1. Assicurati che sia la finestra terminale che la finestra con il client MQTT di test siano visibili durante l'esecuzione di questa procedura. Inoltre, assicurati che il tuo client MQTT di test abbia una sottoscrizione ancora attiva per il filtro di argomenti #. Se non lo è, effettua nuovamente la sottoscrizione al filtro argomento#.
2. Nella finestra del terminale, inserisci questi comandi per eseguire il AWS IoT Device Client utilizzando il file di configurazione creato in. the section called “Creare il file di configurazione”

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
   ```

   Nella finestra del terminale, il AWS IoT Device Client visualizza i messaggi informativi e gli eventuali errori che si verificano durante l'esecuzione.

   Se nella finestra del terminale non vengono visualizzati errori, continuare nella console AWS IoT .
3. Nella AWS IoT console, nel client di test MQTT, scegliete la scheda Pubblica su un argomento.

4. Alla voce Topic name (Nome argomento), inserisci **`test/dc/subtopic`**
5. Alla voce Message Payload (Payload del messaggio), esamina i contenuti del messaggio.
6. Per pubblicare il messaggio MQTT, scegli Publish (Pubblica).
7. Nella finestra del terminale, controllate la voce relativa al messaggio ricevuto dal AWS IoT Device Client che assomiglia a questa.

```
2021-11-10T16:02:20.890Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
 subscribe topic, size: 45 bytes
```

8. Dopo aver visualizzato la voce relativa al messaggio ricevuto che indica che il messaggio è stato ricevuto, digita **`^C`** (Ctrl-C) per interrompere il AWS IoT Device Client.
9. Inserisci questo comando per visualizzare la fine del file di log dei messaggi e visualizzare il messaggio pubblicato dal client MQTT di test.

```
tail ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

Visualizzando il messaggio nel file di log, è stato dimostrato che AWS IoT Device Client ha ricevuto il messaggio pubblicato dal client di test MQTT.

Sottoscrivi a più argomenti del messaggio MQTT utilizzando caratteri jolly

Queste procedure dimostrano come il AWS IoT Device Client può sottoscrivere e registrare i messaggi MQTT utilizzando caratteri jolly. Per fare ciò, dovrai:

1. Aggiornate il filtro degli argomenti utilizzato da AWS IoT Device Client per sottoscrivere gli argomenti MQTT.
2. Aggiornare la policy utilizzata dal dispositivo per consentire le nuove sottoscrizioni.
3. Esegui il AWS IoT Device Client e pubblica messaggi dalla console di test MQTT.

Per creare un file di configurazione per la sottoscrizione a più argomenti di messaggio MQTT utilizzando un filtro argomento MQTT con caratteri jolly

1. Nella finestra del terminale del computer host locale connesso al Raspberry Pi, apri **`~/dc-configs/dc-pubsub-custom-config.json`** per modificare e localizzare l'oggetto `samples`.
2. Nell'editor di testo, individua l'oggetto `samples` e aggiorna il valore `subscribe-topic` in modo che somigli a questo.

```
"samples": {
  "pub-sub": {
    "enabled": true,
    "publish-topic": "test/dc/pubtopic",
    "publish-file": "~/messages/sample-ws-message.json",
    "subscribe-topic": "test/dc/#",
    "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

Il nuovo valore `subscribe-topic` è un filtro di argomento MQTT con un carattere jolly MQTT alla fine. Descrive una sottoscrizione a tutti gli argomenti MQTT che iniziano con `test/dc/`. Il AWS IoT Device Client scrive i payload dei messaggi che riceve da questo abbonamento nel file indicato in. `subscribe-file`

3. Salva il file di configurazione modificato come **`~/dc-configs/dc-pubsub-wild-config.json`** ed esci dall'editor.

Per modificare la policy utilizzata da Raspberry Pi per consentire la sottoscrizione e la ricezione di più argomenti di messaggio MQTT

1. Nella finestra del terminale del computer host locale connesso al tuo Raspberry Pi, nel tuo editor di testo preferito, apri **`~/policies/pubsub_test_thing_policy.json`** per la modifica e quindi individua le istruzioni di policy `iot::Subscribe` e `iot::Receive` nel file.
2. Nell'istruzione di policy `iot::Subscribe`, aggiorna la stringa nell'oggetto Risorsa per sostituire `subtopic` con `*`, in modo tale che l'indirizzo sia simile al seguente.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/*"
  ]
}
```

> **Note**
>
> I caratteri jolly del filtro di argomento MQTT sono il + (segno più) e il # (segno cancelletto). Una richiesta di sottoscrizione con un # alla fine sottoscrive tutti gli argomenti che iniziano con la stringa che precede il carattere # (ad esempio, `test/dc/` in questo caso).
> Tuttavia, il valore della risorsa nell'istruzione di policy che autorizza questa sottoscrizione deve utilizzare un * (un asterisco) al posto di # (segno del cancelletto) nel filtro argomento ARN. Questo perché il processore di policy utilizza un carattere jolly diverso da quello utilizzato da MQTT.
> Per ulteriori informazioni sull'utilizzo di caratteri jolly per argomenti e filtri di argomento nelle policy, consulta Utilizzo di caratteri jolly in MQTT e nelle policy AWS IoT Core.

3. Nell'istruzione di policy `iot::Receive`, aggiorna la stringa nell'oggetto Risorsa per sostituire `subtopic` con `*`, in modo tale che l'indirizzo sia simile al seguente.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Receive"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*"
  ]
}
```

4. Salva il documento della policy aggiornato con nome **`~/policies/pubsub_wild_test_thing_policy.json`** ed esci dall'editor.

5. Inserisci questo comando per aggiornare la policy di questo tutorial per utilizzare le nuove definizioni delle risorse.

```
aws iot create-policy-version \
--set-as-default \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_wild_test_thing_policy.json"
```

Se il comando viene eseguito correttamente, verrà visualizzata una risposta simile alla seguente. Nota che `policyVersionId` ora è 2, indicando che questa è la seconda versione di questa policy.

Se hai aggiornato correttamente la policy, puoi passare alla procedura successiva.


```
{
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/PubSubTestThingPolicy",
    "policyDocument": "{\n  \"Version\": \"2012-10-17\",       \n  \"Statement
\": [\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": [\n
 \"iot:Connect\"\n      ],\n      \"Resource\": [\n        \"arn:aws:iot:us-
west-2:57EXAMPLE833:client/PubSubTestThing\"\n      ]\n    },\n    {\n
 \"Effect\": \"Allow\",\n      \"Action\": [\n        \"iot:Publish\"\n      ],
\n      \"Resource\": [\n        \"arn:aws:iot:us-west-2:57EXAMPLE833:topic/
test/dc/pubtopic\"\n      ]\n    },\n    {\n      \"Effect\": \"Allow\",\n
 \"Action\": [\n        \"iot:Subscribe\"\n      ],\n      \"Resource\": [\n
  \"arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/*\"\n      ]\n    },\n
   {\n      \"Effect\": \"Allow\",\n      \"Action\": [\n        \"iot:Receive\"\n
     ],\n      \"Resource\": [\n        \"arn:aws:iot:us-west-2:57EXAMPLE833:topic/
test/dc/*\"\n      ]\n    }\n  ]\n}\n",
    "policyVersionId": "2",
    "isDefaultVersion": true
}
```


Se si verifica un errore che contiene troppe versioni di policy per salvarne una nuova, inserisci questo comando per elencare le versioni attuali della policy. Esamina l'elenco restituito da questo comando per trovare una versione della policy che è possibile eliminare.

```
aws iot list-policy-versions --policy-name "PubSubTestThingPolicy"
```

Inserisci questo comando per eliminare una versione che non serve più. Attenzione: non puoi eliminare la versione di default della policy. La versione della policy di default è quella con un valore `isDefaultVersion` di `true`.

```
aws iot delete-policy-version \
--policy-name "PubSubTestThingPolicy" \
--policy-version-id policyId
```

Dopo aver eliminato una versione della policy, riprova questo passaggio.

Con il file di configurazione e la policy aggiornati, sei pronto a dimostrare gli abbonamenti wild card con Device Client. AWS IoT

Per dimostrare come AWS IoT Device Client sottoscrive e riceve più argomenti relativi ai messaggi MQTT

1. Nel client MQTT di test, controlla le sottoscrizioni. Se il client MQTT di test è sottoscritto al filtro dell'argomento **#**, passare alla fase successiva. In caso contrario, in MQTT test client (Client MQTT di test), nella scheda Subscribe to a topic (Sottoscrivi un argomento), in Topic filter (Filtro argomenti) inserisci **#** (un cancelletto), quindi scegli Subscribe (Sottoscrivi) per sottoscriverlo.
2. Nella finestra del terminale del computer host locale connesso a Raspberry Pi, inserisci questi comandi per avviare AWS IoT Device Client.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-wild-config.json
   ```
3. Mentre guardate l'output del AWS IoT Device Client nella finestra del terminale sul computer host locale, tornate al client di test MQTT. Nella scheda Publish to a topic (Pubblica in un argomento), in Topic name (Nome argomento), inserisci **`test/dc/subtopic`** e quindi scegli Publish (Pubblica).
4. Nella finestra del terminale, conferma che il messaggio è stato ricevuto cercando un messaggio come:

   ```
   2021-11-10T16:34:20.101Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
    subscribe topic, size: 76 bytes
   ```
5. Mentre guardate l'output del AWS IoT Device Client nella finestra del terminale del computer host locale, tornate al client di test MQTT. Nella scheda Publish to a topic (Pubblica in un argomento), in Topic name (Nome argomento), inserisci **`test/dc/subtopic2`** e quindi scegli Publish (Pubblica).
6. Nella finestra del terminale, conferma che il messaggio è stato ricevuto cercando un messaggio come:

   ```
   2021-11-10T16:34:32.078Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
    subscribe topic, size: 77 bytes
   ```
7. Dopo aver visualizzato i messaggi che confermano la ricezione di entrambi i messaggi, digitate **^C** (Ctrl-C) per fermare il AWS IoT Device Client.

8. Inserisci questo comando per visualizzare la fine del file di log dei messaggi e visualizzare il messaggio pubblicato dal client MQTT di test.

```
tail -n 20 ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

> **Note**
>
> Il file di log contiene solo payload del messaggio. Gli argomenti del messaggio non vengono registrati nel file di registro dei messaggi ricevuti.
> Potresti anche vedere il messaggio pubblicato dal AWS IoT Device Client nel registro ricevuto. Questo perché il filtro degli argomenti jolly include l'argomento del messaggio e, a volte, la richiesta di sottoscrizione può essere elaborata dal broker messaggi prima che il messaggio pubblicato venga inviato agli iscritti.

Le voci nel file di registro dimostrano che i messaggi sono stati ricevuti. È possibile ripetere questa procedura utilizzando altri nomi di argomenti. Tutti i messaggi con un nome argomento che iniziano per `test/dc/` devono essere ricevuti e registrati. I messaggi con nomi di argomenti che iniziano con qualsiasi altro testo vengono ignorati.

Dopo aver dimostrato come AWS IoT Device Client può pubblicare e sottoscrivere messaggi MQTT, continua con. Tutorial: Dimostra le azioni (lavori) remote con il AWS IoT Device Client

Tutorial: Dimostra le azioni (lavori) remote con il AWS IoT Device Client

In questi tutorial, configurerai e distribuirai i processi sul tuo Raspberry Pi per dimostrare come inviare operazioni remote ai tuoi dispositivi IoT.

Per iniziare il tutorial:

- Fai in modo che il tuo computer host locale sia configurato come nella sezione precedente.
- Se non hai completato il tutorial nella sezione precedente, puoi provare questo tutorial utilizzando il Raspberry Pi con una scheda microSD che contiene l'immagine che hai salvato dopo aver installato AWS IoT il Device Client in. (Facoltativo) Salvare l'immagine della scheda microSD
- Se hai già eseguito questa demo, prova ??? a eliminare tutte le AWS IoT risorse create nelle esecuzioni precedenti per evitare errori di risorse duplicate.

Questo tutorial dura circa 45 minuti.

Al termine di questo argomento:

- Avrai dimostrato diversi modi in cui il tuo dispositivo IoT può utilizzare AWS IoT Core per eseguire operazioni remote gestite da AWS IoT .

Equipaggiamento necessario:

- Il tuo ambiente locale di sviluppo e di test nella sezione precedente
- Il Raspberry Pi che hai testato in una sezione precedente
- La scheda di memoria microSD del Raspberry Pi utilizzata nella sezione precedente

Procedure all'interno del tutorial

- Prepara il Raspberry Pi per eseguire i lavori
- Crea ed esegui il job AWS IoT con AWS IoT Device Client

Prepara il Raspberry Pi per eseguire i lavori

Le procedure in questa sezione descrivono come preparare il Raspberry Pi all'esecuzione dei job utilizzando il AWS IoT Device Client.

> **Note**
>
> Queste procedure sono specifiche per il dispositivo. Se si desidera eseguire le procedure in questa sezione con più di un dispositivo contemporaneamente, ogni dispositivo avrà bisogno di una propria policy e di un certificato univoco specifico per il dispositivo e il nome dell'oggetto. Per fornire a ciascun dispositivo le sue risorse uniche, eseguire questa procedura una volta per ciascun dispositivo mentre si modificano gli elementi specifici del dispositivo come descritto nelle procedure.

Procedure all'interno del tutorial

- Effettua il provisioning del tuo Raspberry Pi per dimostrare i processi
- Configura il AWS IoT Device Client per eseguire il jobs agent

Effettua il provisioning del tuo Raspberry Pi per dimostrare i processi

Le procedure descritte in questa sezione forniscono il tuo Raspberry Pi AWS IoT creando AWS IoT risorse e certificati dei dispositivi relativi al dispositivo.

Argomenti

- Crea e scarica i file dei certificati dei dispositivi per dimostrare AWS IoT i lavori
- Crea AWS IoT risorse per dimostrare i AWS IoT lavori

Crea e scarica i file dei certificati dei dispositivi per dimostrare AWS IoT i lavori

Questa procedura crea i file di certificato del dispositivo per questa demo.

Se si stanno preparando più di un dispositivo, questa procedura deve essere eseguita su ciascun dispositivo.

Per creare e scaricare i file di certificato del dispositivo per il tuo Raspberry Pi:

Nella finestra terminale del computer host locale connesso al Raspberry Pi, inserisci questi comandi.

1. Inserisci il seguente comando per creare i file dei certificati del dispositivo per il dispositivo.

```
aws iot create-keys-and-certificate \
--set-as-active \
--certificate-pem-outfile "~/certs/jobs/device.pem.crt" \
--public-key-outfile "~/certs/jobs/public.pem.key" \
--private-key-outfile "~/certs/jobs/private.pem.key"
```

   Questo comando restituisce una risposta simile alla seguente. Salva il *certificateArn* per utilizzarlo in un secondo momento.

```
{
"certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificateId":
 "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
\n",
"keyPair": {
```

```
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
}
}
```

2. Inserisci i seguenti comandi per impostare le autorizzazioni sulla directory dei certificati e sui relativi file.

```
chmod 700 ~/certs/jobs
chmod 644 ~/certs/jobs/*
chmod 600 ~/certs/jobs/private.pem.key
```

3. Esegui questo comando per rivedere le autorizzazioni sulle directory e sui file dei certificati.

```
ls -l ~/certs/jobs
```

   L'output del comando dovrebbe essere lo stesso di quello che vedi qui, tranne per le date e gli orari del file.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

Dopo aver scaricato i file del certificato del dispositivo sul tuo Raspberry Pi, sei pronto a proseguire con the section called “Fornisci Raspberry Pi per i lavori”.

Crea AWS IoT risorse per dimostrare i AWS IoT lavori

Crea le AWS IoT risorse per questo dispositivo.

Se si stanno preparando più dispositivi, questa procedura deve essere eseguita su ciascun dispositivo.

Per effettuare il provisioning del dispositivo in AWS IoT:

Nella finestra terminale del computer host locale connesso al Raspberry Pi:

1. Inserisci il seguente comando per ottenere l'indirizzo dell'endpoint dati del dispositivo per il Account AWS.

   ```
   aws iot describe-endpoint --endpoint-type IoT:Data-ATS
   ```

   Il valore dell'endpoint non è cambiato dall'ultima volta che hai eseguito questo comando. Il comando viene eseguito nuovamente per trovare e incollare facilmente il valore dell'endpoint dei dati nel file di configurazione utilizzato in questo tutorial.

   Il comando describe-endpoint restituisce una risposta simile alla seguente. Registra il valore *endpointAddress* per utilizzarlo in seguito.

   ```
   {
   "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Sostituiscilo *uniqueThingName* con un nome univoco per il tuo dispositivo. Se si desidera eseguire questo tutorial con più dispositivi, assegnare a ciascun dispositivo il proprio nome. Ad esempio, **TestDevice01**, **TestDevice02**, e così via.

   Inserisci questo comando per creare una nuova AWS IoT risorsa per il tuo Raspberry Pi.

   ```
   aws iot create-thing --thing-name "uniqueThingName"
   ```

   Poiché una risorsa AWS IoT oggetto è una rappresentazione virtuale del tuo dispositivo nel cloud, possiamo creare più risorse di oggetti AWS IoT da utilizzare per scopi diversi. Possono essere utilizzati tutti dallo stesso dispositivo IoT fisico per rappresentare diversi aspetti del dispositivo.

   > **Note**
   >
   > Se si desidera proteggere la policy per più dispositivi, è possibile utilizzare `${iot:Thing.`*ThingName*`}` invece del nome statico dell'oggetto, *uniqueThingName*.

   Questi tutorial utilizzeranno una sola risorsa alla volta per dispositivo. In questo modo, in questi tutorial, rappresentano le diverse demo in modo che, dopo aver creato le AWS IoT risorse

per una demo, sia possibile tornare indietro e ripetere le demo utilizzando le risorse create appositamente per ciascuna di esse.

Se la risorsa AWS IoT oggetto è stata creata, il comando restituisce una risposta come questa. Registra il valore *thingArn* da utilizzare in un secondo momento quando si crea il processo da eseguire su questo dispositivo.

```
{
"thingName": "uniqueThingName",
"thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/uniqueThingName",
"thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. Nella finestra terminale:

   a. Apri un editor di testo, ad esempio `nano`.

   b. Copia questo documento JSON e incollalo nell'editor di testo aperto.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:client/uniqueThingName"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/pubtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/job/
*",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/
jobExecution/*",
```

```
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topicfilter/test/dc/
subtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/
jobExecution/*",
                "arn:aws:iot:us-west-2:123456789012:topicfilter/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/subtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeJobExecution",
                "iot:GetPendingJobExecutions",
                "iot:StartNextPendingJobExecution",
                "iot:UpdateJobExecution"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName"
            ]
        }
    ]
```

```
}
```

c. Nell'editor, nella `Resource` sezione di ogni dichiarazione politica, sostituisci *us-west-2:57EXAMPLE833* con i tuoi Regione AWS, un carattere con i due punti (:) e il tuo numero di 12 cifre Account AWS .

d. Nell'editor, in ogni dichiarazione politica, sostituisci questa risorsa *uniqueThingName* con il nome dell'oggetto che hai assegnato.

e. Salva il file nell'editor di testo come **~/policies/jobs_test_thing_policy.json**.

Se si esegue questa procedura per più dispositivi, salvare il file con questo nome file su ciascun dispositivo.

4. Sostituisci *uniqueThingName* con il nome dell'oggetto per il dispositivo, quindi esegui questo comando per creare una AWS IoT policy personalizzata per quel dispositivo.

```
aws iot create-policy \
--policy-name "JobTestPolicyFor uniqueThingName" \
--policy-document "file://~/policies/jobs_test_thing_policy.json"
```

Se la policy viene creata, verrà visualizzata una risposta simile alla seguente.

```
{
    "policyName": "JobTestPolicyFor uniqueThingName",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
JobTestPolicyFor uniqueThingName",
    "policyDocument": "{\n\"Version\": \"2012-10-17\",\n\"Statement\": [\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Connect\"\n],\n\"Resource\":
 [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n},\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Publish\"\n],\n\"Resource\":
 [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n},\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Subscribe\"\n],\n\"Resource\": [\n
\"arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic\"\n]\n},\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Receive\"\n],\n\"Resource\": [\n
\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*\"\n]\n}\n]\n}\n",
    "policyVersionId": "1"
}
```

5. Sostituiscilo *uniqueThingName* con il nome dell'oggetto e *certificateArn* con il `certificateArn` valore salvato in precedenza in questa sezione per questo dispositivo, quindi esegui questo comando per allegare la policy al certificato del dispositivo.

```
aws iot attach-policy \
--policy-name "JobTestPolicyFor<uniqueThingName>" \
--target "<certificateArn>"
```

In caso di successo, questo comando non restituisce alcun risultato.

6. Sostituisci *uniqueThingName* con il nome dell'oggetto, sostituiscilo *certificateArn* con il `certificateArn` valore salvato in precedenza in questa sezione e quindi esegui questo comando per allegare il certificato del dispositivo alla risorsa AWS IoT oggetto.

```
aws iot attach-thing-principal \
--thing-name "<uniqueThingName>" \
--principal "<certificateArn>"
```

In caso di successo, questo comando non restituisce alcun risultato.

Dopo aver eseguito correttamente il provisioning del Raspberry Pi, sei pronto a ripetere questa sezione per un altro Raspberry Pi nel test, oppure, se tutti i dispositivi sono stati sottoposti a provisioning, prosegui con the section called “Configura Device Client per eseguire i lavori”.

Configura il AWS IoT Device Client per eseguire il jobs agent

Questa procedura crea un file di configurazione per il AWS IoT Device Client per eseguire il jobs agent:.

Se si sta preparando più di un dispositivo, questa procedura deve essere eseguita su ciascun dispositivo.

Per creare il file di configurazione per testare il AWS IoT Device Client:

1. Nella finestra terminale del computer host locale connesso al Raspberry Pi:

   a. Apri un editor di testo, ad esempio `nano`.

   b. Copia questo documento JSON e incollalo nell'editor di testo aperto.

```
{
  "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
  "cert": "~/certs/jobs/device.pem.crt",
  "key": "~/certs/jobs/private.pem.key",
  "root-ca": "~/certs/AmazonRootCA1.pem",
```

```
"thing-name": "uniqueThingName",
"logging": {
  "enable-sdk-logging": true,
  "level": "DEBUG",
  "type": "STDOUT",
  "file": ""
},
"jobs": {
  "enabled": true,
  "handler-directory": ""
},
"tunneling": {
  "enabled": false
},
"device-defender": {
  "enabled": false,
  "interval": 300
},
"fleet-provisioning": {
  "enabled": false,
  "template-name": "",
  "template-parameters": "",
  "csr-file": "",
  "device-key": ""
},
"samples": {
  "pub-sub": {
    "enabled": false,
    "publish-topic": "",
    "publish-file": "",
    "subscribe-topic": "",
    "subscribe-file": ""
  }
},
"config-shadow": {
  "enabled": false
},
"sample-shadow": {
  "enabled": false,
  "shadow-name": "",
  "shadow-input-file": "",
  "shadow-output-file": ""
}
```

```
}
```

c. Sostituisci il *endpoint* valore con il valore dell'endpoint dei dati del dispositivo Account AWS che hai trovato in. the section called “Effettua il provisioning del dispositivo in AWS IoT Core”

d. Sostituiscilo *uniqueThingName* con il nome dell'oggetto che hai usato per questo dispositivo.

e. Salva il file nell'editor di testo come **~/dc-configs/dc-jobs-config.json**.

2. Esegui questo comando per impostare le autorizzazioni per il file del nuovo file di configurazione.

```
chmod 644 ~/dc-configs/dc-jobs-config.json
```

Non utilizzerai il client MQTT di test per questo test. Sebbene il dispositivo scambia messaggi MQTT relativi ai lavori con AWS IoT, i messaggi di avanzamento del lavoro vengono scambiati solo con il dispositivo che esegue il processo. Poiché i messaggi sullo stato di avanzamento del lavoro vengono scambiati solo con il dispositivo che esegue il processo, non è possibile abbonarsi ad essi da un altro dispositivo, ad esempio la console. AWS IoT

Dopo aver salvato il file di configurazione, è possibile proseguire con the section called “Crea ed esegui un lavoro IoT”.

Crea ed esegui il job AWS IoT con AWS IoT Device Client

Le procedure in questa sezione creano un documento di lavoro e una risorsa AWS IoT lavorativa. Dopo aver creato la risorsa lavorativa, AWS IoT invia il documento del lavoro agli obiettivi di lavoro specificati a cui un agente di lavoro applica il documento di lavoro al dispositivo o al client.

Procedure in questa sezione

- Crea e archivia il documento di lavoro per il lavoro IoT
- Esegui un lavoro AWS IoT per un dispositivo IoT

Crea e archivia il documento di lavoro per il lavoro IoT

Questa procedura crea un semplice documento di lavoro da includere in una risorsa AWS IoT lavorativa. Questo documento di processo mostra "Hello world!" sul target di processo.

Per creare e archiviare un documento di processo:

1. Seleziona il bucket Amazon S3 in cui salvare il documento di processo. Se non disponi di un bucket Amazon S3 esistente da utilizzare per questo, dovrai creane uno. Per informazioni su come creare bucket Amazon S3, consulta gli argomenti in Nozioni di base su Amazon S3.
2. Creare e salvare il documento di processo per questo processo

   a. Sul computer host locale, apri un editor di testo.

   b. Copia e incolla questo testo nell'editor.

   ```
   {
       "operation": "echo",
       "args": ["Hello world!"]
   }
   ```

   c. Nel computer host locale, salvare il contenuto dell'editor in un file denominato **`hello-world-job.json`**.

   d. Conferma che il file sia stato salvato correttamente. Alcuni editor di testo aggiungono automaticamente `.txt` al nome del file quando salva un file di testo. Se il tuo editor ha aggiunto `.txt` al nome del file, correggere il nome del file prima di procedere.

3. Sostituisci il percorso *`path_to_file`* con il percorso verso**`hello-world-job.json`**, se non è nella tua directory corrente, sostituiscilo *`s3_bucket_name`* con il percorso del bucket Amazon S3 che hai selezionato, quindi esegui questo comando per inserire il documento di lavoro nel bucket Amazon S3.

   ```
   aws s3api put-object \
   --key hello-world-job.json \
   --body path_to_file/hello-world-job.json --bucket s3_bucket_name
   ```

   L'URL del documento di lavoro che identifica il documento di lavoro archiviato in Amazon S3 viene determinato sostituendo *`s3_bucket_name`* l'*`AWS_region`*e nel seguente URL. Registra l'URL risultante da utilizzare in seguito come *`job_document_path`*

   ```
   https://s3_bucket_name.s3.AWS_Region.amazonaws.com/hello-world-job.json
   ```

> **Note**
>
> AWS la sicurezza ti impedisce di aprire questo URL al di fuori del tuo Account AWS, ad esempio utilizzando un browser. Per impostazione predefinita, l'URL viene utilizzato dal motore di AWS IoT job, che ha accesso al file. In un ambiente di produzione, è necessario assicurarsi che i servizi AWS IoT abbiano l'autorizzazione per l'accesso ai documenti di processo archiviati in Amazon S3.

Dopo aver salvato l'URL del documento di processo, prosegui con the section called “Esegui un processo per un singolo dispositivo”.

Esegui un lavoro AWS IoT per un dispositivo IoT

Le procedure descritte in questa sezione avviano il AWS IoT Device Client sul tuo Raspberry Pi per eseguire il job agent sul dispositivo e attendere l'esecuzione dei job. Crea anche una risorsa di lavoro in AWS IoT, che invierà il lavoro al tuo dispositivo IoT e lo eseguirà sul tuo dispositivo IoT.

> **Note**
>
> Questa procedura esegue un processo su un solo dispositivo.

Per avviare l'agente di processi sul tuo Raspberry Pi:

1. Nella finestra del terminale del computer host locale collegato al Raspberry Pi, esegui questo comando per avviare il AWS IoT Device Client.

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-jobs-config.json
```

2. Nella finestra del terminale, verifica che il AWS IoT Device Client e visualizzi questi messaggi

```
2021-11-15T18:45:56.708Z [INFO]  {Main.cpp}: Jobs is enabled
                        .
                        .
                        .
2021-11-15T18:45:56.708Z [INFO]  {Main.cpp}: Client base has been notified that
 Jobs has started
```

```
2021-11-15T18:45:56.708Z [INFO]  {JobsFeature.cpp}: Running Jobs!
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 startNextPendingJobExecution accepted and rejected
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 nextJobChanged events
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 updateJobExecutionStatusAccepted for jobId +
2021-11-15T18:45:56.738Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToUpdateJobExecutionAccepted with code {0}
2021-11-15T18:45:56.739Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 updateJobExecutionStatusRejected for jobId +
2021-11-15T18:45:56.753Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToNextJobChanged with code {0}
2021-11-15T18:45:56.760Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToStartNextJobRejected with code {0}
2021-11-15T18:45:56.776Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToStartNextJobAccepted with code {0}
2021-11-15T18:45:56.776Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToUpdateJobExecutionRejected with code {0}
2021-11-15T18:45:56.777Z [DEBUG] {JobsFeature.cpp}: Publishing
 startNextPendingJobExecutionRequest
2021-11-15T18:45:56.785Z [DEBUG] {JobsFeature.cpp}: Ack received for
 StartNextPendingJobPub with code {0}
2021-11-15T18:45:56.785Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

3. Nella finestra del terminale, dopo aver visualizzato questo messaggio, continua con la procedura successiva e crea la risorsa del processo. Attenzione: potrebbe non essere l'ultima voce dell'elenco.

```
2021-11-15T18:45:56.785Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

Per creare una risorsa AWS IoT lavorativa

1. Sul computer host locale:

   a. Sostituisci *job_document_url* con l'URL del documento di lavoro dathe section called “Crea e archivia il documento di lavoro”.

   b. Sostituisci *thing_arn* con l'ARN della risorsa oggetto che hai creato per il tuo dispositivo, quindi esegui questo comando.

```
aws iot create-job \
--job-id hello-world-job-1 \
--document-source "job_document_url" \
--targets "thing_arn" \
--target-selection SNAPSHOT
```

Se il comando viene eseguito correttamente, verrà visualizzato un risultato simile a quanto segue.

```
{
  "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-job-1",
  "jobId": "hello-world-job-1"
}
```

2. Nella finestra del terminale, dovresti vedere un output del AWS IoT Device Client in questo modo.

```
2021-11-15T18:02:26.688Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Job ids differ
2021-11-15T18:10:24.890Z [INFO]  {JobsFeature.cpp}: Executing job: hello-world-
job-1
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Attempting to update job
 execution status!
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Not including stdout with the
 status details
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Not including stderr with the
 status details
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Assuming executable is in PATH
2021-11-15T18:10:24.890Z [INFO]  {JobsFeature.cpp}: About to execute: echo Hello
 world!
2021-11-15T18:10:24.890Z [DEBUG] {Retry.cpp}: Retryable function starting, it will
 retry until success
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Created EphermalPromise for
 ClientToken 3TEWba9Xj6 in the updateJobExecution promises map
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Child process now running
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Child process about to call
 execvp
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Parent process now running, child
 PID is 16737
2021-11-15T18:10:24.891Z [DEBUG] {16737}: Hello world!
```

```
2021-11-15T18:10:24.891Z [DEBUG] {JobEngine.cpp}: JobEngine finished waiting for
 child process, returning 0
2021-11-15T18:10:24.891Z [INFO]  {JobsFeature.cpp}: Job exited with status: 0
2021-11-15T18:10:24.891Z [INFO]  {JobsFeature.cpp}: Job executed successfully!
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Attempting to update job
 execution status!
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Not including stdout with the
 status details
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Not including stderr with the
 status details
2021-11-15T18:10:24.892Z [DEBUG] {Retry.cpp}: Retryable function starting, it will
 retry until success
2021-11-15T18:10:24.892Z [DEBUG] {JobsFeature.cpp}: Created EphermalPromise for
 ClientToken GmQ0HTzWGg in the updateJobExecution promises map
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Ack received for
 PublishUpdateJobExecutionStatus with code {0}
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Removing ClientToken 3TEWba9Xj6
 from the updateJobExecution promises map
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Success response after
 UpdateJobExecution for job hello-world-job-1
2021-11-15T18:10:24.917Z [DEBUG] {JobsFeature.cpp}: Ack received for
 PublishUpdateJobExecutionStatus with code {0}
2021-11-15T18:10:24.918Z [DEBUG] {JobsFeature.cpp}: Removing ClientToken GmQ0HTzWGg
 from the updateJobExecution promises map
2021-11-15T18:10:24.918Z [DEBUG] {JobsFeature.cpp}: Success response after
 UpdateJobExecution for job hello-world-job-1
2021-11-15T18:10:25.861Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

3. Mentre il AWS IoT Device Client è in esecuzione e attende un lavoro, è possibile inviare un altro lavoro modificando il `job-id` valore e rieseguendolo create-job dal passaggio 1.

Quando hai finito di eseguire i lavori, nella finestra del terminale, inserisci ^C (control-C) per arrestare il AWS IoT Device Client.

Tutorial: pulizia dopo l'esecuzione dei tutorial AWS IoT Device Client

Le procedure contenute in questo tutorial ti guidano nella rimozione dei file e delle risorse che hai creato durante il completamento dei tutorial in questo percorso di apprendimento.

Procedure all'interno del tutorial

- Fase 1: Pulizia dei dispositivi dopo aver creato delle demo con Device Client AWS IoT

- Passo 2: Pulizia delle demo Account AWS dopo la creazione con AWS IoT Device Client

Fase 1: Pulizia dei dispositivi dopo aver creato delle demo con Device Client AWS IoT

Questo tutorial descrive due opzioni su come pulire la scheda microSD dopo aver creato le demo in questo percorso di apprendimento. Scegli l'opzione che fornisce il livello di sicurezza di cui hai bisogno.

Tieni presente che la pulizia della scheda microSD del dispositivo non rimuove le risorse che hai AWS IoT creato. Per ripulire le AWS IoT risorse dopo aver pulito la scheda microSD del dispositivo, è necessario consultare il tutorial su. the section called “Pulizia dopo aver creato le demo con Device Client AWS IoT ”

Opzione 1: Effettuare la pulizia riscrivendo la scheda microSD

Il modo più semplice e completo per pulire la scheda microSD dopo aver completato i tutorial in questo percorso di apprendimento è sovrascrivere la scheda microSD con un file immagine salvato, creato durante la preparazione del dispositivo la prima volta.

Questa procedura utilizza il computer host locale per scrivere un'immagine della scheda microSD salvata su una scheda microSD.

> **Note**
>
> Se il dispositivo non utilizza un supporto di archiviazione rimovibile per il sistema operativo, fai riferimento alla procedura per quel dispositivo.

Per scrivere una nuova immagine alla scheda microSD

1. Sul computer host locale, individua l'immagine della scheda microSD salvata che si desidera scrivere sulla scheda microSD.
2. Inserisci la microSD nel computer host locale.
3. Utilizzando uno strumento di imaging per schede SD, scrivi il file immagine selezionato sulla scheda microSD.
4. Dopo aver scritto l'immagine del sistema operativo Raspberry Pi sulla scheda microSD, espelli la scheda microSD e rimuovila in modo sicuro dal computer host locale.

La tua scheda microSD è pronta per l'uso.

Opzione 2: Effettuare la pulizia eliminando le directory utente

Per pulire la scheda microSD dopo aver completato i tutorial senza riscrivere l'immagine della scheda microSD, è possibile eliminare le directory utente singolarmente. Ciò non è accurato come riscrivere la scheda microSD da un'immagine salvata perché non rimuove i file di sistema che potrebbero essere stati installati.

Se la rimozione delle directory utente è sufficientemente completa per le tue esigenze, puoi seguire questa procedura.

Eliminare le directory utente di questo percorso di apprendimento dal dispositivo

1. Esegui questi comandi per eliminare le directory utente, le sottodirectory e tutti i file creati in questo percorso di apprendimento, nella finestra del terminale connesso al dispositivo.

   > **Note**
   >
   > Dopo l'eliminazione di queste directory e file, non potrai eseguire le demo senza completare nuovamente i tutorial.

   ```
   rm -Rf ~/dc-configs
   rm -Rf ~/policies
   rm -Rf ~/messages
   rm -Rf ~/certs
   rm -Rf ~/.aws-iot-device-client
   ```

2. Eseguire questi comandi per eliminare le directory e i file di origine dell'applicazione, nella finestra terminale connessa al dispositivo.

   > **Note**
   >
   > Questi comandi non disinstallano alcun programma. Rimuovono solo i file di origine utilizzati per compilarli e installarli. Dopo aver eliminato questi file, il dispositivo AWS CLI e il client del AWS IoT dispositivo potrebbero non funzionare.

   ```
   rm -Rf ~/aws-cli
   ```

```
rm -Rf ~/aws
rm -Rf ~/aws-iot-device-client
```

Passo 2: Pulizia delle demo Account AWS dopo la creazione con AWS IoT Device Client

Queste procedure consentono di identificare e rimuovere le AWS risorse create durante il completamento dei tutorial di questo percorso di apprendimento.

Pulisci le risorse AWS IoT

Questa procedura consente di identificare e rimuovere le AWS IoT risorse create durante il completamento dei tutorial di questo percorso di apprendimento.

AWS IoT risorse create in questo percorso di apprendimento

Tutorial	Risorse oggetto	Risorse policy
the section called “Installazione e configurazione del client per dispositivi IoT”	DevCliTestThing	DevCliTestThingPolicy
the section called “Comunica con il client del dispositivo utilizzando MQTT”	PubSubTestThing	PubSubTestThingPolicy
the section called “Esegui lavori IoT con Device Client”	definito dall'utente (potrebbero essercene più di uno)	definito dall'utente (potrebbero essercene più di uno)

Per eliminare le AWS IoT risorse, segui questa procedura per ogni risorsa oggetto che hai creato

1. Sostituisci *thing_name* con il nome della risorsa dell'oggetto che desideri eliminare, quindi esegui questo comando per elencare i certificati attribuiti alla risorsa dell'oggetto, dal computer host locale.

   ```
   aws iot list-thing-principals --thing-name thing_name
   ```

   Questo comando restituisce una risposta come questa che elenca i certificati attribuiti a *thing_name*. Nella maggior parte dei casi, nell'elenco sarà presente un solo certificato.

```
{
    "principals": [
        "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/23853eea3cf0edc7f8a69c74abeafa27b2b52823cab5b3e156295e94b26ae8ac"
    ]
}
```

2. Per ogni certificato elencato dal comando precedente:

   a. Sostituisci *certificate_ID* con l'ID del certificato del comando precedente. L'ID certificato è costituito dai caratteri alfanumerici che seguono `cert/` nell'ARN restituito dal comando precedente. Quindi esegui questo comando per disattivare il certificato.

   ```
   aws iot update-certificate --new-status INACTIVE --certificate-
   id certificate_ID
   ```

   In caso di esito positivo, questo comando non restituisce alcun risultato.

   b. Sostituisci *certificate_ARN* con il certificato ARN dall'elenco dei certificati restituiti in precedenza, quindi esegui questo comando per elencare le policy attribuite a questo certificato.

   ```
   aws iot list-attached-policies --target certificate_ARN
   ```

   Questo comando restituisce una risposta come questa che elenca le policy attribuite al certificato. Nella maggior parte dei casi, ci sarà solo una policy nell'elenco.

   ```
   {
       "policies": [
           {
               "policyName": "DevCliTestThingPolicy",
               "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
   DevCliTestThingPolicy"
           }
       ]
   }
   ```

   c. Per ciascuna policy collegata al certificato:

i. Sostituisci *policy_name* con il valore `policyName` del comando precedente, sostituisci *certificate_ARN* con l'ARN del certificato, quindi esegui questo comando per disconnettere la policy dal certificato.

```
aws iot detach-policy --policy-name policy_name --target certificate_ARN
```

In caso di esito positivo, questo comando non restituisce alcun risultato.

ii. Sostituisci *policy_name* con il valore `policyName` e quindi esegui questo comando per verificare se la policy è attribuita ad altri certificati.

```
aws iot list-targets-for-policy --policy-name policy_name
```

Se il comando restituisce un elenco vuoto come questo, la policy non è associata a nessun certificato; continuerai a elencare le versioni della policy. Se sono ancora presenti certificati allegati alla policy, continua con la fase detach-thing-principal.

```
{
    "targets": []
}
```

iii. Sostituisci *policy_name* con il valore `policyName`, quindi esegui questo comando per verificare la presenza di versioni delle policy. Per eliminare la policy, deve avere una sola versione.

```
aws iot list-policy-versions --policy-name policy_name
```

Se la policy ha una sola versione, come in questo esempio, è possibile passare alla fase delete-policy ed eliminare subito la policy.

```
{
    "policyVersions": [
        {
            "versionId": "1",
            "isDefaultVersion": true,
            "createDate": "2021-11-18T01:02:46.778000+00:00"
        }
    ]
}
```

Se la policy ha più di una versione, come in questo esempio, le versioni delle policy con un valore `isDefaultVersion` di `false` devono essere eliminate prima che la policy possa essere eliminata.

```
{
    "policyVersions": [
        {
            "versionId": "2",
            "isDefaultVersion": true,
            "createDate": "2021-11-18T01:52:04.423000+00:00"
        },
        {
            "versionId": "1",
            "isDefaultVersion": false,
            "createDate": "2021-11-18T01:30:18.083000+00:00"
        }
    ]
}
```

Se è necessario eliminare una versione della policy, sostituisci *`policy_name`* con il valore `policyName`, sostituisci *`version_ID`* con il valore `versionId` del comando precedente, quindi esegui questo comando per eliminare una versione della policy.

```
aws iot delete-policy-version --policy-name policy_name --policy-version-
id version_ID
```

In caso di esito positivo, questo comando non restituisce alcun risultato.

Dopo aver eliminato una versione della policy, ripeti questo passaggio fino a quando la policy non ha una sola versione della policy.

iv. Sostituisci *`policy_name`* con il valore `policyName` e quindi esegui questo comando per eliminare la policy.

```
aws iot delete-policy --policy-name policy_name
```

d. Sostituisci *`thing_name`* con il nome dell'oggetto, sostituisci *`certificate_ARN`* con l'ARN del certificato, quindi esegui questo comando per disconnettere il certificato dalla risorsa dell'oggetto.

```
aws iot detach-thing-principal --thing-name thing_name --
principal certificate_ARN
```

In caso di esito positivo, questo comando non restituisce alcun risultato.

e. Sostituisci *certificate_ID* con l'ID del certificato del comando precedente. L'ID certificato è costituito dai caratteri alfanumerici che seguono `cert/` nell'ARN restituito dal comando precedente. Esegui questo comando per eliminare la risorsa del certificato.

```
aws iot delete-certificate --certificate-id certificate_ID
```

In caso di esito positivo, questo comando non restituisce alcun risultato.

3. Sostituisci *thing_name* con il nome dell'oggetto, quindi esegui questo comando per eliminarlo.

```
aws iot delete-thing --thing-name thing_name
```

In caso di esito positivo, questo comando non restituisce alcun risultato.

Pulisci AWS le risorse

Questa procedura consente di identificare e rimuovere altre AWS risorse create durante il completamento dei tutorial di questo percorso di apprendimento.

Altre AWS risorse create in questo percorso di apprendimento

Tutorial	Tipo di risorsa	Nome o ID risorsa
the section called “Esegui lavori IoT con Device Client”	Oggetto Amazon S3	hello-world-job.json
the section called “Esegui lavori IoT con Device Client”	AWS IoT risorse lavorative	definito dall'utente

Per eliminare le AWS risorse create in questo percorso di apprendimento

1. Per eliminare i processi creati in questo percorso di apprendimento

   a. Esegui questo comando per elencare i lavori presenti nel tuo Account AWS.

```
aws iot list-jobs
```

Il comando restituisce un elenco dei AWS IoT lavori presenti nel tuo Account AWS e Regione AWS che assomiglia a questo.

```
{
    "jobs": [
        {
            "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-
job-2",
            "jobId": "hello-world-job-2",
            "targetSelection": "SNAPSHOT",
            "status": "COMPLETED",
            "createdAt": "2021-11-16T23:40:36.825000+00:00",
            "lastUpdatedAt": "2021-11-16T23:40:41.375000+00:00",
            "completedAt": "2021-11-16T23:40:41.375000+00:00"
        },
        {
            "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-
job-1",
            "jobId": "hello-world-job-1",
            "targetSelection": "SNAPSHOT",
            "status": "COMPLETED",
            "createdAt": "2021-11-16T23:35:26.381000+00:00",
            "lastUpdatedAt": "2021-11-16T23:35:29.239000+00:00",
            "completedAt": "2021-11-16T23:35:29.239000+00:00"
        }
    ]
}
```

b. Per ogni job che riconosci dall'elenco come job creato in questo percorso di apprendimento, sostituiscilo *`jobId`* con il `jobId` valore del job da eliminare, quindi esegui questo comando per eliminare un AWS IoT job.

```
aws iot delete-job --job-id jobId
```

In caso di esito positivo, il comando non restituisce alcun risultato.

2. Eliminare i documenti di processo archiviati in un bucket Amazon S3 in questo percorso di apprendimento.

a. Sostituisci *bucket* con il nome del bucket utilizzato, quindi esegui questo comando per elencare gli oggetti nel bucket Amazon S3 utilizzato.

```
aws s3api list-objects --bucket bucket
```

Il comando restituisce un elenco di oggetti Amazon S3 nel bucket simile a questo.

```
{
    "Contents": [
        {
            "Key": "hello-world-job.json",
            "LastModified": "2021-11-18T03:02:12+00:00",
            "ETag": "\"868c8bc3f56b5787964764d4b18ed5ef\"",
            "Size": 54,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        },
        {
            "Key": "iot_job_firmware_update.json",
            "LastModified": "2021-04-13T21:57:07+00:00",
            "ETag": "\"7c68c591949391791ecf625253658c61\"",
            "Size": 66,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        },
        {
            "Key": "order66.json",
            "LastModified": "2021-04-13T21:57:07+00:00",
            "ETag": "\"bca60d5380b88e1a70cc27d321caba72\"",
            "Size": 29,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
```

```
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        }
    ]
}
```

b. Per ogni processo che riconosci dall'elenco come processo creato in questo percorso di apprendimento, sostituisci *bucket* con il nome del bucket e *key* con il valore chiave dell'oggetto da eliminare, quindi esegui questo comando per eliminare un oggetto Amazon S3.

```
aws s3api delete-object --bucket bucket --key key
```

In caso di esito positivo, il comando non restituisce alcun risultato.

Dopo aver eliminato tutte le AWS risorse e gli oggetti creati durante il completamento di questo percorso di apprendimento, puoi ricominciare da capo e ripetere i tutorial.

Creazione di soluzioni con il AWS IoT dispositivo SDKs

I tutorial in questa sezione ti guidano attraverso i passaggi per sviluppare una soluzione IoT che può essere implementata in un ambiente di produzione utilizzando. AWS IoT

Il completamento di questi tutorial può richiedere più tempo rispetto a quelli della sezione precedente, the section called “Creazione di demo con AWS IoT Device Client” poiché utilizzano il AWS IoT dispositivo SDKs e spiegano i concetti applicati in modo più dettagliato per aiutarti a creare soluzioni sicure e affidabili.

Inizia a creare soluzioni con il dispositivo AWS IoT SDKs

Questi tutorial ti guidano attraverso diversi AWS IoT scenari. Se del caso, i tutorial utilizzano il dispositivo. AWS IoT SDKs

Argomenti

- Tutorial: Connessione di un dispositivo AWS IoT Core tramite AWS IoT Device SDK
- Creazione di AWS IoT regole per indirizzare i dati del dispositivo ad altri servizi
- Mantenimento dello stato del dispositivo mentre il dispositivo è disconnesso da Device Shadows

- Tutorial: creazione di un autorizzatore personalizzato per AWS IoT Core
- Tutorial: monitoraggio dell'umidità del suolo con AWS IoT Raspberry Pi

Tutorial: Connessione di un dispositivo AWS IoT Core tramite AWS IoT Device SDK

Questo tutorial mostra come connettere un dispositivo a AWS IoT Core cui inviare e ricevere dati da e verso. AWS IoT Dopo aver completato questo tutorial, il dispositivo sarà configurato per la connessione AWS IoT Core e capirai come comunicano i AWS IoT dispositivi.

Argomenti

- Prerequisiti
- Prepara il tuo dispositivo per AWS IoT
- Esamina il protocollo MQTT
- Revisione dell'app di esempio pubsub.py SDK per dispositivi
- Connect il dispositivo e comunica con AWS IoT Core
- Rivedi i risultati
- Tutorial: Utilizzo del SDK per dispositivi AWS IoT per Embedded C

Prerequisiti

Prima di iniziare il tutorial, assicurati di disporre:

- Aver completato Guida introduttiva ai AWS IoT Core tutorial

  Nella sezione di quel tutorial in cui è necessario the section called “Configurazione del dispositivo”, seleziona l’opzione the section called “Connettere un Raspberry Pi o altro dispositivo” per il tuo dispositivo e usa le opzioni del linguaggio Python per configurare il tuo dispositivo.

  > **Note**
  >
  > Assicurati di tenere aperta la finestra terminale che usi in quel tutorial perché la userai anche in questo.

- Un dispositivo in grado di eseguire AWS IoT Device SDK v2 per Python.

Questo tutorial mostra come connettere un dispositivo AWS IoT Core utilizzando esempi di codice Python, che richiedono un dispositivo relativamente potente. Se lavori con dispositivi con vincoli di risorse, questi esempi di codice potrebbero non funzionare su di essi. In tal caso, potresti avere più successo con il the section called “Utilizzando il SDK per dispositivi AWS IoT per Embedded C” tutorial.

- Hai ottenuto le informazioni richieste per la connessione al dispositivo

  Per connettere il dispositivo a AWS IoT, è necessario disporre di informazioni sul nome dell'oggetto, sul nome host e sul numero di porta.

  > **Note**
  >
  > Puoi anche utilizzare l'autenticazione personalizzata a cui connettere i dispositivi AWS IoT Core. I dati di connessione trasmessi alla funzione Lambda dell'autorizzatore dipendono dal protocollo utilizzato.

  - Nome dell'oggetto: il nome dell' AWS IoT oggetto a cui vuoi connetterti. Devi prima esserti registrato come AWS IoT dispositivo. Per ulteriori informazioni, consulta Gestione dei dispositivi con AWS IoT.
  - Nome host: il nome host per l'endpoint IoT specifico dell'account.
  - Numero di porta: il numero di porta a cui connettersi.

  Puoi usare il `configureEndpoint` metodo in AWS IoT Python SDK per configurare il nome host e il numero di porta.

  ```
  myAWSIoTMQTTClient.configureEndpoint("random.iot.region.amazonaws.com", 8883)
  ```

Prepara il tuo dispositivo per AWS IoT

In Guida introduttiva ai AWS IoT Core tutorial, hai preparato il tuo dispositivo e account AWS in modo che potessero comunicare. Questa sezione esamina gli aspetti di tale preparazione che si applicano alla connessione di qualsiasi dispositivo con AWS IoT Core.

Affinché un dispositivo si connetta a AWS IoT Core:

1. È necessario disporre di un Account AWS.

La procedura riportata di seguito Configurare Account AWS descrive come crearne uno Account AWS se non ne hai già uno.

2. In quell'account, devi avere le seguenti AWS IoT risorse definite per il dispositivo nella tua Account AWS regione.

   La procedura in Crea AWS IoT risorse descrive come creare queste risorse per il dispositivo nell'account Account AWS e regione.

   - Un certificato del dispositivo registrato con AWS IoT e attivato per autenticare il dispositivo.

     Il certificato viene spesso creato con un oggetto AWS IoT e collegato a esso. Sebbene un oggetto non sia necessario per la connessione di un dispositivo AWS IoT, rende disponibili AWS IoT funzionalità aggiuntive per il dispositivo.

   - Una politica allegata al certificato del dispositivo che lo autorizza a connettersi AWS IoT Core e a eseguire tutte le azioni desiderate.

3. Una connessione a Internet che può accedere all'account Account AWS degli endpoint del dispositivo.

   Gli endpoint del dispositivo sono descritti AWS IoT dati del dispositivo e endpoint di servizio e possono essere visualizzati nella pagina delle impostazioni della AWS IoT console.

4. Software di comunicazione SDKs fornito dal AWS IoT dispositivo. Questo tutorial utilizza AWS IoT Device SDK v2 per Python.

Esamina il protocollo MQTT

Prima di parlare dell'app di esempio, questo ti aiuta a capire il protocollo MQTT. Il protocollo MQTT offre alcuni vantaggi rispetto ad altri protocolli di comunicazione di rete, come HTTP, così da essere una scelta popolare per i dispositivi IoT. Questa sezione esamina gli aspetti chiave di MQTT applicabili a questo tutorial. Per informazioni sul confronto tra MQTT e HTTP, consulta Scelta di un protocollo applicativo per la comunicazione del dispositivo.

MQTT utilizza un modello di publish/subscribe comunicazione

Il protocollo MQTT utilizza un modello di publish/subscribe comunicazione con il suo host. Questo modello è diverso dal request/response modello utilizzato da HTTP. Con MQTT, i dispositivi stabiliscono una sessione con l'host identificata da un ID client univoco. Per inviare dati, i dispositivi pubblicano i messaggi identificati da argomenti al broker messaggi nell'host. Per ricevere messaggi

dal broker di messaggi, i dispositivi sottoscrivono agli argomenti inviando al broker messaggi i filtri argomento nelle richieste di sottoscrizione al broker di messaggi.

MQTT supporta sessioni persistenti

Il broker di messaggi riceve i messaggi dai dispositivi e pubblica i messaggi ai dispositivi che hanno effettuato una sottoscrizione. Con le sessioni persistenti che rimangono attive anche quando il dispositivo di avvio è disconnesso, i dispositivi possono recuperare i messaggi pubblicati mentre erano disconnessi. Sul lato dispositivo, MQTT supporta i livelli di qualità del servizio (QoS) che assicurano che l'host riceva i messaggi inviati dal dispositivo.

Revisione dell'app di esempio pubsub.py SDK per dispositivi

Questa sezione esamina l'app di esempio `pubsub.py` SDK per dispositivi AWS IoT v2 per Python utilizzata in questo tutorial. Qui esamineremo come si connette per AWS IoT Core pubblicare e sottoscrivere i messaggi MQTT. La sezione successiva presenta alcuni esercizi per aiutarti a scoprire come un dispositivo si connette e comunica. AWS IoT Core

L'app pubsub.py di esempio illustra questi aspetti di una connessione MQTT con: AWS IoT Core

- Protocolli di comunicazione
- Sessioni persistenti
- Qualità del servizio
- Pubblicazione di messaggi
- Abbonamento messaggi
- Disconnessione e riconnessione del dispositivo

Protocolli di comunicazione

L'esempio `pubsub.py` illustra una connessione MQTT utilizzando i protocolli MQTT e MQTT su WSS. La libreria AWS Common Runtime (AWS CRT) fornisce il supporto del protocollo di comunicazione di basso livello ed è inclusa in AWS IoT Device SDK v2 for Python.

MQTT

Le chiamate `pubsub.py` di esempio `mtls_from_path` (mostrate qui) in cui `mqtt_connection_builder`stabilire una connessione utilizzando il protocollo MQTT AWS IoT Core . `mtls_from_path`utilizza certificati X.509 e TLS v1.2 per autenticare il dispositivo. La libreria AWS CRT gestisce i dettagli di livello inferiore di tale connessione.

```
mqtt_connection = mqtt_connection_builder.mtls_from_path(
    endpoint=args.endpoint,
    cert_filepath=args.cert,
    pri_key_filepath=args.key,
    ca_filepath=args.ca_file,
    client_bootstrap=client_bootstrap,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

`endpoint`

L'endpoint Account AWS del tuo dispositivo IoT

Nell'app di esempio, il valore viene passato dalla riga di comando.

`cert_filepath`

Il percorso del certificato del file del dispositivo

Nell'app di esempio, il valore viene passato dalla riga di comando.

`pri_key_filepath`

Il percorso al file della chiave privata del dispositivo creata con il relativo file di certificato

Nell'app di esempio, il valore viene passato dalla riga di comando.

`ca_filepath`

Il percorso del file Root CA. Obbligatorio solo se il server MQTT utilizza un certificato che non è già presente nell'archivio.

Nell'app di esempio, il valore viene passato dalla riga di comando.

`client_bootstrap`

L'oggetto runtime comune che gestisce le attività di comunicazione socket

Nell'app di esempio, questo oggetto viene istanziato appena prima della chiamata a `mqtt_connection_builder.mtls_from_path`.

`on_connection_interrupted`, `on_connection_resumed`

Le funzioni di richiamata da richiamare quando la connessione del dispositivo viene interrotta e ripresa

`client_id`

L'ID che identifica il dispositivo in modo univoco nella Regione AWS

Nell'app di esempio, il valore viene passato dalla riga di comando.

`clean_session`

Se avviare una nuova sessione persistente o, se presente, riconnettersi a una esistente

`keep_alive_secs`

Il valore keep alive, in secondi, per inviare la richiesta `CONNECT`. Un ping verrà inviato automaticamente a questo intervallo. Se il server non riceve un ping dopo 1,5 volte questo valore, presuppone che la connessione sia stata persa.

MQTT su WSS

L'esempio `pubsub.py` chiama `websockets_with_default_aws_signing` (mostrato qui) in `mqtt_connection_builder` per stabilire una connessione con AWS IoT Core utilizzando il protocollo MQTT su WSS. `websockets_with_default_aws_signing` crea una connessione MQTT su WSS usando Firma V4 per autenticare il dispositivo.

```
mqtt_connection = mqtt_connection_builder.websockets_with_default_aws_signing(
    endpoint=args.endpoint,
    client_bootstrap=client_bootstrap,
    region=args.signing_region,
    credentials_provider=credentials_provider,
    websocket_proxy_options=proxy_options,
    ca_filepath=args.ca_file,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

`endpoint`

L'endpoint Account AWS del tuo dispositivo IoT

Nell'app di esempio, il valore viene passato dalla riga di comando.

`client_bootstrap`

L'oggetto runtime comune che gestisce le attività di comunicazione socket

Nell'app di esempio, questo oggetto viene istanziato appena prima della chiamata a `mqtt_connection_builder.websockets_with_default_aws_signing`.

`region`

La regione di AWS firma utilizzata dall'autenticazione Signature V4. Nello stato `pubsub.py`, passa il parametro inserito nella riga di comando.

Nell'app di esempio, il valore viene passato dalla riga di comando.

`credentials_provider`

Le AWS credenziali fornite da utilizzare per l'autenticazione

Nell'app di esempio, questo oggetto viene istanziato appena prima della chiamata a `mqtt_connection_builder.websockets_with_default_aws_signing`.

`websocket_proxy_options`

Opzioni proxy HTTP, se si utilizza un host proxy

Nell'app di esempio, questo valore viene inizializzato appena prima della chiamata a `mqtt_connection_builder.websockets_with_default_aws_signing`.

`ca_filepath`

Il percorso del file Root CA. Obbligatorio solo se il server MQTT utilizza un certificato che non è già presente nell'archivio.

Nell'app di esempio, il valore viene passato dalla riga di comando.

`on_connection_interrupted`, `on_connection_resumed`

Le funzioni di richiamata da richiamare quando la connessione del dispositivo viene interrotta e ripresa

`client_id`

L'ID che identifichi il dispositivo in modo univoco nella Regione AWS.

Nell'app di esempio, il valore viene passato dalla riga di comando.

`clean_session`

Se avviare una nuova sessione persistente o, se presente, riconnettersi a una esistente

`keep_alive_secs`

Il valore keep alive, in secondi, per inviare la richiesta `CONNECT`. Un ping verrà inviato automaticamente a questo intervallo. Se il server non riceve un ping dopo 1,5 volte questo valore, presuppone che la connessione venga persa.

HTTPS

Che dire di HTTPS? AWS IoT Core supporta dispositivi che pubblicano richieste HTTPS. Dal punto di vista della programmazione, i dispositivi inviano richieste HTTPS a AWS IoT Core come qualsiasi altra applicazione. Per un esempio di un programma Python che invia un messaggio HTTP da un dispositivo, consulta l'esempio di codice HTTPS utilizzando la libreria di Python `requests`. Questo esempio invia un messaggio all' AWS IoT Core utilizzo di HTTPS in modo che lo AWS IoT Core interpreti come un messaggio MQTT.

Sebbene AWS IoT Core supporti le richieste HTTPS dai dispositivi, assicuratevi di esaminare le informazioni Scelta di un protocollo applicativo per la comunicazione del dispositivo in merito in modo da poter prendere una decisione informata sul protocollo da utilizzare per le comunicazioni tra i dispositivi.

Sessioni persistenti

Nell'app di esempio, l'impostazione del parametro `clean_session` di `False` indica che la connessione deve essere persistente. In pratica, ciò significa che la connessione aperta da questa chiamata si riconnette a una sessione persistente esistente, se esiste. In caso contrario, crea e si connette a una nuova sessione persistente.

Con una sessione persistente, i messaggi inviati al dispositivo vengono archiviati dal broker di messaggi mentre il dispositivo non è connesso. Quando un dispositivo si riconnette a una sessione persistente, il broker di messaggi invia al dispositivo tutti i messaggi archiviati a cui è stato sottoscritto.

Senza una sessione persistente, il dispositivo non riceverà messaggi inviati mentre il dispositivo non è connesso. L'opzione da utilizzare dipende dall'applicazione e se i messaggi che si verificano mentre

un dispositivo non è connesso devono essere comunicati. Per ulteriori informazioni, consulta Sessioni persistenti MQTT.

Qualità del servizio

Quando il dispositivo pubblica e sottoscrive messaggi, è possibile impostare la qualità del servizio (QoS) preferita. AWS IoT supporta i livelli QoS 0 e 1 per le operazioni di pubblicazione e sottoscrizione. Per ulteriori informazioni sui livelli QoS in AWS IoT, vedere. Opzioni di qualità del servizio MQTT (QoS)

Il runtime AWS CRT per Python definisce queste costanti per i livelli QoS che supporta:

Livelli della qualità del servizio di Python

Livello QoS MQTT	Valore simbolico Python utilizzato da SDK	Description
QoS livello 0	`mqtt.QoS.AT_MOST_ONCE`	Verrà effettuato un solo tentativo di inviare il messaggio, che sia ricevuto o meno. Il messaggio potrebbe non venire inviato, ad esempio se il dispositivo non è connesso o si verifica un errore di rete.
QoS livello 1	`mqtt.QoS.AT_LEAST_ONCE`	Il messaggio viene inviato ripetutamente finché non si riceve un riconoscimento PUBACK.

Nell'app di esempio, le richieste di pubblicazione e sottoscrizione vengono effettuate con un livello QoS di 1 (`mqtt.QoS.AT_LEAST_ONCE`).

- QoS in fase di pubblicazione

  Quando un dispositivo pubblica un messaggio con QoS livello 1, invia il messaggio ripetutamente fino a quando non riceve una risposta PUBACK dal broker di messaggi. Se il dispositivo non è connesso, il messaggio viene messo in coda per essere inviato dopo la riconnessione.

- QoS in abbonamento

  Quando un dispositivo sottoscrive un messaggio con QoS livello 1, il broker di messaggi salva i messaggi a cui il dispositivo è sottoscritto fino a quando non possono essere inviati al dispositivo. Il broker di messaggi restituisce i messaggi fino a quando non riceve una risposta PUBACK dal dispositivo.

Pubblicazione di messaggi

Dopo aver stabilito con successo una connessione a AWS IoT Core, i dispositivi possono pubblicare messaggi. L'esempio `pubsub.py` fa ciò chiamando l'operazione `publish` dell'oggetto `mqtt_connection`.

```
mqtt_connection.publish(
    topic=args.topic,
    payload=message,
    qos=mqtt.QoS.AT_LEAST_ONCE
)
```

`topic`

Nome dell'argomento del messaggio che identifica il messaggio

Nell'app di esempio, ciò viene passato dalla riga di comando.

`payload`

Il payload del messaggio formattato come stringa (ad esempio, un documento JSON)

Nell'app di esempio, ciò viene passato dalla riga di comando.

Un documento JSON è un formato di payload comune e riconosciuto da altri AWS IoT servizi; tuttavia, il formato dei dati del payload del messaggio può essere qualsiasi cosa concordata tra editori e abbonati. Ttuttavia, altri servizi AWS IoT , riconoscono solo JSON e CBOR in alcuni casi per la maggior parte delle operazioni.

`qos`

Il livello QoS per questo messaggio

Abbonamento messaggi

Per ricevere messaggi da AWS IoT e altri servizi e dispositivi, i dispositivi si iscrivono a tali messaggi in base al nome dell'argomento. I dispositivi possono sottoscrivere singoli messaggi specificando un nome argomento e a un gruppo di messaggi specificando un filtro di argomenti che può includere caratteri jolly. L'esempio `pubsub.py` utilizza il codice mostrato qui per iscriversi ai messaggi e registrare le funzioni di callback per elaborare il messaggio dopo che è stato ricevuto.

```
subscribe_future, packet_id = mqtt_connection.subscribe(
    topic=args.topic,
    qos=mqtt.QoS.AT_LEAST_ONCE,
    callback=on_message_received
)
subscribe_result = subscribe_future.result()
```

`topic`

L'argomento a cui effettuare la sottoscrizione. Può trattarsi di un nome di argomento o di un filtro dell'argomento.

Nell'app di esempio, ciò viene passato dalla riga di comando.

`qos`

Se il broker di messaggi deve archiviare questi messaggi mentre il dispositivo è disconnesso.

Valore di `mqtt.QoS.AT_LEAST_ONCE` (QoS livello 1), richiede una sessione persistente da specificare (`clean_session=False`) quando viene creata la connessione.

`callback`

La funzione da chiamare per elaborare il messaggio sottoscritto.

La funzione `mqtt_connection.subscribe` restituisce un futuro e un ID pacchetto. Se la richiesta di sottoscrizione è stata avviata correttamente, l'ID del pacchetto restituito è maggiore di 0. Per assicurarsi che la sottoscrizione sia stata ricevuta e registrata dal broker di messaggi, è necessario attendere la restituzione del risultato dell'operazione asincrona, come illustrato nell'esempio del codice.

La funzione di callback

Il callback nei processi di esempio `pubsub.py` elabora i messaggi sottoscritti quando il dispositivo li riceve.

```
def on_message_received(topic, payload, **kwargs):
    print("Received message from topic '{}': {}".format(topic, payload))
    global received_count
    received_count += 1
    if received_count == args.count:
        received_all_event.set()
```

`topic`

L'argomento del messaggio

Questo è il nome dell'argomento specifico del messaggio ricevuto, anche se è stato sottoscritto un filtro dell'argomento.

`payload`

Payload del messaggio

Il formato è specifico per l'applicazione.

`kwargs`

Possibili argomenti aggiuntivi come descritto in `mqtt.Connection.subscribe`.

Nell'esempio `pubsub.py`, `on_message_received` visualizza solo l'argomento e il relativo payload. Conta inoltre i messaggi ricevuti per terminare il programma dopo il raggiungimento del limite.

La tua app valuterà l'argomento e il payload per determinare quali operazioni eseguire.

Disconnessione e riconnessione del dispositivo

L'esempio `pubsub.py` include funzioni di callback che vengono chiamate quando il dispositivo viene disconnesso e quando la connessione viene ristabilita. Le operazioni intraprese dal dispositivo su questi eventi sono specifiche dell'applicazione.

Quando un dispositivo si connette per la prima volta, deve sottoscrivere gli argomenti da ricevere. Se la sessione di un dispositivo è presente durante la riconnessione, le relative sottoscrizioni vengono ripristinate e tutti i messaggi memorizzati da tali sottoscrizioni vengono inviati al dispositivo dopo la riconnessione.

Se la sessione di un dispositivo non esiste più quando si riconnette, è necessario sottoscrivere nuovamente alle sottoscrizioni. Le sessioni persistenti hanno una durata limitata e possono scadere quando il dispositivo viene disconnesso per troppo tempo.

Connect il dispositivo e comunica con AWS IoT Core

Questa sezione presenta alcuni esercizi che consentono di esplorare diversi aspetti della connessione del dispositivo a AWS IoT Core. Per questi esercizi, utilizzerai il Client di test MQTT nella console AWS IoT per vedere cosa pubblica il dispositivo e pubblicare messaggi sul dispositivo. Questi esercizi utilizzano il `pubsub.py` di esempio dall'SDK del dispositivo AWS IoT v2 per Python e si basano sulla tua esperienza con i tutorial Tutorial introduttivi.

In questa sezione, dovrai:

- Iscriviti ai filtri per argomenti jolly
- Processo sottoscrizioni filtro dell'argomento
- Pubblica messaggi dal tuo dispositivo

Per questi esercizi, inizierai dal programma di esempio `pubsub.py`.

> **Note**
>
> Questi esercizi presuppongono che tu abbia completato i tutorial Tutorial introduttivi e abbia utilizzato la finestra del terminale per il dispositivo da quel tutorial.

Iscriviti ai filtri per argomenti jolly

In questo esercizio, modificherai la riga di comando utilizzata per chiamare `pubsub.py` per sottoscrivere un filtro dell'argomento con caratteri jolly ed elaborare i messaggi ricevuti in base all'argomento del messaggio.

Procedura dell'esercizio

Per questo esercizio, immagina che il dispositivo contenga un controllo della temperatura e un controllo della luce. Utilizza questi nomi di argomenti per identificare i messaggi che li riguardano.

1. Prima di iniziare l'esercizio, prova a eseguire questo comando dai tutorial Tutorial introduttivi sul tuo dispositivo per assicurarti che tutto sia pronto per l'esercizio.

```
cd ~/aws-iot-device-sdk-python-v2/samples
python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
```

Dovresti vedere lo stesso output che hai visto nel Tutorial sulle nozioni di base.

2. Per questo esercizio, modifica questi parametri della riga di comando.

Azione	Parametri della riga di comando	Effetto
aggiungi	--message ""	Configura `pubsub.py` solo per l'ascolto
aggiungi	--count 2	Termina il programma dopo aver ricevuto due messaggi
modifica	--topic `device/+/details`	Definisci il filtro degli argomenti a cui sottoscrivere

L'esecuzione di queste modifiche alla riga di comando iniziale si traduce in questa riga di comando. Inserisci questo comando nella finestra del terminale per il tuo dispositivo.

```
python3 pubsub.py --message "" --count 2 --topic device/+/details --ca_file
 ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
private.pem.key --endpoint your-iot-endpoint
```

Il programma dovrebbe mostrare qualcosa del genere:

```
Connecting to a3qexamplesffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-24d7cdcc-cc01-458c-8488-2d05849691e1'...
Connected!
Subscribing to topic 'device/+/details'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for all messages to be received...
```

Se vedi qualcosa di simile sul tuo terminale, il tuo dispositivo è pronto e ascolta i messaggi in cui i nomi degli argomenti iniziano con `device` e terminano con `/detail`. Quindi, è il momento di provare.

3. Di seguito sono riportati un paio di messaggi che il dispositivo potrebbe ricevere.

Nome dell'argomento	Payload del messaggio
`device/temp/details`	`{ "desiredTemp": 20, "currentTemp": 15 }`
`device/light/details`	`{ "desiredLight": 100, "currentLight": 50 }`

4. Utilizzando il client di test MQTT nella AWS IoT console, inviate i messaggi descritti nel passaggio precedente al dispositivo.

   a. Apri il client di test MQTT nella AWS IoT console.

   b. In Subscribe to a topic (Sottoscrivi un argomento) nel campo Subscription topic field (Argomento sottoscrizione), inserisci il filtro dell'argomento: **`device/+/details`**, quindi scegli Subscribe to topic (Sottoscrivi nell'argomento).

   c. Nella colonna Subscriptions (Sottoscrizioni) del client di test MQTT, seleziona device/+/details (dispositivo/+/dettagli).

   d. Per ciascuno degli argomenti della tabella precedente, esegui le operazioni seguenti nel client di test MQTT:

      1. In Publish (Pubblica), inserisci il valore dalla colonna Topic name (Nome argomento) nella tabella.
      2. Nel campo del payload del messaggio sotto il nome dell'argomento, inserisci il valore della colonna Message payload (Payload del messaggio) nella tabella.
      3. Guarda la finestra del terminale dove `pubsub.py` è in esecuzione e, nel client di test MQTT, scegli Publish to topic (Pubblicazione nell'argomento).

      Dovresti vedere che il messaggio è stato ricevuto da `pubsub.py` nella finestra del terminale.

Risultato dell'esercizio

Con questo `pubsub.py` sottoscritto ai messaggi utilizzando un filtro argomento jolly, li ha ricevuti e visualizzati nella finestra del terminale. Si noti come si è sottoscritto a un singolo filtro dell'argomento e la funzione di callback è stata chiamata per elaborare i messaggi con due argomenti distinti.

Processo sottoscrizioni filtro dell'argomento

Basandosi sull'esercizio precedente, modifica l'app di esempio `pubsub.py` per valutare gli argomenti dei messaggi ed elaborare i messaggi sottoscritti in base all'argomento.

Procedura dell'esercizio

Per valutare l'argomento del messaggio

1. Copia `pubsub.py` su `pubsub2.py`.
2. Apri `pubsub2.py` nell'editor di testo preferito o in IDE.
3. In `pubsub2.py`, trova la funzione `on_message_received`.
4. In `on_message_received`, inserisci il codice seguente dopo la riga che inizia con `print("Received message` e prima della riga che inizia con `global received_count`.

```
    topic_parsed = False
    if "/" in topic:
        parsed_topic = topic.split("/")
        if len(parsed_topic) == 3:
            # this topic has the correct format
            if (parsed_topic[0] == 'device') and (parsed_topic[2] == 'details'):
                # this is a topic we care about, so check the 2nd element
                if (parsed_topic[1] == 'temp'):
                    print("Received temperature request: {}".format(payload))
                    topic_parsed = True
                if (parsed_topic[1] == 'light'):
                    print("Received light request: {}".format(payload))
                    topic_parsed = True
    if not topic_parsed:
        print("Unrecognized message topic.")
```

5. Salva le modifiche ed esegui il programma modificato utilizzando questa riga di comando.

```
python3 pubsub2.py --message "" --count 2 --topic device/+/details --ca_file
 ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
private.pem.key --endpoint your-iot-endpoint
```

6. Nella AWS IoT console, aprite il client di test MQTT.
7. In Subscribe to a topic (Sottoscrivi un argomento) nel campo Subscription topic field (Argomento sottoscrizione), inserisci il filtro dell'argomento: **`device/+/details`**, quindi scegli Subscribe to topic (Sottoscrivi nell'argomento).
8. Nella colonna Subscriptions (Sottoscrizioni) del client di test MQTT, seleziona device/+/details (dispositivo/+/dettagli).
9. Per ciascuno degli argomenti di questa tabella, esegui le operazioni seguenti nel client di test MQTT:

Nome dell'argomento	Payload del messaggio
`device/temp/details`	`{ "desiredTemp": 20, "currentTemp": 15 }`
`device/light/details`	`{ "desiredLight": 100, "currentLight": 50 }`

   1. In Publish (Pubblica), inserisci il valore dalla colonna Topic name (Nome argomento) nella tabella.
   2. Nel campo del payload del messaggio sotto il nome dell'argomento, inserisci il valore della colonna Message payload (Payload del messaggio) nella tabella.
   3. Guarda la finestra del terminale dove `pubsub.py` è in esecuzione e, nel client di test MQTT, scegli Publish to topic (Pubblicazione nell'argomento).

   Dovresti vedere che il messaggio è stato ricevuto da `pubsub.py` nella finestra del terminale.

Dovrebbe essere visualizzato un elemento simile a quello nella finestra del terminale.

```
Connecting to a3qexamplesffp-ats.iot.us-west-2.amazonaws.com with client ID 'test-
af794be0-7542-45a0-b0af-0b0ea7474517'...
Connected!
```

```
Subscribing to topic 'device/+/details'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for all messages to be received...
Received message from topic 'device/light/details': b'{ "desiredLight": 100,
 "currentLight": 50 }'
Received light request: b'{ "desiredLight": 100, "currentLight": 50 }'
Received message from topic 'device/temp/details': b'{ "desiredTemp": 20,
 "currentTemp": 15 }'
Received temperature request: b'{ "desiredTemp": 20, "currentTemp": 15 }'
2 message(s) received.
Disconnecting...
Disconnected!
```

Risultato dell'esercizio

In questo esercizio è stato aggiunto il codice in modo che l'app di esempio riconosca ed elabori più messaggi nella funzione di callback. Con questo, il tuo dispositivo potrebbe ricevere messaggi e agire su di essi.

Un altro modo per consentire al tuo dispositivo di ricevere e processare diversi messaggi separatamente e assegnare ogni abbonamento alla propria funzione di callback.

Pubblica messaggi dal tuo dispositivo

È possibile utilizzare l'app di esempio pubsub.py per pubblicare messaggi dal dispositivo. Mentre pubblicherà i messaggi così com'è, i messaggi non possono essere letti come documenti JSON. Questo esercizio modifica l'app di esempio per poter pubblicare documenti JSON nel payload dei messaggi che possono essere letti da. AWS IoT Core

Procedura dell'esercizio

In questo esercizio, il seguente messaggio verrà inviato con l'argomento `device/data`.

```
{
    "timestamp": 1601048303,
    "sensorId": 28,
    "sensorData": [
        {
        "sensorName": "Wind speed",
        "sensorValue": 34.2211224
        }
    ]
}
```

Per preparare il client di test MQTT per monitorare i messaggi di questo esercizio

1. In Subscribe to a topic (Sottoscrivi un argomento) nel campo Subscription topic field (Argomento sottoscrizione), inserisci il filtro dell'argomento: **`device/data`**, quindi scegli Subscribe to topic (Sottoscrivi nell'argomento).
2. Nella colonna Subscriptions (Sottoscrizioni) del client di test MQTT, seleziona device/details (dispositivo/dati).
3. Tieni aperta la finestra del client di test MQTT per attendere i messaggi dal tuo dispositivo.

Per inviare documenti JSON con l'app di esempio pubsub.py

1. Sul dispositivo, copia `pubsub.py` su `pubsub3.py`.
2. Modifica `pubsub3.py` per cambiare il modo in cui formatta i messaggi pubblicati.
   a. Aprire `pubsub3.py` in un editor di testo.
   b. Individua la riga di codice:

      ```
      message = "{} [{}]".format(message_string, publish_count)
      ```

   c. Modificalo in:

      ```
      message = "{}".format(message_string)
      ```

   d. Individua la riga di codice:

      ```
      message_json = json.dumps(message)
      ```

   e. Modificalo in:

      ```
      message = "{}".json.dumps(json.loads(message))
      ```

   f. Salvare le modifiche.
3. Sul dispositivo, esegui questo comando per inviare il messaggio due volte.

   ```
   python3 pubsub3.py  --ca_file ~/certs/Amazon-root-CA-1.pem  --cert ~/certs/
   device.pem.crt  --key ~/certs/private.pem.key  --topic device/data  --count 2 --
   message '{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
    speed","sensorValue":34.2211224}]}'  --endpoint your-iot-endpoint
   ```

4. Nel client di test MQTT, verifica che abbia interpretato e formattato il documento JSON nel payload del messaggio, ad esempio:

Per impostazione predefinita, `pubsub3.py` effettua la sottoscrizione anche ai messaggi che invia. Dovresti vedere che ha ricevuto i messaggi nell'output dell'app. La finestra del terminale dovrebbe essere simile a questa.

```
Connecting to a3qEXAMPLEsffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-5cff18ae-1e92-4c38-a9d4-7b9771afc52f'...
Connected!
Subscribing to topic 'device/data'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 2 message(s)
Publishing message to topic 'device/data':
 {"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}
Received message from topic 'device/data':
 b'{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'
Publishing message to topic 'device/data':
 {"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}
Received message from topic 'device/data':
 b'{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'
2 message(s) received.
Disconnecting...
Disconnected!
```

Risultato dell'esercizio

In questo modo, il dispositivo può generare messaggi da inviare per AWS IoT Core testare la connettività di base e fornire messaggi da elaborare. AWS IoT Core Ad esempio, puoi utilizzare questa app per inviare dati di test dal tuo dispositivo per testare le azioni delle AWS IoT regole.

Rivedi i risultati

Gli esempi di questo tutorial ti hanno fornito un'esperienza pratica sulle nozioni di base con cui i dispositivi possono comunicare, AWS IoT Core una parte fondamentale della tua soluzione. AWS IoT Quando i dispositivi sono in grado di comunicare AWS IoT Core, possono trasmettere messaggi ai AWS servizi e ad altri dispositivi su cui possono agire. Allo stesso modo, AWS i servizi e gli altri dispositivi possono elaborare informazioni che generano l'invio di messaggi ai dispositivi dell'utente.

Quando sei pronto per AWS IoT Core approfondire, prova questi tutorial:

- the section called "Invio di una notifica Amazon SNS"
- the section called "Archiviazione dei dati del dispositivo in una tabella Dynamo DB"
- the section called "Formattazione di una notifica utilizzando una funzione AWS Lambda "

Tutorial: Utilizzo del SDK per dispositivi AWS IoT per Embedded C

Questa sezione descrive come eseguire SDK per dispositivi AWS IoT per Embedded C.

Procedure in questa sezione

- Passaggio 1: installare il SDK per dispositivi AWS IoT per Embedded C
- Fase 2: configurare l'app di esempio
- Fase 3: Crea ed esegui l'applicazione di esempio

Passaggio 1: installare il SDK per dispositivi AWS IoT per Embedded C

SDK per dispositivi AWS IoT per Embedded C È generalmente destinato a dispositivi con risorse limitate che richiedono un runtime ottimizzato in linguaggio C. È possibile utilizzare l'SDK su qualsiasi sistema operativo e ospitarlo su qualsiasi tipo di processore (ad esempio, MCUs e). MPUs Se disponi di più risorse di memoria ed elaborazione, ti consigliamo di utilizzare uno dei AWS IoT dispositivi e dispositivi mobili di ordine superiore SDKs (ad esempio, C++ JavaScript, Java e Python).

In generale, SDK per dispositivi AWS IoT per Embedded C è destinato a sistemi che utilizzano MCUs o di fascia bassa MPUs che eseguono sistemi operativi integrati. Per l'esempio di programmazione in questa sezione, presumiamo che il tuo dispositivo utilizzi Linux.

Example

1. Scaricalo sul SDK per dispositivi AWS IoT per Embedded C tuo dispositivo da GitHub.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-embedded-c.git --recurse-submodules
   ```

   Questo crea una directory denominata `aws-iot-device-sdk-embedded-c` nella directory corrente.
2. Passa a quella directory ed estrai la versione più recente. Consulta github. com/aws/aws-iot-device-sdk-embedded-C/tagsper il tag dell'ultima versione.

   ```
   cd aws-iot-device-sdk-embedded-c
   git checkout latest-release-tag
   ```
3. Installa OpenSSL versione 1.1.0 o successiva. Le librerie di sviluppo OpenSSL sono solitamente chiamate “libssl-dev” o “openssl-devel” quando installate tramite un gestore di pacchetti.

   ```
   sudo apt-get install libssl-dev
   ```

Fase 2: configurare l'app di esempio

SDK per dispositivi AWS IoT per Embedded C Include applicazioni di esempio da provare. Per semplicità, questo tutorial utilizza l'`mqtt_demo_mutual_auth`applicazione, che illustra come connettersi al broker di AWS IoT Core messaggi e iscriversi e pubblicare su argomenti MQTT.

1. Copia nella directory `build/bin/certificates` il certificato e la chiave privata che hai creato in Guida introduttiva ai AWS IoT Core tutorial.

   > **Note**
   >
   > I certificati emessi da una CA root e dei dispositivi sono soggetti a scadenza o revoca. Se i certificati scadono o vengono revocati, è necessario copiare sul dispositivo un nuovo certificato emesso da una CA o una chiave privata e un certificato del dispositivo.

2. È necessario configurare l'esempio con l' AWS IoT Core endpoint personale, la chiave privata, il certificato e il certificato CA principale. Passa alla directory `aws-iot-device-sdk-embedded-c/demos/mqtt/mqtt_demo_mutual_auth`.

   Se lo hai AWS CLI installato, puoi usare questo comando per trovare l'URL dell'endpoint del tuo account.

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

   Se non lo hai AWS CLI installato, apri la AWS IoT console. Nel pannello di navigazione scegliere Manage (Gestisci), quindi Things (Oggetti). Seleziona l'oggetto IoT per il dispositivo, quindi seleziona Interact (Interagisci). L'endpoint viene visualizzato nella sezione HTTPS della pagina dei dettagli degli oggetti.

3. Apri il file `demo_config.h` e aggiorna i valori per gli elementi seguenti:

   AWS_IOT_ENDPOINT

   Endpoint personale.

   CLIENT_CERT_PATH

   Il percorso del file del certificato, ad esempio `certificates/device.pem.crt`".

   CLIENT_PRIVATE_KEY_PATH

   Il nome del file della chiave privata, ad esempio `certificates/private.pem.key`.

   Per esempio:

   ```
   // Get from demo_config.h
   // =================================================
   #define AWS_IOT_ENDPOINT               "my-endpoint-ats.iot.us-
   east-1.amazonaws.com"
   #define AWS_MQTT_PORT                  8883
   #define CLIENT_IDENTIFIER              "testclient"
   #define ROOT_CA_CERT_PATH              "certificates/AmazonRootCA1.crt"
   #define CLIENT_CERT_PATH               "certificates/my-device-cert.pem.crt"
   #define CLIENT_PRIVATE_KEY_PATH        "certificates/my-device-private-key.pem.key"
   // =================================================
   ```

4. Controlla se hai CMake installato sul tuo dispositivo usando questo comando.

```
cmake --version
```

Se visualizzi le informazioni sulla versione per il compilatore, puoi passare alla sezione successiva.

Se ricevi un errore o non vedi alcuna informazione, dovrai installare il pacchetto CMake utilizzando questo comando.

```
sudo apt-get install cmake
```

Esegui nuovamente il cmake --version comando e conferma che è CMake stato installato e che sei pronto per continuare.

5. Controlla di aver installato gli strumenti di sviluppo sul dispositivo usando questo comando.

```
gcc --version
```

Se visualizzi le informazioni sulla versione per il compilatore, puoi passare alla sezione successiva.

Se ricevi un errore o non vedi alcuna informazione del compilatore, dovrai installare il pacchetto `build-essential` usando questo comando.

```
sudo apt-get install build-essential
```

Esegui nuovamente il comando gcc --version e conferma che gli strumenti di compilazione sono stati installati e di essere pronto a continuare.

Fase 3: Crea ed esegui l'applicazione di esempio

Questa procedura spiega come generare l'`mqtt_demo_mutual_auth`applicazione sul dispositivo e collegarla alla AWS IoT console utilizzando il. SDK per dispositivi AWS IoT per Embedded C

Per eseguire le applicazioni di esempio SDK per dispositivi AWS IoT per Embedded C

1. Passa a `aws-iot-device-sdk-embedded-c` e crea una directory di compilazione.

```
mkdir build && cd build
```

2. Immettete il seguente CMake comando per generare i Makefile necessari per la compilazione.

```
cmake ..
```

3. Inserisci il comando seguente per creare l'app del file eseguibile.

```
make
```

4. Esegui l'app `mqtt_demo_mutual_auth` con questo comando.

```
cd bin
./mqtt_demo_mutual_auth
```

Verrà visualizzato un output simile al seguente:

```
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:584] Establishing a TLS session to a2zk5tjv9x07ct-ats.iot.us-west-2.amazonaws.com:8883.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1264] Creating an MQTT connection to a2zk5tjv9x07ct-ats.iot.us-west-2.amazonaws.com.
[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=2.
[INFO] [MQTT] [core_mqtt_serializer.c:970] CONNACK session present bit not set.
[INFO] [MQTT] [core_mqtt_serializer.c:912] Connection accepted.
[INFO] [MQTT] [core_mqtt.c:1526] Received MQTT CONNACK successfully from broker.
[INFO] [MQTT] [core_mqtt.c:1792] MQTT connection established with the broker.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1033] MQTT connection successfully established with broker.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1296] A clean MQTT connection is established. Cleaning up all the stored outgoing publishes.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1314] Subscribing to the MQTT topic testclient/example/topic.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1097] SUBSCRIBE sent for topic testclient/example/topic to broker.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=3.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:921] Subscribed to the topic testclient/example/topic. with maximum QoS 1.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1358] Sending Publish to the MQTT topic testclient/example/topic.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1195] PUBLISH sent for topic testclient/example/topic to broker with packet ID 2.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=2.
[INFO] [MQTT] [core_mqtt.c:1126] Ack packet deserialized with result: MQTTSuccess.
[INFO] [MQTT] [core_mqtt.c:1139] State record updated. New state=MQTTPublishDone.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:946] PUBACK received for packet id 2.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:672] Cleaned up outgoing publish packet with packet id 2.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=40.
[INFO] [MQTT] [core_mqtt.c:1015] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
```

Il dispositivo è ora connesso AWS IoT tramite SDK per dispositivi AWS IoT per Embedded C.

È inoltre possibile utilizzare la AWS IoT console per visualizzare i messaggi MQTT pubblicati dall'app di esempio. Per informazioni su come utilizzare il client MQTT nella console AWS IoT, consulta the section called "Visualizza i messaggi MQTT con il AWS IoT client MQTT".

Creazione di AWS IoT regole per indirizzare i dati del dispositivo ad altri servizi

Questi tutorial mostrano come creare e testare AWS IoT regole utilizzando alcune delle azioni più comuni.

AWS IoT le regole inviano dati dai dispositivi ad altri AWS servizi. Ascoltano messaggi MQTT specifici, formattano i dati nei payload dei messaggi e inviano il risultato ad altri servizi AWS .

Ti suggeriamo di provarli nell'ordine in cui vengono mostrati qui, anche se il tuo obiettivo è creare una regola che utilizzi una funzione Lambda o qualcosa di più complesso. I tutorial sono presentati in ordine dal più semplice al più complesso. Presentano nuovi concetti in modo incrementale per aiutarti ad apprendere i concetti che è possibile utilizzare per creare le operazioni delle regole che non dispongono di un tutorial specifico.

> **Note**
>
> AWS IoT le regole ti aiutano a inviare i dati dai tuoi dispositivi IoT ad altri AWS servizi. Per farlo correttamente, tuttavia, è necessaria una conoscenza operativa degli altri servizi in cui si desidera inviare i dati. Sebbene questi tutorial forniscano le informazioni necessarie per completare le attività, potrebbe essere utile ottenere ulteriori informazioni sui servizi a cui si desidera inviare i dati prima di utilizzarli nella soluzione. Una spiegazione dettagliata degli altri AWS servizi non rientra nell'ambito di questi tutorial.

Panoramica dello scenario tutorial

Lo scenario di questi tutorial consiste in un dispositivo con sensore meteo che pubblica periodicamente i suoi dati. In questo sistema immaginario sono presenti molti dispositivi sensori di questo tipo. I tutorial in questa sezione, tuttavia, si concentrano su un singolo dispositivo mentre mostrano come è possibile ospitare più sensori.

I tutorial di questa sezione mostrano come utilizzare AWS IoT le regole per eseguire le seguenti attività con questo sistema immaginario di dispositivi con sensori meteorologici.

- Tutorial: ripubblicazione di un messaggio MQTT

  Questo tutorial mostra come ripubblicare un messaggio MQTT ricevuto dai sensori meteorologici come messaggio contenente solo l'ID del sensore e il valore della temperatura. Utilizza solo i

servizi AWS IoT Core e dimostra una semplice query SQL e come utilizzare il client MQTT per testare la regola.

- Tutorial: Invio di una notifica Amazon SNS

  Questo tutorial mostra come inviare un messaggio SNS quando un valore di un dispositivo con sensore meteorologico supera un valore specifico. Si basa sui concetti presentati nel tutorial precedente e spiega come lavorare con un altro AWS servizio, Amazon Simple Notification Service (Amazon SNS).

  Se non hai mai utilizzato Amazon SNS, consulta le Nozioni di base prima di iniziare questo tutorial.

- Tutorial: Archiviazione dei dati del dispositivo in una tabella DynamoDB

  Questo tutorial mostra come memorizzare i dati dai dispositivi con sensori meteorologici in una tabella di database. Utilizza l'istruzione di query della regola e i modelli di sostituzione per formattare i dati dei messaggi per il servizio di destinazione Amazon DynamoDB.

  Se non hai mai usato DynamoDB, consulta le Nozioni di base prima di iniziare questo tutorial.

- Tutorial: Formattazione di una notifica utilizzando una funzione AWS Lambda

  Questo tutorial mostra come chiamare una funzione Lambda per riformattare i dati del dispositivo e quindi inviarli come messaggio di testo. Aggiunge uno script Python e funzioni AWS SDK in una AWS Lambdafunzione per formattare con i dati del payload del messaggio dai dispositivi dei sensori meteorologici e inviare un messaggio di testo.

  Se non hai mai usato Lambda, consulta le Nozioni di base prima di iniziare questo tutorial.

AWS IoT panoramica delle regole

Tutti questi tutorial creano AWS IoT regole.

Per inviare i dati da un dispositivo a un altro AWS servizio, una AWS IoT regola utilizza:

- Un'istruzione di query della regola costituita da:
  - Una clausola SQL SELECT che seleziona e formatta i dati dal payload del messaggio
  - Un filtro argomento (l'oggetto FROM nell'istruzione query della regola) che identifica i messaggi da utilizzare
  - Un'istruzione condizionale facoltativa (una clausola SQL WHERE) che specifica condizioni specifiche su cui agire

- Almeno un'operazione della regola

I dispositivi pubblicano i messaggi MQTT negli argomenti. Il filtro argomento nell'istruzione SQL SELECT identifica gli argomenti MQTT a cui applicare la regola. I campi specificati nell'istruzione SQL SELECT formattano i dati del payload dei messaggi MQTT in arrivo per l'utilizzo da parte delle azioni della regola. Per un elenco completo delle operazioni delle regole, consulta Operazioni delle regole AWS IoT.

Tutorial in questa sezione

- Tutorial: ripubblicazione di un messaggio MQTT
- Tutorial: Invio di una notifica Amazon SNS
- Tutorial: Archiviazione dei dati del dispositivo in una tabella DynamoDB
- Tutorial: Formattazione di una notifica utilizzando una funzione AWS Lambda

Tutorial: ripubblicazione di un messaggio MQTT

Questo tutorial dimostra come creare una AWS IoT regola che pubblichi un messaggio MQTT quando viene ricevuto un messaggio MQTT specificato. Il payload del messaggio in arrivo può essere modificato dalla regola prima della pubblicazione. In questo modo è possibile creare messaggi su misura per applicazioni specifiche senza la necessità di modificare il dispositivo o il firmware. È inoltre possibile utilizzare l'aspetto filtrante di una regola per pubblicare messaggi solo quando viene soddisfatta una condizione specifica.

I messaggi ripubblicati in base a una regola agiscono come messaggi inviati da qualsiasi altro dispositivo o client. AWS IoT I dispositivi possono sottoscrivere i messaggi ripubblicati nello stesso modo in cui possono iscriversi a qualsiasi altro argomento del messaggio MQTT.

Cosa imparerai in questo tutorial:

- Come utilizzare semplici query SQL e funzioni in un'istruzione di query della regola
- Come usare il client MQTT per testare una regola AWS IoT

Questo tutorial dura circa 30 minuti.

In questo tutorial, dovrai:

- Consulta gli argomenti e AWS IoT le regole di MQTT

- Fase 1: Creare una AWS IoT regola per ripubblicare un messaggio MQTT
- Fase 2. Test della nuova regola
- Fase 3: Esamina i risultati e i passaggi successivi

Prima di iniziare questo tutorial, assicurati di disporre di:

- Configurare Account AWS

  Avrai bisogno della tua Account AWS AWS IoT console per completare questo tutorial.
- Aver rivisto Visualizza i messaggi MQTT con il AWS IoT client MQTT

  Assicurati di poter usare il client MQTT per sottoscrivere e pubblicare un argomento. Utilizzerai il client MQTT per testare la nuova regola in questa procedura.

Consulta gli argomenti e AWS IoT le regole di MQTT

Prima di parlare AWS IoT delle regole, è utile comprendere il protocollo MQTT. Nelle soluzioni IoT, il protocollo MQTT offre alcuni vantaggi rispetto ad altri protocolli di comunicazione di rete, come HTTP, così da renderlo una scelta popolare per l'uso da parte dei dispositivi IoT. Questa sezione esamina gli aspetti chiave di MQTT applicati a questo tutorial. Per informazioni sul confronto tra MQTT e HTTP, consulta Scelta di un protocollo applicativo per la comunicazione del dispositivo.

Protocollo MQTT

Il protocollo MQTT utilizza un modello di publish/subscribe comunicazione con il suo host. Per inviare dati, i dispositivi pubblicano messaggi identificati da argomenti nel broker di AWS IoT messaggi. Per ricevere messaggi dal broker di messaggi, i dispositivi sottoscrivono gli argomenti che riceveranno inviando al broker messaggi filtri argomento nelle richieste di sottoscrizione. Il motore di regole AWS IoT riceve i messaggi MQTT dal broker di messaggi.

AWS IoT regole

AWS IoT le regole sono costituite da un'istruzione di interrogazione delle regole e da una o più azioni relative alle regole. Quando il motore di regole AWS IoT riceve un messaggio MQTT, questi elementi agiscono sul messaggio come segue.

- Istruzione query della regola

L'istruzione query della regola descrive gli argomenti MQTT da utilizzare, interpreta i dati del payload del messaggio e formatta i dati come descritto da un'istruzione SQL simile alle istruzioni utilizzate dai database SQL più diffusi. Il risultato dell'istruzione della query sono i dati inviati alle azioni della regola.

- Operazione delle regole

  Ogni azione contenuta in una regola agisce sui dati risultanti dall'istruzione di interrogazione della regola. AWS IoT supporta molte azioni relative alle regole. In questo tutorial, tuttavia, ti concentrerai sull'operazione della regola Ripubblica, che pubblica il risultato dell'istruzione query come messaggio MQTT con un argomento specifico.

Fase 1: Creare una AWS IoT regola per ripubblicare un messaggio MQTT

La AWS IoT regola che creerai in questo tutorial riguarda gli argomenti `device/`*`device_id`*`/data` MQTT dove si *`device_id`* trova l'ID del dispositivo che ha inviato il messaggio. Questi argomenti sono descritti da un filtro argomenti come `device/+/data`, dove + è un carattere jolly che corrisponde a qualsiasi stringa tra i due caratteri di barra in avanti.

Quando la regola riceve un messaggio da un argomento corrispondente, estrae i valori `device_id` e `temperature` e li ripubblica come nuovo messaggio MQTT con l'argomento `device/data/temp`.

Ad esempio, il payload di un messaggio MQTT con l'argomento `device/22/data` è simile a questo:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

La regola prende il valore `temperature` dal payload del messaggio e il valore `device_id` dall'argomento e li ripubblica come messaggio MQTT con l'argomento `device/data/temp` e un payload del messaggio simile al seguente:

```
{
  "device_id": "22",
```

```
  "temperature": 28
}
```

Con questa regola, i dispositivi che richiedono solo l'ID del dispositivo e i dati di temperatura si sottoscrivono all'argomento `device/data/temp` per ricevere solo tali informazioni.

Creazione di una regola che ripubblica un messaggio MQTT

1. Apri l'hub Rules della AWS IoT console.
2. In Rules (Regole), scegli Create (Crea) e inizia a creare la nuova regola.
3. Nella parte superiore di Create a rule (Crea una regola):

   a. In Name (Nome), inserisci il nome della regola. Per questo tutorial, rinominala **`republish_temp`**.

      Ricorda che un nome di regola deve essere univoco all'interno dell'account e della regione e che non può contenere spazi. Abbiamo usato un carattere di sottolineatura in questo nome per separare le due parole nel nome della regola.

   b. In Description (Descrizione), descrivi la regola.

      Una descrizione significativa ti aiuta a ricordare cosa fa questa regola e perché l'hai creata. La descrizione può essere lunga quanto necessario, quindi sii il più dettagliato possibile.
4. In Rule query statement (Istruzione query regola) di Create a rule (Crea una regola):

   a. In Using SQL version (Uso della versione SQL), seleziona **`2016-03-23`**.

   b. Nella casella di modifica Rule query statement (Istruzione query regola), inserisci l'istruzione:

      ```
      SELECT topic(2) as device_id, temperature FROM 'device/+/data'
      ```

      Questa istruzione:

      - Ascolta i messaggi MQTT con un argomento che corrisponde al filtro argomenti `device/+/data`.
      - Seleziona il secondo elemento dalla stringa dell'argomento e lo assegna al campo `device_id`.
      - Seleziona il valore del campo `temperature` dal payload del messaggio e lo assegna al campo `temperature`.

5. In Set one or more actions (Imposta una o più operazioni):

   a. Per aprire l'elenco delle azioni della regola per questa regola, scegli Add action (Aggiungi operazione).

   b. In Seleziona un'azione, scegli Ripubblica un messaggio su un AWS IoT argomento.

   c. Nella parte inferiore dell'elenco delle azioni, scegli Configure action (Configura operazione) per aprire la pagina di configurazione dell'azione selezionata.

6. In Configure action (Configura operazione):

   a. In Topic (Argomento), inserisci **`device/data/temp`**. Questo è l'argomento MQTT del messaggio che questa regola pubblicherà.

   b. In Quality of Service (Qualità del servizio), scegli 0 - Il messaggio viene recapitato zero o più volte.

   c. In Scegli o crea un ruolo a cui concedere AWS IoT l'accesso per eseguire questa azione:

      i. Selezionare Crea ruolo. Si aprirà la finestra di dialogo Create a new role (Crea un nuovo ruolo).

      ii. Inserisci un nome che descrive il nuovo ruolo. In questo tutorial, utilizza **`republish_role`**.

      Quando si crea un nuovo ruolo, le policy corrette per eseguire l'operazione della regola vengono create e associate al nuovo ruolo. Se si modifica l'argomento dell'operazione della regola o si utilizza questo ruolo in un'altra operazione della regola, è necessario aggiornare la policy per tale ruolo per autorizzare il nuovo argomento o operazione. Per aggiornare un ruolo esistente, scegli Update role (Aggiorna ruolo) in questa sezione.

      iii. Scegli Create role (Crea ruolo) per creare il ruolo e chiudere la finestra di dialogo.

   d. Scegli Add action (Aggiungi operazione) per aggiungere l'operazione alla regola e tornare alla pagina Create a rule (Crea una regola).

7. L'azione Ripubblica un messaggio su un AWS IoT argomento è ora elencata in Imposta una o più azioni.

   Nel titolo della nuova azione, sotto Republish a message to an AWS IoT topic (Ripubblicazione di un messaggio in un argomento di Amazon IoT), è possibile visualizzare l'argomento in cui verrà pubblicata l'operazione di ripubblicazione.

   Questa è l'unica operazione della regola che aggiungerai a questa regola.

8. In Create a rule (Crea una regola), scorri verso il basso e seleziona Create rule (Crea regola) per creare la regola e completare questo passaggio.

Fase 2. Test della nuova regola

Per testare la nuova regola, utilizza il client MQTT per pubblicare e sottoscrivere i messaggi MQTT utilizzati da questa regola.

Apri il Client MQTT nella console AWS IoT in una nuova finestra. In questo modo è possibile modificare la regola senza perdere la configurazione del client MQTT. Il client MQTT non conserva sottoscrizioni o registri dei messaggi se si lascia per andare a un'altra pagina della console.

Utilizzare il client MQTT per testare la regola

1. Nel Client MQTT nella console AWS IoT, effettua la sottoscrizione agli argomenti di input, in questo caso, `device/+/data`.

   a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni) scegli Subscribe to a topic (Effettua la sottoscrizione a un argomento).

   b. In Subscription topic (Argomento sottoscrizione), inserisci l'argomento del filtro dell'argomento di input, **`device/+/data`**.

   c. Lascia gli altri campi ai valori predefiniti.

   d. Scegli Subscribe to topic (Effettua sottoscrizione all'argomento).

      Nella colonna Subscriptions (Sottoscrizioni), sotto Publish to a topic (Pubblicaz in un argomento), viene visualizzato **`device/+/data`**.

2. Effettua la sottoscrizione all'argomento che verrà pubblicato dalla regola: `device/data/temp`.

   a. Sotto Subscriptions (Sottoscrizioni), scegli di nuovo Subscribe to a topic (Sottoscrizione a un argomento), e in Subscription topic (Argomento sottoscrizione) inserisci l'argomento del messaggio ripubblicato, **`device/data/temp`**.

   b. Lascia gli altri campi ai valori predefiniti.

   c. Scegli Subscribe to topic (Effettua sottoscrizione all'argomento).

      Nella colonna Subscriptions (Sottoscrizioni), sotto device/+/data (dispositivo/+/dati), **`device/data/temp`** viene visualizzato.

3. Pubblicazione di un messaggio nell'argomento di input con un ID dispositivo specifico, **`device/22/data`**. Non è possibile pubblicare su argomenti MQTT che contengono caratteri jolly.

   a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni) scegli Publish to topic (Pubblica in un argomento).

   b. Nel campo Publish (Pubblica), inserisci il nome dell'argomento di input, **`device/22/data`**.

   c. Copia i dati di esempio mostrati qui e, nella casella di modifica sotto il nome dell'argomento, incolla i dati di esempio.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. Per inviare il messaggio MQTT, scegli Publish to topic (Pubblica in un argomento).

4. Esamina i messaggi che sono stati inviati.

   a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni), accanto ai due argomenti a cui hai effettuato la sottoscrizione in precedenza, è presente un punto verde.

      I punti verdi indicano che uno o più nuovi messaggi sono stati ricevuti dall'ultima volta che li hai guardati.

   b. Sotto Subscriptions (Sottoscrizioni), scegli device/+/data (dispositivo/+/dati) per verificare che il payload del messaggio corrisponda a quello che hai appena pubblicato e assomigli a questo:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
```

```
}
```

c. In Abbonamenti, scegli di device/data/tempverificare che il payload del messaggio ripubblicato abbia il seguente aspetto:

```
{
  "device_id": "22",
  "temperature": 28
}
```

Nota che il valore `device_id` è una stringa tra virgolette e il valore `temperature` è numerico. Questo perché la funzione `topic()` ha estratto la stringa dal nome dell'argomento del messaggio di input mentre il valore `temperature` utilizza il valore numerico dal payload del messaggio di input.

Se desideri rendere il valore `device_id` un valore numerico, sostituisci `topic(2)`nell'istruzione query della regola con:

```
cast(topic(2) AS DECIMAL)
```

Nota che la trasformazione del valore `topic(2)` in un valore numerico funzionerà solo se la parte dell'argomento contiene solo caratteri numerici.

5. Se vedi che sull'device/data/tempargomento è stato pubblicato il messaggio corretto, allora la tua regola ha funzionato. Scopri di più sull'azione Ripubblica regola nella sezione successiva.

   Se non vedi che il messaggio corretto è stato pubblicato sul dispositivo/+/data o device/data/tempsugli argomenti, consulta i suggerimenti per la risoluzione dei problemi.

Risoluzione dei problemi della regola di ripubblicazione dei messaggi

Ecco alcune cose da controllare nel caso in cui non vedi i risultati che ti aspetti.

- Hai un banner di errore

  Se viene visualizzato un errore quando è stato pubblicato il messaggio di input, correggilo prima. I seguenti passaggi potrebbero aiutarti a correggere l'errore.
- Il messaggio di input non viene visualizzato nel client MQTT

Ogni volta che pubblichi il messaggio di input nell'argomento `device/22/data`, tale messaggio dovrebbe essere visualizzato nel client MQTT se hai effettuato la sottoscrizione al filtro argomento `device/+/data` come descritto nella procedura.

Controlli

- Controlla il filtro degli argomenti a cui hai effettuato la sottoscrizione

  Se hai effettuato la sottoscrizione all'argomento del messaggio di input come descritto nella procedura, visualizzerai una copia del messaggio di input ogni volta che lo pubblichi.

  Se il messaggio non viene visualizzato, controlla il nome dell'argomento sottoscritto e confrontalo con l'argomento in cui è stato pubblicato. I nomi degli argomenti fanno distinzione tra maiuscole e minuscole e l'argomento a cui è stato sottoscritto deve essere identico all'argomento in cui hai pubblicato il payload dei messaggi.
- Controlla la funzione di pubblicazione dei messaggi

  Nel client MQTT, sotto Subscriptions (Sottoscrizioni), scegli device/+/data (dispositivo/+/dati), controlla l'argomento del messaggio di pubblicazione e quindi scegli Publish to topic (Pubblica in un argomento). Dovresti vedere il payload del messaggio dalla casella di modifica sotto l'argomento visualizzato nell'elenco dei messaggi.

- Il messaggio ripubblicato non viene visualizzato nel client MQTT

  Affinché la regola funzioni, deve disporre della policy corretta che la autorizzi a ricevere e ripubblicare un messaggio e deve ricevere il messaggio.

  Controlli

  - Controlla il Regione AWS tuo client MQTT e la regola che hai creato

    La console in cui si esegue il client MQTT deve trovarsi nella stessa regione AWS della regola creata.
  - Controlla l'argomento del messaggio di input nell'istruzione query della regola

    Affinché la regola funzioni, è necessario che venga visualizzato un messaggio con il nome dell'argomento corrispondente al filtro dell'argomento nella clausola FROM dell'istruzione query della regola.

Controlla l'ortografia del filtro argomento nell'istruzione query regola con quella dell'argomento nel client MQTT. I nomi degli argomenti fanno distinzione tra maiuscole e minuscole e l'argomento del messaggio deve corrispondere al filtro argomento nell'istruzione query della regola.

- Controllare il contenuto del payload del messaggio di input

  Affinché la regola funzioni, è necessario trovare il campo dati nel payload del messaggio dichiarato nell'istruzione SELECT.

  Controlla l'ortografia del campo `temperature` nell'istruzione query della regola con quella del payload del messaggio nel client MQTT. I nomi dei campi fanno distinzione tra maiuscole e minuscole e il campo `temperature`nell'istruzione query della regola deve essere identico a quello del campo `temperature` nel payload del messaggio.

  Assicurati che il documento JSON nel payload del messaggio sia formattato correttamente. Se il JSON presenta errori, come una virgola mancante, la regola non sarà in grado di leggerlo.

- Controlla l'argomento del messaggio ripubblicato nell'operazione della regola

  L'argomento a cui l'operazione Ripubblica regola pubblica il nuovo messaggio deve corrispondere all'argomento a cui è stato sottoscritto nel client MQTT.

  Apri la regola creata nella console e controlla l'argomento in cui l'operazione della regola ripubblicherà il messaggio.

- Controlla il ruolo utilizzato dalla regola

  L'operazione della regola deve disporre dell'autorizzazione per ricevere l'argomento originale e pubblicare il nuovo argomento.

  Le policy che autorizzano la regola a ricevere i dati dei messaggi e a ripubblicarli sono specifiche degli argomenti utilizzati. Se si modifica l'argomento utilizzato per ripubblicare i dati del messaggio, è necessario aggiornare il ruolo dell'operazione della regola per aggiornare la policy in modo che corrisponda all'argomento corrente.

  Se si sospetta che questo sia il problema, modificare l'operazione Ripubblica regola e creare un nuovo ruolo. I nuovi ruoli creati dall'operazione della regola ricevono le autorizzazioni necessarie per eseguire queste operazioni.

Fase 3: Esamina i risultati e i passaggi successivi

In questo tutorial

- È stata utilizzata una query SQL semplice e un paio di funzioni in un'istruzione di query della regola per produrre un nuovo messaggio MQTT.
- Hai creato una regola che ha ripubblicato il nuovo messaggio.
- Hai usato il client MQTT per testare la tua AWS IoT regola.

Fasi successive

Dopo aver ripubblicato alcuni messaggi con questa regola, provare a sperimentarla per verificare come la modifica di alcuni aspetti del tutorial influisca sul messaggio ripubblicato. Ecco alcune idee per iniziare.

- `device_id`Modificate l'argomento del messaggio di input e osservate l'effetto nel payload del messaggio ripubblicato.
- Modifica i campi selezionati nell'istruzione query della regola e osserva l'effetto nel payload del messaggio ripubblicato.
- Prova il prossimo tutorial di questa serie e scopri come Tutorial: Invio di una notifica Amazon SNS.

L'operazione Ripubblica regola utilizzata in questo tutorial consente inoltre di eseguire il debug delle istruzioni di query della regola. Ad esempio, è possibile aggiungere questa operazione a una regola per verificare come l'istruzione della regola query formatta i dati utilizzati dalle relative operazioni delle regole.

Tutorial: Invio di una notifica Amazon SNS

Questo tutorial dimostra come creare una AWS IoT regola che invii i dati dei messaggi MQTT a un argomento di Amazon SNS in modo che possano essere inviati come messaggio di testo SMS.

In questo tutorial potrai creare una regola per l'invio dei dati dei messaggi da un sensore meteo a tutti i sottoscrittori di un argomento Amazon SNS, ogni volta che la temperatura supera il valore impostato nella regola. La regola rileva quando la temperatura segnalata supera il valore impostato dalla regola e quindi crea un nuovo payload del messaggio che include solo l'ID del dispositivo, la temperatura segnalata e il limite di temperatura superato. La regola invia il nuovo payload del messaggio come documento JSON a un argomento SNS, che notifica a tutti gli abbonati l'argomento SNS.

Cosa imparerai in questo tutorial:

- Come creare e testare una notifica Amazon SNS
- Come richiamare una notifica Amazon SNS da una regola AWS IoT
- Come utilizzare semplici query SQL e funzioni in un'istruzione di query della regola
- Come utilizzare il client MQTT per testare una regola AWS IoT

Questo tutorial dura circa 30 minuti.

In questo tutorial, dovrai:

- Fase 1: creare un argomento Amazon SNS che invii un messaggio di testo SMS
- Passaggio 2: crea una AWS IoT regola per inviare il messaggio di testo
- Fase 3: Verifica la AWS IoT regola e la notifica di Amazon SNS
- Fase 4: Esamina i risultati e i passaggi successivi

Prima di iniziare questo tutorial, assicurati di disporre di:

- Configurare Account AWS

  Avrai bisogno della tua Account AWS AWS IoT console per completare questo tutorial.
- Aver rivisto Visualizza i messaggi MQTT con il AWS IoT client MQTT

  Assicurati di poter usare il client MQTT per sottoscrivere e pubblicare un argomento. Utilizzerai il client MQTT per testare la nuova regola in questa procedura.
- Aver rivisto Amazon Simple Notification Service

  Se non hai mai utilizzato Amazon SNS, consulta Configurazione degli accessi per Amazon SNS. Se hai già completato altri AWS IoT tutorial, Account AWS dovresti già essere configurato correttamente.

Fase 1: creare un argomento Amazon SNS che invii un messaggio di testo SMS

Questa procedura spiega come creare l'argomento Amazon SNS a cui il sensore meteo può inviare i dati dei messaggi. L'argomento Amazon SNS notificherà quindi a tutti i suoi abbonati tramite un messaggio di testo SMS il limite di temperatura superato.

Per creare un argomento Amazon SNS per l'invio di un SMS di testo

1. Creazione di un argomento Amazon SNS.

   a. Accedi alla console Amazon SNS.

   b. Nel pannello di navigazione a sinistra, selezionare Topics (Argomenti).

   c. Nella pagina Topics (Argomenti), seleziona Create topic (Crea argomento).

   d. In Details (Dettagli), scegli il tipo standard. Per impostazione predefinita, la console crea un argomento FIFO.

   e. In Name (Nome), inserisci il nome dell'argomento SNS. Per questo tutorial, digita **`high_temp_notice`**.

   f. Scorri fino alla parte inferiore della pagina e scegli Crea argomento.

      La console apre la pagina Details (Dettagli) del nuovo argomento.

2. Creazione di una sottoscrizione Amazon SNS.

   > **Note**
   >
   > Il numero di telefono utilizzato in questo abbonamento potrebbe comportare costi di messaggistica di testo per i messaggi inviati in questo tutorial.

   a. Nella pagina dei dettagli dell'argomento high_temp_notice, scegli Create subscription (Crea sottoscrizione).

   b. In Create subscription (Crea sottoscrizione), nella sezione Details (Dettagli) nell'elenco Protocol (Protocollo), scegli SMS.

   c. In Endpoint, inserisci il numero di un telefono che può ricevere messaggi di testo. Assicurati di inserirlo in modo che inizi con un +, includa il paese e il prefisso locale e non includa altri caratteri di punteggiatura.

   d. Scegli Create Subscription (Crea sottoscrizione).

3. Testa la notifica di Amazon SNS.

   a. Nella console Amazon SNS, nel pannello di navigazione a sinistra, scegli Topics (Argomenti).

   b. Per aprire la pagina dei dettagli dell'argomento, in Topics (Argomenti), nell'elenco degli argomenti, seleziona high_temp_notice.

c. Per aprire la pagina Publish message to topic (Pubblica il messaggio nell'argomento), nella pagina dei dettagli high_temp_notice, scegli Publish message (Pubblica messaggio).

d. In Publish message to topic (Pubblica il messaggio nell'argomento), nella sezione Message body (Corpo del messaggio), in Message body to send to the endpoint (Corpo del messaggio da inviare all'endpoint), inserisci un breve messaggio.

e. Scorri fino alla parte inferiore della pagina e scegli Publish message (Pubblica messaggio).

f. Sul telefono con il numero che hai usato in precedenza durante la creazione della sottoscrizione, conferma che il messaggio è stato ricevuto.

Se non hai ricevuto il messaggio di prova, controlla due volte il numero di telefono e le impostazioni del tuo cellulare.

Assicurati di poter pubblicare i messaggi di prova dalla Console Amazon SNS prima di continuare il tutorial.

Passaggio 2: crea una AWS IoT regola per inviare il messaggio di testo

La AWS IoT regola che creerai in questo tutorial riguarda gli argomenti `device/`*`device_id`*`/data` MQTT dove si *`device_id`* trova l'ID del dispositivo che ha inviato il messaggio. Questi argomenti sono descritti in un filtro argomento come `device/+/data`, dove + è un carattere jolly che corrisponde a qualsiasi stringa tra i due caratteri di barra in avanti. Questa regola verifica anche il valore del campo `temperature` nel payload del messaggio.

Quando la regola riceve un messaggio da un argomento corrispondente, prende il *`device_id`* dal nome dell'argomento, il valore `temperature` dal payload del messaggio, aggiunge un valore costante per il limite che sta testando e invia questi valori come documento JSON a un argomento di notifica Amazon SNS.

Ad esempio, un messaggio MQTT dal dispositivo sensore meteo numero 32 utilizza l'argomento `device/32/data` e ha un payload del messaggio simile al seguente:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
```

```
  }
}
```

L'istruzione di query regola della regola accetta il valore `temperature` dal payload del messaggio, il valore *`device_id`* dal nome dell'argomento e aggiunge il valore costante `max_temperature` per inviare un payload del messaggio simile a questo all'argomento Amazon SNS:

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30
}
```

Per creare una AWS IoT regola per rilevare un valore di temperatura superiore al limite e creare i dati da inviare all'argomento Amazon SNS

1. Apri l'hub Rules della console. AWS IoT
2. Se è la prima regola, scegli Create (Crea) oppure Create a rule (Crea una regola).
3. In Create a rule (Crea una regola):

   a. In Nome, inserisci **`temp_limit_notify`**.

      Ricorda che il nome di una regola deve essere univoco all'interno della tua regione Account AWS e non può avere spazi. Abbiamo usato un carattere di sottolineatura in questo nome per separare le parole nel nome della regola.

   b. In Description (Descrizione), descrivi la regola.

      Una descrizione significativa ti aiuta a ricordare cosa fa questa regola e perché l'hai creata. La descrizione può essere lunga quanto necessario, quindi sii il più dettagliato possibile.
4. In Rule query statement (Istruzione query regola) di Create a rule (Crea una regola):

   a. In Using SQL version (Uso della versione SQL), seleziona 23/03/2016.

   b. Nella casella di modifica Rule query statement (Istruzione query regola), inserisci l'istruzione:

      ```
      SELECT topic(2) as device_id,
          temperature as reported_temperature,
          30 as max_temperature
        FROM 'device/+/data'
      ```

```
WHERE temperature > 30
```

Questa istruzione:

- Ascolta i messaggi MQTT con un argomento che corrisponde al filtro dell'argomento `device/+/data` e che hanno un valore `temperature` maggiore di 30.
- Seleziona il secondo elemento dalla stringa dell'argomento e lo assegna al campo `device_id`.
- Seleziona il valore del campo `temperature` dal payload del messaggio e lo assegna al campo `reported_temperature`.
- Crea un valore costante `30` per rappresentare il valore limite e lo assegna al campo `max_temperature`.

5. Per aprire l'elenco delle azioni della regola per questa regola, in Set one or more actions (Imposta una o più azioni), scegli Add action (Aggiungi operazione).
6. In Select an action (Seleziona un'operazione), scegli Send a message as an SNS push notification (Invia un messaggio come notifica push SNS).
7. Per aprire la pagina di configurazione dell'operazione selezionata, nella parte inferiore dell'elenco delle operazioni, scegli Configure action (Configura operazione).
8. In Configure action (Configura operazione):

   a. In SNS target (Obiettivo SNS), scegli Select (Seleziona), trova l'argomento SNS denominato high_temp_notice e scegli Select (Seleziona).

   b. In Message format (Formato messaggio), scegli RAW.

   c. In Scegli o crea un ruolo a cui concedere AWS IoT l'accesso per eseguire questa azione, scegli Crea ruolo.

   d. In Create a new role (Crea un nuovo ruolo), in Name (Nome), inserisci un nome univoco per il nuovo ruolo. Ai fini di questo tutorial, utilizza **`sns_rule_role`**.

   e. Scegli Crea ruolo.

   Se si ripete questo tutorial o si riutilizza un ruolo esistente, scegli Update role (Aggiorna ruolo) prima di continuare. In questo modo il documento della policy del ruolo viene aggiornato in modo che funzioni con la destinazione SNS.
9. Scegli Add action (Aggiungi operazione) e torna alla pagina Create a rule (Crea una regola).

Nel titolo della nuova operazione, sotto Send a message as an SNS push notification (Invia un messaggio come notifica push SNS), è possibile visualizzare l'argomento SNS che verrà chiamato dalla regola.

Questa è l'unica operazione della regola che aggiungerai a questa regola.

10. Per creare la regola e completare questo passaggio, in Create a rule (Crea una regola), scorri verso il basso e scegli Create a rule (Crea una regola).

Fase 3: Verifica la AWS IoT regola e la notifica di Amazon SNS

Per testare la nuova regola, utilizza il client MQTT per pubblicare e sottoscrivere i messaggi MQTT utilizzati da questa regola.

Apri il Client MQTT nella console AWS IoT in una nuova finestra. In questo modo è possibile modificare la regola senza perdere la configurazione del client MQTT. Se abbandoni il client MQTT per andare su un'altra pagina della console, non conserverà nessuna sottoscrizione o registri di messaggi.

Utilizzare il client MQTT per testare la regola

1. Nel Client MQTT nella console AWS IoT, effettua la sottoscrizione agli argomenti di input, in questo caso, `device/+/data`.

   a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni) scegli Subscribe to a topic (Effettua la sottoscrizione a un argomento).

   b. In Subscription topic (Argomento sottoscrizione), inserisci l'argomento del filtro dell'argomento di input, **`device/+/data`**.

   c. Lascia gli altri campi ai valori predefiniti.

   d. Scegli Subscribe to topic (Effettua sottoscrizione all'argomento).

      Nella colonna Subscriptions (Sottoscrizioni), sotto Publish to a topic (Pubblica un argomento), **`device/+/data`** viene visualizzato.

2. Pubblicazione di un messaggio nell'argomento di input con un ID dispositivo specifico, **`device/32/data`**. Non è possibile pubblicare su argomenti MQTT che contengono caratteri jolly.

a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni) scegli Publish to topic (Pubblica in un argomento).

b. Nel campo Publish (Pubblica), inserisci il nome dell'argomento di input, **`device/32/data`**.

c. Copia i dati di esempio mostrati qui e, nella casella di modifica sotto il nome dell'argomento, incolla i dati di esempio.

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Scegli Publish to topic (Pubblica in un argomento) per pubblicare il messaggio MQTT.

3. Conferma che il messaggio di testo è stato inviato.

a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni), accanto all'argomento a cui hai effettuato la sottoscrizione in precedenza è presente un punto verde.

I punti verdi indicano che uno o più nuovi messaggi sono stati ricevuti dall'ultima volta che li hai visualizzati.

b. Sotto Subscriptions (Sottoscrizioni), scegli device/+/data (dispositivo/+/dati) per verificare che il payload del messaggio corrisponda a quello che hai appena pubblicato e assomigli a questo:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

c. Controlla il telefono che hai usato per effettuare la sottoscrizione all'argomento SNS e conferma che il contenuto del payload del messaggio sia simile al seguente:

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

Nota che il valore `device_id` è una stringa tra virgolette e il valore `temperature` è numerico. Questo perché la funzione `topic()` ha estratto la stringa dal nome dell'argomento del messaggio di input mentre il valore `temperature` utilizza il valore numerico dal payload del messaggio di input.

Se desideri rendere il valore `device_id` un valore numerico, sostituisci `topic(2)` nell'istruzione query della regola con:

```
cast(topic(2) AS DECIMAL)
```

Nota che la trasformazione del valore `topic(2)` in un valore numerico `DECIMAL` funzionerà solo se quella parte dell'argomento contiene solo caratteri numerici.

4. Prova a inviare un messaggio MQTT in cui la temperatura non supera il limite.

a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni), scegli Publish to topic (Pubblica in un argomento).

b. Nel campo Publish (Pubblica), inserisci il nome dell'argomento di input, **`device/33/data`**.

c. Copia i dati di esempio mostrati qui e, nella casella di modifica sotto il nome dell'argomento, incolla i dati di esempio.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Per inviare il messaggio MQTT, scegli Publish to topic (Pubblica in un argomento).

Dovrebbe essere visualizzato il messaggio inviato nella sottoscrizione **`device/+/data`**. Tuttavia, poiché il valore della temperatura è inferiore alla temperatura massima nell'istruzione di query della regola, non si dovrebbe ricevere un messaggio di testo.

Se il comportamento non viene visualizzato correttamente, controlla i suggerimenti per la risoluzione dei problemi.

Risoluzione dei problemi della regola di messaggi SNS

Ecco alcune cose da controllare nel caso in cui non stai vedendo i risultati che ti aspetti.

- Hai un banner di errore

  Se viene visualizzato un errore quando è stato pubblicato il messaggio di input, correggilo prima. I seguenti passaggi potrebbero aiutarti a correggere l'errore.
- Il messaggio di input non viene visualizzato nel client MQTT

  Ogni volta che pubblichi il tuo messaggio di input nell'argomento `device/22/data`, tale messaggio dovrebbe essere visualizzato nel client MQTT, se hai effettuato la sottoscrizione al filtro dell'argomento `device/+/data` come descritto nella procedura.

  Controlli

  - Controlla il filtro degli argomenti a cui hai effettuato la sottoscrizione

    Se hai effettuato la sottoscrizione all'argomento del messaggio di input come descritto nella procedura, visualizzerai una copia del messaggio di input ogni volta che lo pubblichi.

    Se il messaggio non viene visualizzato, controlla il nome dell'argomento sottoscritto e confrontalo con l'argomento in cui è stato pubblicato. I nomi degli argomenti fanno distinzione tra maiuscole e minuscole e l'argomento a cui è stato sottoscritto deve essere identico all'argomento in cui hai pubblicato il payload dei messaggi.
  - Controlla la funzione di pubblicazione dei messaggi

    Nel client MQTT, sotto Subscriptions (Sottoscrizioni), scegli device/+/data (dispositivo/+/dati), controlla l'argomento del messaggio di pubblicazione e quindi scegli Publish to topic (Pubblica in un argomento). Dovresti vedere il payload del messaggio dalla casella di modifica sotto l'argomento visualizzato nell'elenco dei messaggi.

- Se non ricevi un messaggio SMS

  Affinché la regola funzioni, deve disporre della policy corretta che la autorizzi a ricevere un messaggio e inviare una notifica SNS e deve ricevere il messaggio.

  Controlli

  - Controlla il Regione AWS tuo client MQTT e la regola che hai creato

    La console in cui si esegue il client MQTT deve trovarsi nella stessa regione AWS della regola creata.

  - Verifica che il valore della temperatura nel payload del messaggio superi la soglia di prova

    Se il valore della temperatura è minore o uguale a 30, come definito nell'istruzione query della regola, la regola non eseguirà alcuna delle operazioni.

  - Controlla l'argomento del messaggio di input nell'istruzione query della regola

    Affinché la regola funzioni, è necessario che venga visualizzato un messaggio con il nome dell'argomento corrispondente al filtro dell’argomento nella clausola FROM dell'istruzione query della regola.

    Controlla l'ortografia del filtro argomento nell'istruzione query regola con quella dell'argomento nel client MQTT. I nomi degli argomenti fanno distinzione tra maiuscole e minuscole e l'argomento del messaggio deve corrispondere al filtro argomento nell'istruzione query della regola.

  - Controllare il contenuto del payload del messaggio di input

    Affinché la regola funzioni, è necessario trovare il campo dati nel payload del messaggio dichiarato nell'istruzione SELECT.

    Controlla l'ortografia del campo `temperature` nell'istruzione query della regola con quella del payload del messaggio nel client MQTT. I nomi dei campi fanno distinzione tra maiuscole e minuscole e il campo `temperature`nell'istruzione query della regola deve essere identico a quello del campo `temperature` nel payload del messaggio.

    Assicurati che il documento JSON nel payload del messaggio sia formattato correttamente. Se il JSON presenta errori, come una virgola mancante, la regola non sarà in grado di leggerlo.

  - Controlla l'argomento del messaggio ripubblicato nell'operazione della regola

L'argomento a cui l'operazione Ripubblica regola pubblica il nuovo messaggio deve corrispondere all'argomento a cui è stato sottoscritto nel client MQTT.

Apri la regola creata nella console e controlla l'argomento in cui l'operazione della regola ripubblicherà il messaggio.

- Controlla il ruolo utilizzato dalla regola

  L'operazione della regola deve disporre dell'autorizzazione per ricevere l'argomento originale e pubblicare il nuovo argomento.

  Le policy che autorizzano la regola a ricevere i dati dei messaggi e a ripubblicarli sono specifiche degli argomenti utilizzati. Se si modifica l'argomento utilizzato per ripubblicare i dati del messaggio, è necessario aggiornare il ruolo dell'operazione della regola per aggiornare la policy in modo che corrisponda all'argomento corrente.

  Se si sospetta che questo sia il problema, modificare l'operazione Ripubblica regola e creare un nuovo ruolo. I nuovi ruoli creati dall'operazione della regola ricevono le autorizzazioni necessarie per eseguire queste operazioni.

Fase 4: Esamina i risultati e i passaggi successivi

In questo tutorial:

- Hai creato e testato un argomento di notifica e una sottoscrizione Amazon SNS.
- È stata utilizzata una query SQL semplice che funziona in un'istruzione di query della regola per creare un nuovo messaggio per la notifica.
- Hai creato una AWS IoT regola per inviare una notifica Amazon SNS che utilizza il tuo payload di messaggi personalizzato.
- Hai usato il client MQTT per testare la tua regola. AWS IoT

Fasi successive

Dopo aver inviato alcuni messaggi di testo con questa regola, prova a sperimentarla per vedere come la modifica di alcuni aspetti del tutorial influisce sul messaggio e quando viene inviato. Ecco alcune idee per iniziare.

- *device_id*Modificate l'argomento del messaggio di input e osservate l'effetto nel contenuto del messaggio di testo.
- Modifica i campi selezionati nell'istruzione query della regola e osserva l'effetto nel contenuto del messaggio di testo.
- Modifica il test nell'istruzione query della regola per verificare una temperatura minima anziché una temperatura massima. Ricordati di cambiare il nome di `max_temperature`!
- Aggiungi un'operazione della regola di ripubblicazione per inviare un messaggio MQTT quando viene inviata una notifica SNS.
- Prova il prossimo tutorial di questa serie e scopri come Tutorial: Archiviazione dei dati del dispositivo in una tabella DynamoDB.

Tutorial: Archiviazione dei dati del dispositivo in una tabella DynamoDB

Questo tutorial dimostra come creare una AWS IoT regola che invii i dati dei messaggi a una tabella DynamoDB.

In questo tutorial potrai creare una regola per l'invio di dati di messaggi da un dispositivo immaginario di sensori meteorologici a una tabella Dynamo DB. La regola formatta i dati da molti sensori meteorologici in modo che possano essere aggiunti a una singola tabella di database.

Cosa imparerai in questo tutorial

- Come creare una tabella Dynamo DB
- Come inviare i dati dei messaggi a una tabella DynamoDB da una regola AWS IoT
- Come utilizzare i modelli sostitutivi in una regola AWS IoT
- Come utilizzare semplici query SQL e funzioni in un'istruzione di query della regola
- Come usare il client MQTT per testare una regola AWS IoT

Questo tutorial dura circa 30 minuti.

In questo tutorial, dovrai:

- Fase 1: Creazione della tabella Dynamo DB per questo tutorial
- Fase 2: Creare una AWS IoT regola per inviare dati alla tabella DynamoDB
- Fase 3: Verificare la AWS IoT regola e la tabella DynamoDB
- Fase 4: Esamina i risultati e i passaggi successivi

Prima di iniziare questo tutorial, assicurati di disporre di:

- Configurare Account AWS

  Avrai bisogno della tua Account AWS AWS IoT console per completare questo tutorial.
- Aver rivisto Visualizza i messaggi MQTT con il AWS IoT client MQTT

  Assicurati di poter usare il client MQTT per sottoscrivere e pubblicare un argomento. Utilizzerai il client MQTT per testare la nuova regola in questa procedura.
- Aver rivisto la panoramica di Amazon DynamoDB

  Se non hai già usato Dynamo DB, consulta Nozioni di base su Dynamo DB per acquisire familiarità con i concetti e le operazioni di base di Dynamo DB.

Fase 1: Creazione della tabella Dynamo DB per questo tutorial

In questo tutorial creerai una tabella Dynamo DB con questi attributi per registrare i dati dai dispositivi immaginari con sensori meteorologici:

- `sample_time` è una chiave primaria e descrive l'ora in cui il campione è stato registrato.
- `device_id` è una chiave di ordinamento e descrive il dispositivo che ha fornito l'esempio
- `device_data` sono i dati ricevuti dal dispositivo e formattati dall'istruzione query della regola

Per creare la tabella Dynamo DB per questo tutorial

1. Accedi alla console Dynamo DB e scegli Create table (Crea una tabella).
2. In Create table (Crea tabella):
   a. In Table name (Nome tabella), inserisci il nome della tabella: **`wx_data`**.
   b. In Partition key (Chiave di partizione) inserisci **`sample_time`** e, nell'elenco delle opzioni accanto al campo, scegli **`Number`**.
   c. In Sort key (chiave di ordinamento), inserisci **`device_id`** e nell'elenco delle opzioni accanto al campo, scegli **`Number`**.
   d. Nella parte inferiore della pagina, scegli Create (Crea).

Definisci `device_data` in seguito, quando configuri l'operazione della regola Dynamo DB.

Fase 2: Creare una AWS IoT regola per inviare dati alla tabella DynamoDB

In questo passaggio utilizzerai l'istruzione query della regola per formattare i dati provenienti dai dispositivi sensore meteo immaginari per scrivere nella tabella del database.

Un esempio di payload di messaggio ricevuto da un dispositivo con sensore meteo è simile al seguente:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

Per la voce del database, si utilizzerà l'istruzione query della regola per appiattire la struttura del payload del messaggio in modo che sia simile al seguente:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind_velocity": 22,
  "wind_bearing": 255
}
```

In questa regola, utilizzerai anche un paio di Modelli di sostituzione. I modelli di sostituzione sono espressioni che consentono di inserire valori dinamici dalle funzioni e dai dati dei messaggi.

Per creare la AWS IoT regola per inviare dati alla tabella DynamoDB

1. Apri l'hub Rules (Regole) della console AWS IoT. In alternativa, puoi aprire la AWS IoT home page all'interno di Console di gestione AWS e passare a Message Routing>Rules.
2. Per iniziare a creare la nuova regola in Rules (Regole), scegli Create rule (Crea regola).
3. In Proprietà delle regole:
   a. In Role name (Nome ruolo) immettere **`wx_data_ddb`**.

Ricorda che il nome di una regola deve essere univoco all'interno della tua regione Account AWS e non può avere spazi. Abbiamo usato un carattere di sottolineatura in questo nome per separare le due parole nel nome della regola.

b. In Description (Descrizione), descrivi la regola.

Una descrizione significativa ti aiuta a ricordare cosa fa questa regola e perché l'hai creata. La descrizione può essere lunga quanto necessario, quindi sii il più dettagliato possibile.

4. Seleziona Successivo per continuare.
5. In Istruzione SQL:

a. In Versione SQL, seleziona **`2016-03-23`**.

b. Nella casella di modifica Istruzione SQL, inserisci l'istruzione:

```
SELECT temperature, humidity, barometer,
  wind.velocity as wind_velocity,
  wind.bearing as wind_bearing,
FROM 'device/+/data'
```

Questa istruzione:

- Ascolta i messaggi MQTT con un argomento che corrisponde al filtro dell'argomento `device/+/data`.
- Formatta gli elementi dell'attributo `wind` come attributi individuali.
- Consente di passare gli attributi `temperature`, `humidity` e `barometer` invariati.

6. Seleziona Successivo per continuare.
7. In Azioni delle regole:

a. Per aprire l'elenco delle azioni della regola per questa regola, in Azione 1, scegli **`DynamoDB`**.

> **Note**
>
> Assicurati di scegliere DynamoDB e non DBv2 Dynamo come azione della regola.

b. In Table name (Nome tabella) scegli il nome della tabella Dynamo DB creata in un passaggio precedente: **`wx_data`**.

I campi Partition key type (Tipo di chiave di partizione) e Sort key type (Tipo di chiave di ordinamento) sono compilati con i valori dalla tabella Dynamo DB.

c. In Chiave di partizione, immettere **`sample_time`**.

d. In Partition key value (Valore della chiave di partizione), immettere **`${timestamp()}`**.

   Questo è il primo di Modelli di sostituzione che userai in questa regola. Invece di utilizzare un valore dal payload del messaggio, utilizzerà il valore restituito dal metodo della funzione timestamp. Per ulteriori informazioni, consulta timestamp nella Guida per gli sviluppatori di AWS IoT Core .

e. In Sort key (chiave di ordinamento), immettere **`device_id`**.

f. In Sort key value (Valore della chiave di ordinamento), immettere **`${cast(topic(2) AS DECIMAL)}`**.

   Questa è la seconda di Modelli di sostituzione che userai in questa regola. Inserisce il valore del secondo elemento nel nome dell'argomento, che è l'ID del dispositivo, dopo la sua trasformazione in un valore DECIMALE per corrispondere al formato numerico della chiave. Per ulteriori informazioni sugli argomenti, consulta la sezione argomenti nella Guida per sviluppatori di AWS IoT Core . Oppure per saperne di più sul casting, vedi cast nella Guida per lo Sviluppatore di AWS IoT Core .

g. In Write message data to this column (Scrivi i dati del messaggio in questa colonna), immettere **`device_data`**.

   Questo creerà la colonna `device_data` nella tabella Dynamo DB.

h. Lascia vuoto il campo Operation (Operazione).

i. In IAM Role (Ruolo IAM), scegli Create new role (Crea nuovo ruolo).

j. Nella finestra di dialogo Create role (Crea ruolo), per Role name (Nome ruolo), immetti wx_ddb_role. Questo nuovo ruolo conterrà automaticamente una policy con prefisso "aws-iot-rule" che consentirà alla **`wx_data_ddb`** regola di inviare dati alla tabella **`wx_data`** DynamoDB che hai creato.

k. In IAM role (Ruolo IAM) scegliere **`wx_ddb_role`**.

l. Nella parte inferiore della pagina scegli Next (Avanti).

8. Nella parte inferiore della pagina Rivedi e crea, scegli Create per creare la regola.

Fase 3: Verificare la AWS IoT regola e la tabella DynamoDB

Per testare la nuova regola, utilizza il client MQTT per pubblicare e sottoscrivere i messaggi MQTT utilizzati in questo test.

Apri il Client MQTT nella console AWS IoT in una nuova finestra. In questo modo è possibile modificare la regola senza perdere la configurazione del client MQTT. Il client MQTT non conserva sottoscrizioni o registri dei messaggi se si lascia per andare a un'altra pagina della console. Ti consigliamo inoltre di aprire una finestra della console separata sull'hub DynamoDB Tables nella console per visualizzare AWS IoT le nuove voci inviate dalla regola.

Utilizzare il client MQTT per testare la regola

1. Nel Client MQTT nella console AWS IoT, effettua la sottoscrizione all'argomento di input, `device/+/data`.

   a. Nel client MQTT, scegli Subscribe to a topic (Effettua la sottoscrizione a un argomento).
   b. Per Topic filter (Filtro argomenti), inserisci l'argomento del filtro dell'argomento di input, **`device/+/data`**.
   c. Scegli Abbonati.
2. Ora pubblica un messaggio per l'argomento di input con un ID dispositivo specifico, **`device/22/data`**. Non è possibile pubblicare su argomenti MQTT che contengono caratteri jolly.

   a. Nel client MQTT, scegli Publish to a topic (Pubblica in un argomento).
   b. Per Topic name (Nome argomento), inserisci un nome per l'argomento di input, **`device/22/data`**.
   c. Per Message payload (Payload del messaggio)(, inserisci i seguenti dati di esempio.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. Per pubblicare il messaggio MQTT, scegli Publish (Pubblica).

e. Nel client MQTT, scegli Subscribe to a topic (Effettua la sottoscrizione a un argomento). Nella colonna Subscribe (Effettua sottoscrizione), scegli la sottoscrizione **`device/+/data`**. Verifica che vengano visualizzati i dati di esempio del passaggio precedente.

3. Seleziona per visualizzare la riga della tabella Dynamo DB creata dalla regola.

   a. Nell'hub DynamoDB Tables AWS IoT della console, scegli wx_data, quindi scegli la scheda Items.

      Se sei già sulla scheda Items (Elementi), è possibile che sia necessario aggiornare la visualizzazione selezionando l'icona Aggiorna nell'angolo in alto a destra dell'intestazione della tabella.

   b. Nota che i valori sample_time nella tabella sono collegamenti, quindi aprine uno. Se hai appena inviato il tuo primo messaggio, sarà l'unico nella lista.

      Questo collegamento visualizza tutti i dati nella riga della tabella.

   c. Espandi la voce device_dati per visualizzare i dati risultanti dall'istruzione query della regola.

   d. Esplora le diverse rappresentazioni dei dati disponibili in questo display. È possibile modificare i dati anche in questo display.

   e. Al termine della revisione di questa riga di dati, per salvare le modifiche apportate, scegli Save (Salva) o, per uscire senza salvare le modifiche, scegli Cancel (Annulla).

Se il comportamento non viene visualizzato correttamente, controlla i suggerimenti per la risoluzione dei problemi.

Risoluzione dei problemi relativi alla regola Dynamo DB

Ecco alcune cose da controllare nel caso in cui non vedi i risultati che ti aspetti.

- Hai un banner di errore

  Se viene visualizzato un errore quando è stato pubblicato il messaggio di input, correggilo prima. I seguenti passaggi potrebbero aiutarti a correggere l'errore.

- Il messaggio di input non viene visualizzato nel client MQTT

  Ogni volta che pubblichi il messaggio di input nell'argomento `device/22/data`, tale messaggio dovrebbe essere visualizzato nel client MQTT se hai effettuato la sottoscrizione al filtro argomento `device/+/data` come descritto nella procedura.

Controlli

- Controlla il filtro degli argomenti a cui hai effettuato la sottoscrizione

  Se hai effettuato la sottoscrizione all'argomento del messaggio di input come descritto nella procedura, visualizzerai una copia del messaggio di input ogni volta che lo pubblichi.

  Se il messaggio non viene visualizzato, controlla il nome dell'argomento sottoscritto e confrontalo con l'argomento in cui è stato pubblicato. I nomi degli argomenti fanno distinzione tra maiuscole e minuscole e l'argomento a cui è stato sottoscritto deve essere identico all'argomento in cui hai pubblicato il payload dei messaggi.
- Controlla la funzione di pubblicazione dei messaggi

  Nel client MQTT, sotto Subscriptions (Sottoscrizioni), scegli device/+/data (dispositivo/+/dati), controlla l'argomento del messaggio di pubblicazione e quindi scegli Publish to topic (Pubblica in un argomento). Dovresti vedere il payload del messaggio dalla casella di modifica sotto l'argomento visualizzato nell'elenco dei messaggi.

- I dati non vengono visualizzati nella tabella Dynamo DB

  Per prima cosa, aggiorna la visualizzazione selezionando l'icona di aggiornamento nell'angolo in alto a destra dell'intestazione della tabella. Se non vengono visualizzati i dati che si sta cercando, controlla quanto segue.

  Controlli

  - Controlla il Regione AWS tuo client MQTT e la regola che hai creato

    La console in cui si esegue il client MQTT deve trovarsi nella stessa regione AWS della regola creata.
  - Controlla l'argomento del messaggio di input nell'istruzione query della regola

    Affinché la regola funzioni, è necessario che venga visualizzato un messaggio con il nome dell'argomento corrispondente al filtro dell’argomento nella clausola FROM dell'istruzione query della regola.

    Controlla l'ortografia del filtro argomento nell'istruzione query regola con quella dell'argomento nel client MQTT. I nomi degli argomenti fanno distinzione tra maiuscole e minuscole e l'argomento del messaggio deve corrispondere al filtro argomento nell'istruzione query della regola.

- Controllare il contenuto del payload del messaggio di input

  Affinché la regola funzioni, è necessario trovare il campo dati nel payload del messaggio dichiarato nell'istruzione SELECT.

  Controlla l'ortografia del campo `temperature` nell'istruzione query della regola con quella del payload del messaggio nel client MQTT. I nomi dei campi fanno distinzione tra maiuscole e minuscole e il campo `temperature`nell'istruzione query della regola deve essere identico a quello del campo `temperature` nel payload del messaggio.

  Assicurati che il documento JSON nel payload del messaggio sia formattato correttamente. Se il JSON presenta errori, come una virgola mancante, la regola non sarà in grado di leggerlo.
- Controlla i nomi delle chiavi e dei campi utilizzati nell'operazione della regola

  I nomi dei campi utilizzati nella regola dell'argomento devono corrispondere a quelli trovati nel payload del messaggio JSON del messaggio pubblicato.

  Apri la regola creata nella console e controlla i nomi dei campi nella configurazione dell'operazione della regola con quelli utilizzati nel client MQTT.
- Controlla il ruolo utilizzato dalla regola

  L'operazione della regola deve disporre dell'autorizzazione per ricevere l'argomento originale e pubblicare il nuovo argomento.

  Le policy che autorizzano la regola a ricevere i dati dei messaggi e ad aggiornare la tabella Dynamo DB sono specifiche degli argomenti utilizzati. Se si modifica l'argomento o il nome della tabella Dynamo DB utilizzato dalla regola, è necessario aggiornare il ruolo dell'operazione della regola per aggiornare la policy in modo che corrisponda.

  Se si sospetta che questo sia il problema, modificare l'operazione della regola e creare un nuovo ruolo. I nuovi ruoli creati dall'operazione della regola ricevono le autorizzazioni necessarie per eseguire queste operazioni.

Fase 4: Esamina i risultati e i passaggi successivi

Dopo aver inviato alcuni messaggi alla tabella Dynamo DB con questa regola, provare a sperimentarla per verificare come la modifica di alcuni aspetti del tutorial influisca sui dati scritti nella tabella. Ecco alcune idee per iniziare.

- Modifica l'*device_id*argomento del messaggio di input e osserva l'effetto sui dati. È possibile utilizzarlo per simulare la ricezione di dati da più sensori meteorologici.
- Modifica i campi selezionati nell'istruzione query della regola e osserva l'effetto sui dati. Questo strumento consente di filtrare i dati archiviati nella tabella.
- Aggiungi un'operazione della regola di ripubblicazione per inviare un messaggio MQTT per ogni riga aggiunta alla tabella. È possibile utilizzarlo per il debug.

Dopo aver completato questo tutorial, consulta the section called “Formattazione di una notifica utilizzando una funzione AWS Lambda ”.

Tutorial: Formattazione di una notifica utilizzando una funzione AWS Lambda

Questo tutorial dimostra come inviare i dati dei messaggi MQTT a un' AWS Lambda azione per la formattazione e l'invio a un altro servizio. AWS In questo tutorial, l' AWS Lambda azione utilizza l' AWS SDK per inviare il messaggio formattato all'argomento Amazon SNS che hai creato nel tutorial su come farlo. the section called “Invio di una notifica Amazon SNS”

Nel tutorial su come the section called “Invio di una notifica Amazon SNS”, il documento JSON risultante dall'istruzione di query della regola è stato inviato come corpo del messaggio di testo. Il risultato è un messaggio di testo simile a questo esempio:

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

In questo tutorial, utilizzerai un'azione di AWS Lambda regola per richiamare una AWS Lambda funzione che formatta i dati dall'istruzione di query della regola in un formato più intuitivo, come questo esempio:

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

La AWS Lambda funzione che creerai in questo tutorial formatta la stringa del messaggio utilizzando i dati dell'istruzione rule query e richiama la funzione di pubblicazione SNS dell' AWS SDK per creare la notifica.

Cosa imparerai in questo tutorial

- Come creare e testare una funzione AWS Lambda
- Come utilizzare l' AWS SDK in una AWS Lambda funzione per pubblicare una notifica Amazon SNS

- Come utilizzare semplici query SQL e funzioni in un'istruzione di query della regola
- Come utilizzare il client MQTT per testare una regola AWS IoT

Questo tutorial dura circa 45 minuti.

In questo tutorial, dovrai:

- Passaggio 1: creare una AWS Lambda funzione che invii un messaggio di testo
- Passaggio 2: creare una AWS IoT regola con un'azione di regola AWS Lambda
- Fase 3: Verificare la AWS IoT regola e AWS Lambda l'azione della regola
- Fase 4: Esamina i risultati e i passaggi successivi

Prima di iniziare questo tutorial, assicurati di disporre di:

- Configurare Account AWS

  Avrai bisogno della tua Account AWS AWS IoT console per completare questo tutorial.
- Aver rivisto Visualizza i messaggi MQTT con il AWS IoT client MQTT

  Assicurati di poter usare il client MQTT per sottoscrivere e pubblicare un argomento. Utilizzerai il client MQTT per testare la nuova regola in questa procedura.
- Aver completato gli altri tutorial sulle regole in questa sezione

  Questo tutorial richiede l'argomento di notifica SNS creato nel tutorial su come the section called “Invio di una notifica Amazon SNS”. Si presuppone inoltre che in questa sezione siano stati completati i tutorial relativi alle regole.
- Aver rivisto la panoramica di AWS Lambda

  Se non l'hai mai usato AWS Lambda prima, AWS Lambdaconsulta la sezione Guida introduttiva a Lambda per conoscerne i termini e i concetti.

Passaggio 1: creare una AWS Lambda funzione che invii un messaggio di testo

La AWS Lambda funzione di questo tutorial riceve il risultato dell'istruzione di query della regola, inserisce gli elementi in una stringa di testo e invia la stringa risultante ad Amazon SNS come messaggio in una notifica.

A differenza del tutorial su come farethe section called “Invio di una notifica Amazon SNS”, che utilizzava un'azione di AWS IoT regola per inviare la notifica, questo tutorial invia la notifica dalla funzione Lambda utilizzando una funzione dell' AWS SDK. L'argomento effettivo di notifica Amazon SNS utilizzato in questo tutorial, tuttavia, è lo stesso utilizzato nel tutorial su come the section called “Invio di una notifica Amazon SNS”.

Per creare una AWS Lambda funzione che invii un messaggio di testo

1. Crea una nuova AWS Lambda funzione.

   a. Nella console AWS Lambda scegliere Create function (Crea funzione).

   b. In Create function (Crea funzione) seleziona Use a blueprint (Usa un blueprint).

      Cerca e seleziona il blueprint **`hello-world-python`**, quindi scegli Configure (Configura).

   c. In Basic information (Informazioni di base):

      i. In Function name (Nome funzione), inserisci il nome della funzione, **`format-high-temp-notification`**.

      ii. In Ruolo di esecuzione, scegli Crea un nuovo ruolo dai modelli di AWS policy.

      iii. In Nome ruolo, inserisci il nome del nuovo ruolo, **`format-high-temp-notification-role`**.

      iv. In Policy templates (Modelli di policy) - facoltativo, cerca e seleziona Amazon SNS publish policy (Policy di pubblicazione Amazon SNS).

      v. Scegli Crea funzione.

2. Modifica il codice del blueprint per formattare e inviare una notifica Amazon SNS.

   a. Dopo aver creato la funzione, dovresti vedere la pagina dei format-high-temp-notificationdettagli. In caso contrario, aprilo dalla pagina Lambda Funzioni.

   b. Nella pagina dei format-high-temp-notificationdettagli, scegli la scheda Configurazione e scorri fino al pannello Codice funzione.

   c. Nella finestra Function code (Codice della funzione), nel pannello Environment (Ambiente), scegli il file Python, `lambda_function.py`.

   d. Nella finestra Function code (Codice della funzione), elimina tutto il codice originale del programma dal blueprint e sostituiscilo con questo codice.

```
import boto3
#
```

```
#   expects event parameter to contain:
#   {
#       "device_id": "32",
#       "reported_temperature": 38,
#       "max_temperature": 30,
#       "notify_topic_arn": "arn:aws:sns:us-
east-1:57EXAMPLE833:high_temp_notice"
#   }
#
#   sends a plain text string to be used in a text message
#
#      "Device {0} reports a temperature of {1}, which exceeds the limit of
 {2}."
#
#   where:
#       {0} is the device_id value
#       {1} is the reported_temperature value
#       {2} is the max_temperature value
#
def lambda_handler(event, context):

    # Create an SNS client to send notification
    sns = boto3.client('sns')

    # Format text message from data
    message_text = "Device {0} reports a temperature of {1}, which exceeds the
 limit of {2}.".format(
            str(event['device_id']),
            str(event['reported_temperature']),
            str(event['max_temperature'])
        )

    # Publish the formatted message
    response = sns.publish(
            TopicArn = event['notify_topic_arn'],
            Message = message_text
        )

    return response
```

e. Seleziona Implementa.

3. In una nuova finestra, controlla l'Amazon Resource Name (ARN) dell'argomento Amazon SNS dal turtorial su come the section called “Invio di una notifica Amazon SNS”.

a. In una nuova finestra, apri la pagina degli argomenti della console Amazon SNS.

b. Nella pagina Topics (Argomenti), trova l'argomento della notifica high_temp_notice nell'elenco degli argomenti Amazon SNS.

c. Trova l'ARN dell'argomento della notifica high_temp_notice da utilizzare nel passaggio successivo.

4. Crea un test per la tua funzione Lambda.

a. Nella pagina Funzioni Lambda della console, nella pagina dei format-high-temp-notificationdettagli, scegli Seleziona un evento di test nell'angolo in alto a destra della pagina (anche se sembra disabilitato), quindi scegli Configura eventi di test.

b. In Configure test event (Configura eventi di test), scegli Create new test event (Crea un nuovo evento di test).

c. In Event name (Nome evento), inserisci **`SampleRuleOutput`**.

d. Nell'editor JSON riportato di seguito Event name (Nome evento), incolla questo documento JSON di esempio. Questo è un esempio di ciò che la AWS IoT regola invierà alla funzione Lambda.

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

e. Fai riferimento alla finestra che ha l'ARN dell'argomento della notifica high_temp_notice e copia il valore dell'ARN.

f. Sostituisci il valore `notify_topic_arn` nell'editor JSON con l'ARN dall'argomento di notifica.

Tieni aperta questa finestra in modo da poter utilizzare di nuovo questo valore ARN quando crei la regola AWS IoT .

g. Scegli Create (Crea).

5. Testare la funzione con i dati di esempio.

a. Nella pagina dei format-high-temp-notificationdettagli, nell'angolo in alto a destra della pagina, conferma che SampleRuleOutputappare accanto al pulsante Test. In caso contrario, sceglilo dall'elenco degli eventi di test disponibili.

b. Per inviare il messaggio di output della regola di esempio alla funzione, scegli Test.

Se la funzione e la notifica hanno funzionato, riceverai un messaggio di testo sul tuo cellulare che ha sottoscritto la notifica.

Se non hai ricevuto un messaggio di testo sul telefono, controlla il risultato dell'operazione. Nel pannello Function code (Codice della funzione), nella casella Execution result (Risultato dell'esecuzione), rivedi la risposta per trovare eventuali errori che si sono verificati. Non proseguire con il passaggio successivo finché la funzione non è in grado di inviare la notifica al telefono.

Passaggio 2: creare una AWS IoT regola con un'azione di regola AWS Lambda

In questo passaggio, si utilizzerà l'istruzione query della regola per formattare i dati dal dispositivo sensore meteo immaginario da inviare a una funzione Lambda, che formatta e invia un messaggio di testo.

Un esempio di messaggio payload ricevuto da un dispositivo con sensore meteo è simile al seguente:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

In questa regola, si utilizzerà l'istruzione query della regola per creare un payload dei messaggi per la funzione Lambda simile al seguente:

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
```

```
}
```

Contiene tutte le informazioni necessarie alla funzione Lambda per formattare e inviare il messaggio di testo corretto.

Per creare la AWS IoT regola per chiamare una funzione Lambda

1. Apri l'hub Rules della AWS IoT console.
2. Per iniziare a creare la nuova regola in Rules (Regole), scegli Create (Crea).
3. Nella parte superiore di Create a rule (Crea una regola):
   a. In Name (Nome), inserisci il nome della regola, **`wx_friendly_text`**.

      Ricorda che il nome di una regola deve essere univoco all'interno della tua regione Account AWS e non può avere spazi. Abbiamo usato un carattere di sottolineatura in questo nome per separare le due parole nel nome della regola.
   b. In Description (Descrizione), descrivi la regola.

      Una descrizione significativa ti aiuta a ricordare cosa fa questa regola e perché l'hai creata. La descrizione può essere lunga quanto necessario, quindi sii il più dettagliato possibile.
4. In Rule query statement (Istruzione query regola) di Create a rule (Crea una regola):
   a. In Using SQL version (Uso della versione SQL), seleziona **`2016-03-23`**.
   b. Nella casella di modifica Rule query statement (Istruzione query regola), inserisci l'istruzione:

      ```
      SELECT
        cast(topic(2) AS DECIMAL) as device_id,
        temperature as reported_temperature,
        30 as max_temperature,
        'arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice' as notify_topic_arn
      FROM 'device/+/data' WHERE temperature > 30
      ```

      Questa istruzione:

      - Ascolta i messaggi MQTT con un argomento che corrisponde al filtro dell'argomento `device/+/data` e che hanno un valore `temperature` maggiore di 30.
      - Seleziona il secondo elemento dalla stringa dell'argomento, lo converte in un numero decimale e quindi lo assegna al campo `device_id`.

- Seleziona il valore del campo `temperature` dal payload del messaggio e lo assegna al campo `reported_temperature`.
- Crea un valore costante, `30`, per rappresentare il valore limite e lo assegna al campo `max_temperature`.
- Crea un valore costante per il campo `notify_topic_arn`.

c. Fai riferimento alla finestra che ha l'ARN dell'argomento della notifica high_temp_notice e copia il valore dell'ARN.

d. Sostituisci il valore ARN (*`arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice`*) nell'editor delle istruzioni di interrogazione delle regole con l'ARN dell'argomento della notifica.

5. In Set one or more actions (Imposta una o più operazioni):

a. Per aprire l'elenco delle operazioni delle regole per questa regola, scegli Add action (Aggiungi operazione).

b. In Select an action (Seleziona un'operazione), scegli Send a message to a Lambda function (Invia un messaggio a una funzione Lambda).

c. Per aprire la pagina di configurazione dell'operazione selezionata, nella parte inferiore dell'elenco delle operazioni, scegli Configure action (Configura operazione).

6. In Configure action (Configura operazione):

a. In Function name (Nome della funzione), scegli Select (Seleziona).

b. Scegli format-high-temp-notification.

c. Nella parte inferiore di Configure action (Configura operazione), scegli Add action (Aggiungi operazione).

d. Per creare la regola, nella parte inferiore di Create a rule (Crea una regola), scegli Create a rule (Crea una regola).

Fase 3: Verificare la AWS IoT regola e AWS Lambda l'azione della regola

Per testare la nuova regola, utilizza il client MQTT per pubblicare e sottoscrivere i messaggi MQTT utilizzati da questa regola.

Apri il Client MQTT nella console AWS IoT in una nuova finestra. Ora è possibile modificare la regola senza perdere la configurazione del client MQTT. Se abbandoni il client MQTT per andare in un'altra pagina della console, perderai le tue sottoscrizioni o i registri dei messaggi.

Utilizzare il client MQTT per testare la regola

1. Nel Client MQTT nella console AWS IoT, effettua la sottoscrizione agli argomenti di input, in questo caso, `device/+/data`.

   a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni) scegli Subscribe to a topic (Effettua la sottoscrizione a un argomento).

   b. In Subscription topic (Argomento sottoscrizione), inserisci l'argomento del filtro dell'argomento di input, **`device/+/data`**.

   c. Lascia gli altri campi ai valori predefiniti.

   d. Scegli Subscribe to topic (Effettua sottoscrizione all'argomento).

      Nella colonna Subscriptions (Sottoscrizioni), sotto Publish to a topic (Pubblica un argomento), **`device/+/data`** viene visualizzato.

2. Pubblicazione di un messaggio nell'argomento di input con un ID dispositivo specifico, **`device/32/data`**. Non è possibile pubblicare su argomenti MQTT che contengono caratteri jolly.

   a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni) scegli Publish to topic (Pubblica in un argomento).

   b. Nel campo Publish (Pubblica), inserisci il nome dell'argomento di input, **`device/32/data`**.

   c. Copia i dati di esempio mostrati qui e, nella casella di modifica sotto il nome dell'argomento, incolla i dati di esempio.

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. Scegli Publish to topic (Pubblica in un argomento) per pubblicare il messaggio MQTT.

3. Conferma che il messaggio di testo è stato inviato.

   a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni), accanto all'argomento a cui hai effettuato la sottoscrizione in precedenza è presente un punto verde.

I punti verdi indicano che uno o più nuovi messaggi sono stati ricevuti dall'ultima volta che li hai visualizzati.

b. Sotto Subscriptions (Sottoscrizioni), scegli device/+/data (dispositivo/+/dati) per verificare che il payload del messaggio corrisponda a quello che hai appena pubblicato e assomigli a questo:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

c. Controlla il telefono che hai usato per effettuare la sottoscrizione all'argomento SNS e conferma che il contenuto del payload del messaggio sia simile al seguente:

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

Se si modifica l'elemento ID dell'argomento nell'argomento del messaggio, tenere presente che la trasformazione del valore `topic(2)` in un valore numerico funzionerà solo se tale elemento nell'argomento del messaggio contiene solo caratteri numerici.

4. Prova a inviare un messaggio MQTT in cui la temperatura non supera il limite.

a. Nel client MQTT, sotto Subscriptions (Sottoscrizioni), scegli Publish to topic (Pubblica in un argomento).

b. Nel campo Publish (Pubblica), inserisci il nome dell'argomento di input, **`device/33/data`**.

c. Copia i dati di esempio mostrati qui e, nella casella di modifica sotto il nome dell'argomento, incolla i dati di esempio.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
```

```
    "bearing": 255
  }
}
```

d. Per inviare il messaggio MQTT, scegli Publish to topic (Pubblica in un argomento).

Il messaggio inviato dovrebbe essere visualizzato nella sottoscrizione **`device/+/data`**; tuttavia, poiché il valore della temperatura è inferiore alla temperatura massima nell'istruzione query della regola, non si dovrebbe ricevere un messaggio di testo.

Se il comportamento non viene visualizzato correttamente, controlla i suggerimenti per la risoluzione dei problemi.

Risoluzione dei problemi relativi alla AWS Lambda regola e alla notifica

Ecco alcune cose da controllare nel caso in cui non stai vedendo i risultati che ti aspetti.

- Hai un banner di errore

  Se viene visualizzato un errore quando è stato pubblicato il messaggio di input, correggilo prima. I seguenti passaggi potrebbero aiutarti a correggere l'errore.
- Il messaggio di input non viene visualizzato nel client MQTT

  Ogni volta che pubblichi il tuo messaggio di input nell'argomento `device/32/data`, tale messaggio dovrebbe essere visualizzato nel client MQTT, se hai effettuato la sottoscrizione al filtro dell'argomento `device/+/data` come descritto nella procedura.

  Controlli

  - Controlla il filtro degli argomenti a cui hai effettuato la sottoscrizione

    Se hai effettuato la sottoscrizione all'argomento del messaggio di input come descritto nella procedura, visualizzerai una copia del messaggio di input ogni volta che lo pubblichi.

    Se il messaggio non viene visualizzato, controlla il nome dell'argomento sottoscritto e confrontalo con l'argomento in cui è stato pubblicato. I nomi degli argomenti fanno distinzione tra maiuscole e minuscole e l'argomento a cui è stato sottoscritto deve essere identico all'argomento in cui hai pubblicato il payload dei messaggi.
  - Controlla la funzione di pubblicazione dei messaggi

Nel client MQTT, sotto Subscriptions (Sottoscrizioni), scegli device/+/data (dispositivo/+/dati), controlla l'argomento del messaggio di pubblicazione e quindi scegli Publish to topic (Pubblica in un argomento). Dovresti vedere il payload del messaggio dalla casella di modifica sotto l'argomento visualizzato nell'elenco dei messaggi.

- Se non ricevi un messaggio SMS

  Affinché la regola funzioni, deve disporre della policy corretta che la autorizzi a ricevere un messaggio e inviare una notifica SNS e deve ricevere il messaggio.

  Controlli

  - Controlla il Regione AWS tuo client MQTT e la regola che hai creato

    La console in cui si esegue il client MQTT deve trovarsi nella stessa regione AWS della regola creata.

  - Verifica che il valore della temperatura nel payload del messaggio superi la soglia di prova

    Se il valore della temperatura è minore o uguale a 30, come definito nell'istruzione query della regola, la regola non eseguirà alcuna delle operazioni.

  - Controlla l'argomento del messaggio di input nell'istruzione query della regola

    Affinché la regola funzioni, è necessario che venga visualizzato un messaggio con il nome dell'argomento corrispondente al filtro dell’argomento nella clausola FROM dell'istruzione query della regola.

    Controlla l'ortografia del filtro argomento nell'istruzione query regola con quella dell'argomento nel client MQTT. I nomi degli argomenti fanno distinzione tra maiuscole e minuscole e l'argomento del messaggio deve corrispondere al filtro argomento nell'istruzione query della regola.

  - Controllare il contenuto del payload del messaggio di input

    Affinché la regola funzioni, è necessario trovare il campo dati nel payload del messaggio dichiarato nell'istruzione SELECT.

    Controlla l'ortografia del campo `temperature` nell'istruzione query della regola con quella del payload del messaggio nel client MQTT. I nomi dei campi fanno distinzione tra maiuscole e minuscole e il campo `temperature`nell'istruzione query della regola deve essere identico a quello del campo `temperature` nel payload del messaggio.

Assicurati che il documento JSON nel payload del messaggio sia formattato correttamente. Se il JSON presenta errori, come una virgola mancante, la regola non sarà in grado di leggerlo.

- Controlla le notifiche Amazon SNS

  In Fase 1: creare un argomento Amazon SNS che invii un messaggio di testo SMS, fai riferimento al passaggio 3 che descrive come testare la notifica Amazon SNS e testa la notifica per assicurarti che la notifica funzioni.

- Controlla la funzione Lambda

  In Passaggio 1: creare una AWS Lambda funzione che invii un messaggio di testo, fai riferimento al passaggio 5 che descrive come testare la funzione Lambda utilizzando i dati di test e testa la funzione Lambda.

- Controlla il ruolo utilizzato dalla regola

  L'operazione della regola deve disporre dell'autorizzazione per ricevere l'argomento originale e pubblicare il nuovo argomento.

  Le policy che autorizzano la regola a ricevere i dati dei messaggi e a ripubblicarli sono specifiche degli argomenti utilizzati. Se si modifica l'argomento utilizzato per ripubblicare i dati del messaggio, è necessario aggiornare il ruolo dell'operazione della regola per aggiornare la policy in modo che corrisponda all'argomento corrente.

  Se si sospetta che questo sia il problema, modificare l'operazione Ripubblica regola e creare un nuovo ruolo. I nuovi ruoli creati dall'operazione della regola ricevono le autorizzazioni necessarie per eseguire queste operazioni.

Fase 4: Esamina i risultati e i passaggi successivi

In questo tutorial:

- Hai creato una AWS IoT regola per chiamare una funzione Lambda che ha inviato una notifica Amazon SNS utilizzando il payload di messaggi personalizzato.
- È stata utilizzata una semplice query SQL e funzioni in un'istruzione di query della regola per creare un nuovo payload dei messaggi per la funzione Lambda.
- Hai usato il client MQTT per testare la tua regola. AWS IoT

Fasi successive

Dopo aver inviato alcuni messaggi di testo con questa regola, prova a sperimentarla per vedere come la modifica di alcuni aspetti del tutorial influisce sul messaggio e quando viene inviato. Ecco alcune idee per iniziare.

- *device_id*Modificate l'argomento del messaggio di input e osservate l'effetto nel contenuto del messaggio di testo.
- Modifica i campi selezionati nell'istruzione query della regola, aggiorna la funzione Lambda per utilizzarli in un nuovo messaggio e osserva l'effetto nel contenuto del messaggio di testo.
- Modifica il test nell'istruzione query della regola per verificare una temperatura minima anziché una temperatura massima. Aggiorna la funzione Lambda per formattare un nuovo messaggio e ricordati di cambiare il nome di `max_temperature`.
- Per ulteriori informazioni su come trovare gli errori che potrebbero verificarsi durante lo sviluppo e l'utilizzo AWS IoT delle regole, consultaMonitoraggio AWS IoT.

Mantenimento dello stato del dispositivo mentre il dispositivo è disconnesso da Device Shadows

Questi tutorial mostrano come utilizzare il servizio AWS IoT Device Shadow per archiviare e aggiornare le informazioni sullo stato di un dispositivo. Il documento Shadow, che è un documento JSON, mostra la modifica dello stato del dispositivo in base ai messaggi pubblicati da un dispositivo, un'app locale o un servizio. In questo tutorial, il documento Shadow mostra la modifica del colore di una lampadina. Questi tutorial mostrano anche come Shadow archivia queste informazioni anche quando il dispositivo è disconnesso da Internet e trasmette le informazioni più recenti sullo stato al dispositivo quando torna online e richiede queste informazioni.

Ti consigliamo di provare questi tutorial nell'ordine in cui sono mostrati qui, a partire dalle risorse AWS IoT necessarie per creare e la configurazione hardware necessaria, che consente anche di apprendere i concetti in modo incrementale. Questi tutorial mostrano come configurare e connettere un dispositivo Raspberry Pi da utilizzare con. AWS IoT Se non si dispone dell'hardware necessario, è possibile seguire questi tutorial adattandoli a un dispositivo di propria scelta o con la creazione di un dispositivo virtuale con Amazon EC2.

Panoramica dello scenario tutorial

Lo scenario di questi tutorial è un'app o un servizio locale che cambia il colore di una lampadina e che pubblica i dati in argomenti shadow riservati. Questi tutorial sono simili alla funzionalità Device Shadow descritta in Interactive getting started tutorial (Tutorial interattivo sulle nozioni di base) e

sono implementati su un dispositivo Raspberry Pi. I tutorial in questa sezione si concentrano su una singola shadow classica, mostrando il modo in cui è possibile inserire le shadow o più dispositivi.

I seguenti tutorial ti aiuteranno a imparare a usare il servizio AWS IoT Device Shadow.

- Tutorial: Preparazione del Raspberry Pi per eseguire l'applicazione shadow

  Questo tutorial mostra come configurare un dispositivo Raspberry Pi con cui connettersi. AWS IoT Potrai anche creare un documento di AWS IoT policy e una risorsa, scaricare i certificati e quindi allegare la policy a quella risorsa. Questo tutorial dura circa 30 minuti.
- Tutorial: Installare l'SDK di dispositivo ed eseguire l'applicazione di esempio per Device Shadows

  Questo tutorial mostra come installare gli strumenti, il software e il AWS IoT Device SDK for Python necessari, quindi eseguire l'applicazione shadow di esempio. Questo tutorial si basa sui concetti presentati in Connettere un Raspberry Pi o altro dispositivo e il completamento richiede 20 minuti.
- Tutorial: Interagisci con Device Shadow utilizzando l'app di esempio e il client di test MQTT

  Questo tutorial mostra come utilizzare l'app e la AWS IoT console di `shadow.py` esempio per osservare l'interazione tra AWS IoT Device Shadows e i cambiamenti di stato della lampadina. Il tutorial mostra anche come inviare messaggi MQTT agli argomenti riservati di Device Shadow. Questo tutorial dura circa 45 minuti.

AWS IoT Panoramica di Device Shadow

Un Device Shadow è una rappresentazione virtuale persistente di un dispositivo gestito da una risorsa oggetto creata nel AWS IoT registro. Il documento Shadow è un documento JSON o di JavaScript notazione utilizzato per archiviare e recuperare le informazioni sullo stato corrente di un dispositivo. È possibile utilizzare lo shadow per ottenere e impostare lo stato di un dispositivo tramite argomenti MQTT o HTTP REST APIs, indipendentemente dal fatto che il dispositivo sia connesso a Internet.

Il documento di una copia shadow contiene una proprietà `state` che descrive questi aspetti dello stato del dispositivo.

- `desired`: le app specificano gli stati desiderati delle proprietà del dispositivo aggiornando l'oggetto `desired`.
- `reported`: i dispositivi segnalano il loro stato corrente nell'oggetto `reported`.
- `delta`: AWS IoT riporta le differenze tra lo stato desiderato e quello riportato nell'`delta`oggetto.

Di seguito è illustrato un esempio di documento di uno stato Shadow.

```
{
  "state": {
    "desired": {
      "color": "green"
      },
    "reported": {
      "color": "blue"
      },
    "delta": {
      "color": "green"
      }
   }
}
```

Per aggiornare il documento Shadow di un dispositivo, è possibile utilizzare gli argomenti MQTT riservati, il Device Shadow REST APIs che supporta `GETUPDATE`, `DELETE` le operazioni con HTTP e la AWS IoT CLI.

Nell'esempio precedente, supponiamo che tu voglia modificare il colore `desired` con `yellow`. A tale scopo, invia una richiesta all'API UpdateThingShadow o pubblica un messaggio nell'argomento Aggiorna, `$aws/things/THING_NAME/shadow/update`.

```
{
  "state": {
    "desired": {
      "color": yellow
    }
  }
}
```

Gli aggiornamenti interessano solo i campi specificati nella richiesta. Dopo aver aggiornato con successo il Device Shadow, AWS IoT pubblica il nuovo `desired` stato sull'`delta`argomento,`$aws/things/THING_NAME/shadow/delta`. Il documento Shadow in questo caso ha il seguente aspetto:

```
{
  "state": {
    "desired": {
      "color": yellow
```

```
    },
    "reported": {
      "color": green
    },
    "delta": {
      "color": yellow
      }
  }
}
```

Il nuovo stato viene quindi segnalato al AWS IoT Device Shadow utilizzando l'`Update`argomento `$aws/things/THING_NAME/shadow/update` con il seguente messaggio JSON:

```
{
  "state": {
    "reported": {
      "color": yellow
    }
  }
}
```

Se desideri ottenere le informazioni sullo stato attuale, invia una richiesta all'API GetThingShadow o pubblica un messaggio MQTT nell'argomento Ottieni, `$aws/things/THING_NAME/shadow/get`.

Per ulteriori informazioni sul funzionamento del Servizio Device Shadow, consulta AWS IoT Servizio Device Shadow.

Per ulteriori informazioni sull'utilizzo di Device Shadows su dispositivi, app e servizi, consulta Utilizzo delle copie shadow nei dispositivi e Utilizzo delle copie shadow in app e servizi.

Per informazioni sull'interazione con le AWS IoT ombre, consulta. Interazione con le copia shadow

Per informazioni sugli argomenti riservati MQTT e HTTP REST APIs, vedere e. Argomenti MQTT di Device Shadow API REST del servizio Device Shadow

Tutorial: Preparazione del Raspberry Pi per eseguire l'applicazione shadow

Questo tutorial dimostra come impostare e configurare un dispositivo Raspberry Pi e creare le AWS IoT risorse necessarie a un dispositivo per connettersi e scambiare messaggi MQTT.

> **Note**
>
> Se hai intenzione di the section called “Crea un dispositivo virtuale con Amazon EC2”, è possibile saltare questa pagina e continuare su the section called “Configurazione del dispositivo”. Creerai queste risorse quando creerai l’oggetto virtuale. Se desideri utilizzare un dispositivo diverso invece di Raspberry Pi, puoi provare a seguire questi tutorial adattandoli a un dispositivo di tua scelta.

In questo tutorial, imparerai come:

- Configura un dispositivo Raspberry Pi e configuralo per l'uso con. AWS IoT
- Crea un documento di AWS IoT policy che autorizzi il tuo dispositivo a interagire con AWS IoT i servizi.
- Crea una risorsa nei certificati AWS IoT del dispositivo X.509, quindi allega il documento relativo alla policy.

  L’oggetto è la rappresentazione virtuale del dispositivo nel registro AWS IoT . Il certificato autentica il dispositivo su AWS IoT Core e il documento di policy autorizza il dispositivo a interagire con. AWS IoT

Come eseguire questo tutorial

Per eseguire l’applicazione di esempio `shadow.py` per Device Shadows, avrai bisogno di un dispositivo Raspberry Pi che si connette a AWS IoT. Si consiglia di seguire questo tutorial nell'ordine in cui viene presentato qui, iniziando con la configurazione del Raspberry Pi e dei suoi accessori, e quindi creando una policy e allegando il criterio a una risorsa dell’oggetto che si crea. Puoi quindi seguire questo tutorial utilizzando l'interfaccia utente grafica (GUI) supportata da Raspberry Pi per aprire la AWS IoT console sul browser web del dispositivo, il che semplifica anche il download dei certificati direttamente sul tuo Raspberry Pi per la connessione. AWS IoT

Prima di iniziare questo tutorial, assicurati di disporre di:

- Un. Account AWS Se non disponi dell'account, effettua la procedura descritta in Configurare Account AWS, prima di continuare. Avrai bisogno della tua Account AWS AWS IoT console per completare questo tutorial.
- Il Raspberry Pi e i suoi accessori necessari. Sarà necessario:

- Un modello Raspberry Pi 3 B o più recente. Questo tutorial potrebbe funzionare su versioni precedenti del Raspberry Pi, ma non sono state testate.
- Sistema operativo Raspberry Pi (32 bit) o versione successiva. Ti consigliamo di utilizzare sempre la versione più recente del sistema operativo Raspberry Pi. Le versioni precedenti del sistema operativo potrebbero funzionare, ma non le abbiamo testate.
- Una connessione Ethernet o Wi-Fi.
- Tastiera, mouse, monitor, cavi e alimentatori.

Questo tutorial dura circa 30 minuti.

Fase 1: Imposta e configura il dispositivo Raspberry Pi

In questa sezione, configureremo un dispositivo Raspberry Pi da utilizzare con AWS IoT.

> **Important**
>
> Adattare queste istruzioni ad altri dispositivi e sistemi operativi può essere difficile. Dovrai avere una buona capacità di utilizzare il tuo dispositivo in modo da poter interpretare queste istruzioni e applicarle ad esso. In caso di difficoltà, è possibile provare una delle altre opzioni del dispositivo come alternativa, come Crea un dispositivo virtuale con Amazon EC2 o Usa il tuo PC o Mac Windows o Linux come dispositivo AWS IoT.

Dovrai configurare il tuo Raspberry Pi in modo che possa avviare il sistema operativo (OS), connettersi a Internet e consentirti di interagire con esso tramite un'interfaccia a riga di comando. Puoi anche utilizzare l'interfaccia utente grafica (GUI) supportata con Raspberry Pi per aprire la AWS IoT console ed eseguire il resto di questo tutorial.

Per impostare Raspberry Pi

1. Inserire la scheda SD nello slot per schede MicroSD nel Raspberry Pi. Alcune schede SD sono precaricate con un gestore di installazione che richiede tramite un menu di installare il sistema operativo dopo l'avvio del modulo. È inoltre possibile utilizzare l'imager Raspberry Pi per installare il sistema operativo sulla scheda.
2. Connetti un televisore HDMI o un monitor al cavo HDMI che si collega alla porta HDMI del Raspberry Pi.

3. Connetti la tastiera e il mouse alle porte USB del Raspberry Pi, quindi collega l'adattatore di alimentazione per avviare la scheda.

Dopo l'avvio del Raspberry Pi, se la scheda SD è stata precaricata con il gestore di installazione, viene visualizzato un menu per installare il sistema operativo. Se hai problemi nell'installazione del sistema operativo, puoi provare a eseguire la procedura riportata di seguito. Per informazioni sull'impostazione di Raspberry Pi, consulta Impostazione di Raspberry Pi.

In caso di problemi nell'impostazione di Raspberry Pi:

- Verifica se è stata inserita la scheda SD prima di avviare la scheda. Se si collega la scheda SD dopo l'avvio del modulo, il menu di installazione potrebbe non essere visualizzato.
- Assicurati che il televisore o il monitor sia acceso e che sia selezionato l'ingresso corretto.
- Assicurati di utilizzare il software compatibile con Raspberry Pi.

Dopo aver installato e configurato il sistema operativo Raspberry Pi, apri il browser Web di Raspberry Pi e accedi alla AWS IoT Core console per continuare il resto dei passaggi di questo tutorial.

Se riesci ad aprire la AWS IoT Core console, il tuo Raspberry Pi è pronto e puoi continuare a farlo. the section called “Eseguire il provisioning del dispositivo in AWS IoT ”

In caso di problemi o di bisogno di ulteriore assistenza, consulta la sezione Getting help for your Raspberry Pi (Ottenere assistenza per Raspberry Pi).

Tutorial: Eseguire il provisioning del dispositivo in AWS IoT

Questa sezione crea le AWS IoT Core risorse che verranno utilizzate dal tutorial.

Passaggi per effettuare il provisioning del dispositivo AWS IoT

- Fase 1: Creare una AWS IoT policy per il Device Shadow
- Fase 2: creare una risorsa dell'oggetto e connettere la policy all'oggetto
- Fase 3: Esamina i risultati e i passaggi successivi

Fase 1: Creare una AWS IoT policy per il Device Shadow

I certificati X.509 autenticano il dispositivo con. AWS IoT Core AWS IoT al certificato sono allegate politiche che consentono al dispositivo di eseguire AWS IoT operazioni, come la sottoscrizione o

la pubblicazione di argomenti riservati MQTT utilizzati dal servizio Device Shadow. Il dispositivo presenta il certificato quando si connette e invia messaggi a. AWS IoT Core

In questa procedura creerai un policy che consente di eseguire le operazioni AWS IoT necessarie per eseguire il programma di esempio. Ti consigliamo di creare una policy che conceda solo le autorizzazioni richieste per eseguire l'attività. Prima crei la AWS IoT policy e poi la alleghi al certificato del dispositivo che creerai in seguito.

Per creare una AWS IoT politica

1. Nel menu di sinistra scegli Secure (Sicurezza) e quindi Policies (Policy). Se il tuo account dispone di policy esistenti, scegli Create (Crea); in caso contrario, nella pagina You don't have a policy yet (Non hai ancora una policy) scegli Create a policy (Crea una policy).
2. Nella pagina Create a policy (Crea una policy):

   a. Nel campo Name (Nome) inserire un nome per la policy (ad esempio **`My_Device_Shadow_policy`**). Non utilizzare dati personali identificabili nei nomi delle policy.

   b. Nel documento di policy vengono descritte le operazioni di connessione, sottoscrizione, ricezione e pubblicazione che consentono al dispositivo l'autorizzazione di pubblicare e sottoscrivere gli argomenti riservati MQTT.

      Copia la policy di esempio seguente e incollala nel documento di policy. Sostituiscilo `thingname` con il nome dell'oggetto che creerai (ad esempio,`My_light_bulb`), `region` con la AWS IoT regione in cui utilizzi i servizi e `account` con il tuo Account AWS numero. Per ulteriori informazioni sulle AWS IoT politiche, consultaAWS IoT Core politiche.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/get",
```

```
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/get/rejected",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update/delta"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/get/rejected",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/update/delta"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Connect",
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/test-*"
```

```
        }
    ]
}
```

Fase 2: creare una risorsa dell'oggetto e connettere la policy all'oggetto

I dispositivi collegati a AWS IoT possono essere rappresentati dalle risorse degli oggetti presenti nel AWS IoT registro. Una risorsa dell'oggetto rappresenta un dispositivo specifico o un'entità logica, come la lampadina in questo tutorial.

Per imparare a creare un oggetto in AWS IoT, segui i passaggi descritti inCrea un oggetto. Ecco alcune cose fondamentali da notare mentre segui i passaggi di tale tutorial:

1. Scegli Create a single thing (Crea un oggetto singolo), e nel campo Name (Nome) inserisci un nome per l'oggetto che è uguale a quello del `thingname` (ad esempio, `My_light_bulb`) che hai specificato quando hai creato la policy in precedenza.

   Non è possibile modificare il nome di un oggetto dopo averlo creato. Se gli hai dato un nome diverso da `thingname`, crea un nuovo oggetto con nome `thingname` ed elimina il vecchio oggetto.

   > **Note**
   >
   > Non utilizzare dati personali identificabili nel nome dell'oggetto. Il nome dell'oggetto può essere visualizzato nelle comunicazioni e nei report non crittografati.

2. Si consiglia di scaricare ciascuno dei file di certificato nella pagina Certificato creato! in una posizione in cui è possibile trovarli facilmente. È necessario installare questi file per eseguire l'applicazione di esempio.

   Ti consigliamo di scaricare i file in una sottodirectory `certs` nella directory home sul Raspberry Pi e nominare ciascuno di essi con un nome più semplice come suggerito nella tabella seguente.

   Nomi dei file dei certificati

File	Percorso del file
Un certificato emesso da una CA root	`~/certs/Amazon-root-CA-1.pem`

File	Percorso del file
Certificato del dispositivo	`~/certs/device.pem.crt`
Chiave privata	`~/certs/private.pem.key`

3. Dopo aver attivato il certificato a cui abilitare le connessioni AWS IoT, scegli Allega una politica e assicurati di allegare la politica che hai creato in precedenza (ad esempio**`My_Device_Shadow_policy`**) all'oggetto.

   Dopo aver creato un oggetto, puoi vedere la risorsa relativa all'oggetto visualizzata nell'elenco degli elementi nella AWS IoT console.

Fase 3: Esamina i risultati e i passaggi successivi

In questo tutorial, hai appreso come:

- Impostare e configurare il dispositivo Raspberry Pi.
- Crea un documento di AWS IoT policy che autorizzi il tuo dispositivo a interagire con AWS IoT i servizi.
- Creare una risorsa dell'oggetto e il certificato del dispositivo X.509 associato e collegare il documento della policy ad esso.

Fasi successive

Ora puoi installare l'SDK del AWS IoT dispositivo per Python, eseguire `shadow.py` l'applicazione di esempio e utilizzare Device Shadows per controllare lo stato. Per ulteriori informazioni su come eseguire questo tutorial, consulta Tutorial: Installare l'SDK di dispositivo ed eseguire l'applicazione di esempio per Device Shadows.

Tutorial: Installare l'SDK di dispositivo ed eseguire l'applicazione di esempio per Device Shadows

Questa sezione mostra come installare il software richiesto e AWS IoT Device SDK for Python ed eseguire `shadow.py` l'applicazione di esempio per modificare il documento Shadow e controllare lo stato dell'ombra.

In questo tutorial, apprenderai come:

- Usa il software installato e AWS IoT Device SDK for Python per eseguire l'app di esempio.
- Scoprire come l'immissione di un valore utilizzando l'app di esempio pubblichi il valore desiderato nella console AWS IoT .
- Esaminare l'applicazione di esempio `shadow.py` e come utilizzare il protocollo MQTT per aggiornare lo stato di shadow.

Prima di eseguire questo tutorial:

È necessario aver configurato Account AWS, configurato il dispositivo Raspberry Pi e creato una AWS IoT cosa e una politica che consentano al dispositivo le autorizzazioni per pubblicare e sottoscrivere gli argomenti riservati MQTT del servizio Device Shadow. Per ulteriori informazioni, consulta Tutorial: Preparazione del Raspberry Pi per eseguire l'applicazione shadow.

Devi aver installato anche Git, Python e AWS IoT Device SDK for Python. Questo tutorial si basa sui concetti presentati nel tutorial Connettere un Raspberry Pi o altro dispositivo. Se non hai provato questo tutorial, ti consigliamo di seguire i passaggi descritti in tale tutorial per installare i file dei certificati e SDK di dispositivo e quindi tornare a questo tutorial per eseguire l'applicazione di esempio `shadow.py`.

In questo tutorial, dovrai:

- Fase 1: eseguire l'app di esempio shadow.py
- Fase 2: Revisione dell'app di esempio shadow.py dell'SDK di dispositivo
- Fase 3: Risolvi i problemi con l'app di esempio shadow.py
- Fase 4: Esamina i risultati e i passaggi successivi

Questo tutorial dura circa 20 minuti.

Fase 1: eseguire l'app di esempio shadow.py

Prima di eseguire il comando l'app di esempio `shadow.py`, oltre ai nomi e alla posizione dei file di certificato installati, sono necessarie le seguenti informazioni.

Valori dei parametri dell'applicazione

Parametro	Dove trovare il valore
your-iot-thing-name	Nome della AWS IoT cosa che hai creato in precedenza in. the section called “Fase 2: creare una risorsa dell’oggetto e connettere la policy all’oggetto” Per trovare questo valore, nella console AWS IoT, scegli Manage (Gestione), quindi scegli Things (Oggetti).
your-iot-endpoint	Il *your-iot-endpoint* valore ha un formato di:*endpoint_id* `-ats.iot.` *region*`.amazonaws.com` , ad esempio,`a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com` . Per trovare questo valore: 1. Nella console AWS IoT, scegli Manage (Gestione), quindi scegli Things (Oggetti). 2. Scegli l’oggetto IoT che hai creato per il tuo dispositivo, my_light_lampadina, che hai utilizzato in precedenza, quindi scegli Interact (Interagisci). L'endpoint viene visualizzato nella sezione HTTPS della pagina dei dettagli degli oggetti.

Installazione ed esecuzione dell'applicazione di esempio

1. Passare alla directory dell’applicazione di esempio.

```
cd ~/aws-iot-device-sdk-python-v2/samples/service-clients
```

2. Nella finestra della riga di comando, sostituisci *your-iot-endpoint* e *your-iot-thing-name* come indicato ed esegui questo comando.

```
python3 shadow.py --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/
device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint --
thing_name your-iot-thing-name
```

3. Osserva che l'applicazione di esempio:

   1. Si connette al servizio AWS IoT per il tuo account.
   2. Effettui la sottoscrizione ad eventi `Delta` e alle risposte `Update` e `Get`.
   3. Richieda di inserire un valore desiderato nel terminale.
   4. L'output sia simile a quello riportato di seguito:

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to Delta events...
Subscribing to Update responses...
Subscribing to Get responses...
Requesting current shadow state...
Launching thread to read user input...
Finished getting initial shadow state.
Shadow contains reported value 'off'.
Enter desired value:
```

> **Note**
>
> In caso di problemi nell'esecuzione dell'app di esempio `shadow.py`, consulta the section called “Fase 3: Risolvi i problemi con l'app di esempio `shadow.py`”. Per ottenere ulteriori informazioni che potrebbero essere utili per risolvere il problema, aggiungi il parametro `--verbosity debug` alla riga di comando in modo che l'app di esempio visualizzi messaggi dettagliati su ciò che sta facendo.

Inserisci i valori e osserva gli aggiornamenti nel documento Shadow

È possibile inserire i valori nel terminale per specificare il valore `desired`, che aggiorna anche il valore `reported`. Supponiamo di inserire il colore `yellow` nel terminale. Il valore `reported` viene aggiornato anche al colore `yellow`. Di seguito sono riportati i messaggi visualizzati nel terminale:

```
Enter desired value:
yellow
Changed local shadow value to 'yellow'.
Updating reported shadow value to 'yellow'...
Update request published.
Finished updating reported shadow value to 'yellow'.
```

Quando pubblichi questa richiesta di aggiornamento, AWS IoT crea un'ombra classica predefinita per la risorsa thing. È possibile osservare la richiesta di aggiornamento pubblicata su `reported` e i `desired` valori nella AWS IoT console esaminando il documento Shadow relativo alla risorsa oggetto che è stata creata (ad esempio,`My_light_bulb`). Per visualizzare l'aggiornamento nel documento Shadow:

1. Nella AWS IoT console, scegli Gestisci, quindi scegli Cose.
2. Nell'elenco di oggetti visualizzati, seleziona l'oggetto che hai creato, scegli Shadows, quindi scegli Classic Shadow (Shadow classico).

Il documento Shadow dovrebbe essere simile a quanto riportato di seguito, mostrando i valori `reported` e `desired` impostati sul colore `yellow`. Questi valori vengono visualizzati nello Stato shadow del documento.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "yellow"
},
"reported": {
  "welcome": "aws-iot",
  "color": "yellow"
}
}
```

Viene inoltre visualizzato una sezione Metadata (Metadati) che contiene le informazioni sul timestamp e il numero di versione della richiesta.

Puoi usare la versione del documento sullo stato per assicurarti di aggiornare la versione più recente di una copia shadow di un dispositivo. Se invii un'altra richiesta di aggiornamento, il numero di versione aumenta di 1. Quando fornisci una versione con una richiesta di aggiornamento, il

servizio rifiuta la richiesta con un codice di risposta di conflitto HTTP 409 se la versione corrente del documento sullo stato non corrisponde alla versione fornita.


```
{
"metadata": {
  "desired": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620156893
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620156893
    }
  }
},
"version": 10
}
```


Per saperne di più sul documento Shadow e osservare le modifiche alle informazioni sullo stato, procedi con il tutorial successivo Tutorial: Interagisci con Device Shadow utilizzando l'app di esempio e il client di test MQTT come descritto nella sezione Fase 4: Esamina i risultati e i passaggi successivi di questo tutorial. Eventualmente, si può anche sapere di più sul codice di esempio `shadow.py` e su come utilizza il protocollo MQTT nella sezione seguente.

Fase 2: Revisione dell'app di esempio shadow.py dell'SDK di dispositivo

Questa sezione esamina l'app di esempio `shadow.py` dal SDK di dispositivo v2 AWS IoT per Python, usata in questo tutorial. Qui esamineremo come si connette AWS IoT Core utilizzando il protocollo MQTT e MQTT su WSS. La libreria AWS Common Runtime (AWS-CRT) fornisce il supporto del protocollo di comunicazione di basso livello ed è inclusa in AWS IoT Device SDK v2 for Python.

Sebbene questo tutorial utilizzi MQTT e MQTT su WSS, supporta i dispositivi che pubblicano richieste HTTPS. AWS IoT Per un esempio di un programma Python che invia un messaggio HTTP da un dispositivo, vedi l'esempio di codice HTTPS utilizzando la libreria di Python `requests`.

Per informazioni su come prendere una decisione informata sul protocollo da utilizzare per le comunicazioni del dispositivo, consulta Scelta di un protocollo applicativo per la comunicazione del dispositivo.

MQTT

L'esempio `shadow.py` chiama `mtls_from_path` (mostrato qui) in `mqtt_connection_builder` per stabilire una connessione con AWS IoT Core utilizzando il protocollo MQTT. `mtls_from_path` utilizza i certificati X.509 e TLS v1.2 per autenticare il dispositivo. La libreria AWS-CRT gestisce i dettagli di livello inferiore di quella connessione.

```
mqtt_connection = mqtt_connection_builder.mtls_from_path(
  endpoint=args.endpoint,
  cert_filepath=args.cert,
  pri_key_filepath=args.key,
  ca_filepath=args.ca_file,
  client_bootstrap=client_bootstrap,
  on_connection_interrupted=on_connection_interrupted,
  on_connection_resumed=on_connection_resumed,
  client_id=args.client_id,
  clean_session=False,
  keep_alive_secs=6
)
```

- `endpoint`è l' AWS IoT endpoint che hai passato dalla riga di comando ed `client_id` è l'ID che identifica in modo univoco questo dispositivo in. Regione AWS
- `cert_filepath`, `pri_key_filepath` e `ca_filepath` sono i percorsi del certificato e dei file di chiave privata del dispositivo e del file root CA.
- `client_bootstrap` è l'oggetto runtime comune che gestisce le attività di comunicazione socket e viene istanziato prima della chiamata a `mqtt_connection_builder.mtls_from_path`.
- `on_connection_interrupted` e `on_connection_resumed` sono le funzioni di callback da chiamare quando la connessione del dispositivo viene interrotta e ripresa.
- `clean_session` consente di avviare una nuova sessione persistente o, se è presente, di riconnettersi a una esistente. `keep_alive_secs` è il valore keep alive, in secondi, per inviare la

richiesta `CONNECT`. Un ping verrà inviato automaticamente a questo intervallo. Se il server non riceve un ping dopo 1,5 volte questo valore, presuppone che la connessione sia stata persa.

L'esempio `shadow.py` chiama anche `websockets_with_default_aws_signing` in `mqtt_connection_builder` per stabilire una connessione con AWS IoT Core utilizzando il protocollo MQTT su WSS. MQTT su WSS utilizza anche gli stessi parametri di MQTT e prende questi parametri aggiuntivi:

- `region`è la regione di AWS firma utilizzata dall'autenticazione Signature V4 e `credentials_provider` sono AWS le credenziali fornite da utilizzare per l'autenticazione. La regione viene passata dalla riga di comando, e l'oggetto `credentials_provider` viene istanziato appena prima della chiamata a `mqtt_connection_builder.websockets_with_default_aws_signing`.
- `websocket_proxy_options` sono le opzioni proxy HTTP, se si utilizza un host proxy. Nell'app di esempio `shadow.py`, questo valore viene istanziato appena prima della chiamata a `mqtt_connection_builder.websockets_with_default_aws_signing`.

Iscrizione agli argomenti e agli eventi Shadow

L'esempio `shadow.py` cerca di stabilire una connessione e attende di essere completamente connesso. Se non è connesso, i comandi vengono messi in coda. Una volta connesso, l'esempio sottoscrive gli eventi delta e aggiorna e riceve i messaggi e pubblica i messaggi con un livello Quality of Service (QoS) pari a 1 (`mqtt.QoS.AT_LEAST_ONCE`).

Quando un dispositivo sottoscrive un messaggio con QoS livello 1, il broker di messaggi salva i messaggi a cui il dispositivo è sottoscritto fino a quando non possono essere inviati al dispositivo. Il broker di messaggi restituisce i messaggi fino a quando non riceve una risposta `PUBACK` dal dispositivo.

Per ulteriori informazioni sul protocollo MQTT, consulta Esamina il protocollo MQTT e MQTT.

Per ulteriori informazioni su MQTT, MQTT su WSS, sessioni persistenti e i livelli QoS utilizzati in questa esercitazione, consulta Revisione dell'app di esempio pubsub.py SDK per dispositivi.

Fase 3: Risolvi i problemi con l'app di esempio **`shadow.py`**

Quando si esegue l'app di esempio `shadow.py`, dovresti vedere alcuni messaggi visualizzati nel terminale e un prompt per inserire un valore `desired`. Se il programma genera un errore, per

eseguire il debug dell'errore, puoi iniziare controllando se hai eseguito il comando corretto per il tuo sistema.

In alcuni casi, il messaggio di errore potrebbe indicare problemi di connessione e avere un aspetto simile a: `Host name was invalid for dns resolution` o `Connection was closed unexpectedly`. In questi casi, ecco alcune operazioni disponibili:

- Controlla l'indirizzo dell'endpoint nel comando

  Esaminare il parametro `endpoint` nel comando inserito per eseguire l'app di esempio (ad esempio `a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`) e controllare questo valore nella console AWS IoT .

  Per verificare se hai utilizzato il valore corretto:

  1. Nella console AWS IoT , scegli Manage (Gestione) e poi Things (Oggetti).
  2. Scegli l'oggetto che hai creato per l'app di esempio (ad esempio, my_light_lampadina) e quindi scegli Interact (Interagisci).

  L'endpoint viene visualizzato nella sezione HTTPS della pagina dei dettagli degli oggetti. Dovresti visualizzare anche un messaggio in cui viene indicato: `This thing already appears to be connected.`

- Verifica l'attivazione del certificato

  I certificati autenticano il dispositivo con. AWS IoT Core

  Per verificare se il certificato è attivo:

  1. Nella console AWS IoT , scegli Manage (Gestione) e poi Things (Oggetti).
  2. Scegli l'oggetto che hai creato per l'app di esempio (ad esempio, my_light_lampadina) e quindi scegli Security (Sicurezza).
  3. Seleziona il certificato e quindi, dalla pagina dei dettagli del certificato, scegli Seleziona il certificato e quindi, dalla pagina dei dettagli del certificato, scegli Actions (Operazioni).

  Se nell'elenco a discesa Activate (Attiva) non è disponibile e puoi solo scegliere Deactivate (Disattiva), il certificato è attivo. In caso contrario, scegli Activate (Attiva) ed esegui di nuovo il programma di esempio.

  Se il programma continua a non essere eseguito, controlla i nomi dei file del certificato nel folder `certs`.

- Controlla la policy associata alla risorsa dell'oggetto

  Mentre i certificati autenticano il dispositivo, AWS IoT le policy consentono al dispositivo di eseguire AWS IoT operazioni, come la sottoscrizione o la pubblicazione di argomenti riservati MQTT.

  Per verificare se la policy corretta è allegata:

  1. Individua il certificato come descritto in precedenza, quindi scegli Policies (Policy).
  2. Scegli la policy visualizzata e verifica se descrive le operazioni `connect`, `subscribe`, `receive` e `publish` che consentono al dispositivo l'autorizzazione di pubblicare e sottoscrivere gli argomenti riservati MQTT.

     Consulta Fase 1: Creare una AWS IoT policy per il Device Shadow per una policy di esempio.

  Se visualizzi messaggi di errore che indicano problemi di connessione AWS IoT, ciò potrebbe essere dovuto alle autorizzazioni che stai utilizzando per la politica. In tal caso, ti consigliamo di iniziare con una politica che fornisca l'accesso completo alle AWS IoT risorse e quindi di eseguire nuovamente il programma di esempio. È possibile modificare la policy corrente oppure scegliere la policy corrente. Scegli Detach (Distacca) e quindi crea un altra policy che fornisca accesso completo e collegala alla risorsa dell'oggetto. In seguito è possibile limitare la policy solo alle operazioni e alle policy necessarie per eseguire il programma.

  ```
  {
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:*"
            ],
            "Resource": "*"
        }
    ]
  }
  ```

- Verifica dell'installazione di SDK di dispositivo

  Se il programma non viene ancora eseguito, è possibile reinstallare SDK di dispositivo per assicurarsi che l'installazione dell'SDK sia completa e corretta.

Fase 4: Esamina i risultati e i passaggi successivi

In questo tutorial, hai appreso come:

- Installa il software, gli strumenti e il AWS IoT Device SDK for Python necessari.
- Capire come l'applicazione di esempio ,`shadow.py`, che utilizza il protocollo MQTT per recuperare e aggiornare lo stato corrente di shadow.
- Esegui l'app di esempio per Device Shadows e osserva l'aggiornamento del documento Shadow nella console. AWS IoT Hai anche imparato a risolvere eventuali problemi e correggere gli errori durante l'esecuzione del programma.

Fasi successive

A questo punto è possibile eseguire il l'applicazione di esempio `shadow.py` e utilizzare Device Shadows per controllare lo stato. È possibile osservare gli aggiornamenti del documento Shadow nella console AWS IoT e gli eventi delta a cui risponde l'app di esempio. Utilizzando il client di test MQTT, è possibile iscriversi agli argomenti shadow riservati e osservare i messaggi ricevuti dagli argomenti durante l'esecuzione del programma di esempio. Per ulteriori informazioni su come eseguire questo tutorial, consulta Tutorial: Interagisci con Device Shadow utilizzando l'app di esempio e il client di test MQTT.

Tutorial: Interagisci con Device Shadow utilizzando l'app di esempio e il client di test MQTT

Per interagire con l'applicazione di esempio `shadow.py`, inserisci un valore nel terminale per il valore `desired`. Ad esempio, potete specificare colori simili ai semafori, AWS IoT rispondere alla richiesta e aggiornare i valori riportati.

In questo tutorial, apprenderai come:

- Utilizzare l'app di esempio `shadow.py` per specificare gli stati desiderati e aggiornare lo stato corrente di shadow.
- Modificare il documento Shadow per osservare gli eventi delta e come l'app di esempio `shadow.py` risponde ad essi.
- Utilizzare il client di test MQTT per effettuare la sottoscrizione ad argomenti shadow e osservare gli aggiornamenti quando si esegue il programma di esempio.

Prima di eseguire questo tutorial, è necessario:

Configura il tuo Account AWS, configurato il tuo dispositivo Raspberry Pi e creato AWS IoT qualcosa e una politica. È inoltre necessario aver installato il software richiesto, SDK per dispositivo, i file di certificato ed eseguire il programma di esempio nel terminale. Per ulteriori informazioni, consulta i tutorial precedenti Tutorial: Preparazione del Raspberry Pi per eseguire l'applicazione shadow e Fase 1: eseguire l'app di esempio shadow.py. Se non lo hai già fatto, devi completare questi tutorial.

In questo tutorial, dovrai:

- Fase 1: Aggiornare i valori desiderati e segnalati utilizzando l'app di esempio shadow.py
- Fase 2: Visualizzare i messaggi dall'app di esempio shadow.py nel client di test MQTT
- Fase 3: Risoluzione degli errori con le interazioni Device Shadow
- Fase 4: Esamina i risultati e i passaggi successivi

Questo tutorial dura circa 45 minuti.

Fase 1: Aggiornare i valori desiderati e segnalati utilizzando l'app di esempio **`shadow.py`**

Nel tutorial precedenteFase 1: eseguire l'app di esempio shadow.py, hai imparato a osservare un messaggio pubblicato nel documento Shadow nella AWS IoT console quando inserisci un valore desiderato, come descritto nella sezioneTutorial: Installare l'SDK di dispositivo ed eseguire l'applicazione di esempio per Device Shadows.

Nell'esempio precedente, abbiamo impostato il colore desiderato su `yellow`. Dopo aver inserito ogni valore, il terminale richiede di immettere un altro valore `desired`. Se si inserisce di nuovo lo stesso valore (`yellow`), l'app lo riconosce e ti chiede di inserire un nuovo valore `desired`.

```
Enter desired value:
yellow
Local value is already 'yellow'.
Enter desired value:
```

Ora, supponiamo che tu inserisca il colore`green`. AWS IoT risponde alla richiesta e aggiorna il `reported` valore a`green`. Questo è il modo in cui si verifica l'aggiornamento quando lo stato `desired` è diverso dal `reported`, causando un delta.

In che modo l'app di esempio **`shadow.py`** simula le interazioni Device Shadow:

1. Inserisci un valore `desired` (ad esempio `yellow`) nel terminale per pubblicare lo stato desiderato.

2. Dato che lo stato `desired` è diverso dallo stato `reported` (ad esempio il colore `green`), si verifica un delta e l'app sottoscritta al delta riceve questo messaggio.
3. L'app risponde al messaggio e aggiorna il suo stato al valore `desired`, `yellow`.
4. L'app pubblica quindi un messaggio di aggiornamento con il nuovo valore dello stato del dispositivo, `yellow`.

Di seguito vengono illustrati i messaggi visualizzati nel terminale che mostrano come viene pubblicata la richiesta di aggiornamento.

```
Enter desired value:
green
Changed local shadow value to 'green'.
Updating reported shadow value to 'green'...
Update request published.
Finished updating reported shadow value to 'green'.
```

Nella AWS IoT console, il documento Shadow riflette il valore aggiornato a `green` per entrambi i `desired` campi `reported` e e il numero di versione viene incrementato di 1. Ad esempio, se il numero della versione precedente è stato visualizzato come 10, il numero della versione corrente verrà visualizzato come 11.

> **Note**
>
> L'eliminazione di una shadow non reimposta il numero di versione su 0. Si vedrà che la versione shadow viene incrementata di 1 quando si pubblica una richiesta di aggiornamento o si crea un'altra shadow con lo stesso nome.

Modifica del documento Shadow per osservare gli eventi delta

L'app di esempio `shadow.py` è anche sottoscritta agli eventi `delta` e risponde quando c'è una modifica a un valore `desired`. Ad esempio, è possibile modificare il valore `desired` per il colore `red`. Per fare ciò, nella AWS IoT console, modifica il documento Shadow facendo clic su Modifica, quindi imposta il `desired` valore su `red` in JSON, mantenendo il `reported` valore su. `green` Prima di salvare le modifiche, tieni aperto il terminale sul Raspberry Pi mentre vedrai i messaggi visualizzati nel terminale quando si verifica la modifica.

```
{
```

```
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

Dopo aver salvato il nuovo valore, l'app di esempio `shadow.py` risponde a questa modifica e visualizza i messaggi nel terminale che indicano il delta. Si dovrebbero quindi vedere i seguenti messaggi visualizzati sotto il prompt per l'immissione del valore `desired`.

```
Enter desired value:
Received shadow delta event.
Delta reports that desired value is 'red'. Changing local value...
Changed local shadow value to 'red'.
Updating reported shadow value to 'red'...
Finished updating reported shadow value to 'red'.
Enter desired value:
Update request published.
Finished updating reported shadow value to 'red'.
```

Fase 2: Visualizzare i messaggi dall'app di esempio **`shadow.py`** nel client di test MQTT

Puoi utilizzare il client di test MQTT nella console AWS IoT per monitorare i messaggi MQTT che vengono passati in Account AWS. Utilizzando gli argomenti MQTT riservati utilizzati dal servizio Device Shadow, è possibile osservare i messaggi ricevuti dagli argomenti durante l'esecuzione dell'applicazione di esempio.

Se il client di test MQTT non è ancora stato utilizzato, è possibile consultare Visualizza i messaggi MQTT con il AWS IoT client MQTT. In questo modo si impara come utilizzare il client di test MQTT nella console AWS IoT per visualizzare i messaggi MQTT durante il passaggio attraverso il broker di messaggi.

1. Apertura del client di test MQTT

   Apri il client di test MQTT nella console AWS IoT in una nuova finestra in modo da poter osservare i messaggi ricevuti dagli argomenti MQTT senza perdere la configurazione del client di test MQTT. Il client di test MQTT non conserva sottoscrizioni o registri di messaggi se si lascia

andare a un'altra pagina della console. In questa sezione del tutorial, potete aprire il documento Shadow corrispondente e il AWS IoT client di test MQTT in finestre separate per osservare più facilmente l'interazione con Device Shadows.

2. Iscrizione agli argomenti Shadow riservati MQTT

   È possibile utilizzare il client di test MQTT per inserire i nomi degli argomenti riservati MQTT di Device Shadow e sottoscriverli per ricevere gli aggiornamenti durante l'esecuzione dell'app di esempio `shadow.py`. Per effettuare la sottoscrizione agli argomenti:

   a. Nel client di test MQTT nella console AWS IoT , scegli Subscribe to a topic (Sottoscrizione a un argomento).

   b. Nella sezione Filtro argomento, inserisci: ***thingname***$aws/things/ /shadow/update/ #. Qui, `thingname` è il nome della risorsa dell'oggetto creata in precedenza (ad esempio, `My_light_bulb`).

   c. Mantieni i valori predefiniti per le impostazioni di configurazione aggiuntive, quindi scegli Subscribe (Effettua sottoscrizione).

   Utilizzando il carattere jolly # nella sottoscrizione dell'argomento, è possibile sottoscrivere più argomenti MQTT contemporaneamente e osservare tutti i messaggi che vengono scambiati tra il dispositivo e il suo Shadow in un'unica finestra. Per ulteriori informazioni sui caratteri jolly e sul loro utilizzo, consulta Argomenti MQTT.

3. Esegui il programma di esempio **`shadow.py`** e osserva i messaggi

   Nella finestra della riga di comando di Raspberry Pi, se hai disconnesso il programma, esegui di nuovo l'app di esempio e guarda i messaggi nel client di test MQTT nella console AWS IoT .

   a. Esegui il seguente comando per riavviare il programma di esempio. Sostituisci *`your-iot-thing-name`* e *`your-iot-endpoint`* con i nomi dell' AWS IoT oggetto che hai creato in precedenza (ad esempio,`My_light_bulb`) e l'endpoint per interagire con il dispositivo.

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples/service-clients
   python3 shadow.py --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/
   device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint --
   thing_name your-iot-thing-name
   ```

L'app di esempio `shadow.py` viene eseguita e recupera lo stato shadow corrente. Se hai eliminato lo shadow o cancellato gli stati correnti, il programma imposta il valore corrente su `off` e richiede di inserire un valore `desired`.

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to Delta events...
Subscribing to Update responses...
Subscribing to Get responses...
Requesting current shadow state...
Launching thread to read user input...
Finished getting initial shadow state.
Shadow document lacks 'color' property. Setting defaults...
Changed local shadow value to 'off'.
Updating reported shadow value to 'off'...
Update request published.
Finished updating reported shadow value to 'off'...
Enter desired value:
```

D'altra parte, se il programma era in esecuzione e lo hai riavviato, vedrai il valore di colore più recente riportato nel terminale. Nel client di test MQTT, vedrai un aggiornamento agli argomenti $aws/things/ /shadow/get e ***thingname*** $aws/things//. ***thingname*** shadow/get/accepted

Supponiamo che l'ultimo colore riportato sia `green`. Di seguito viene mostrato ***thingname*** il contenuto del file JSON $aws/things/. shadow/get/accepted

```
{
"state": {
  "desired": {
    "welcome": "aws-iot",
    "color": "green"
  },
  "reported": {
    "welcome": "aws-iot",
    "color": "green"
  }
},
"metadata": {
  "desired": {
```

```
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620161643
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620161643
    }
  }
},
"version": 10,
"timestamp": 1620173908
}
```

b. Inserisci un valore `desired` nel terminale, come `yellow`. L'app di esempio `shadow.py` risponde e visualizza i seguenti messaggi nel terminale che mostrano la modifica nel valore `reported` a `yellow`.

```
Enter desired value:
yellow
Changed local shadow value to 'yellow'.
Updating reported shadow value to 'yellow'...
Update request published.
Finished updating reported shadow value to 'yellow'.
```

In MQTT test client (Client MQTT di test) nella console AWS IoT , in Subscriptions (Sottoscrizioni), viene visualizzato che i seguenti argomenti hanno ricevuto un messaggio:

- $aws/things/ /shadow/update: mostra che entrambi i valori cambiano in base al ***thingname*** colore. `desired` `updated` `yellow`
- ***thingname***$aws/things/ shadow/update/accepted: mostra i valori correnti degli stati and e e i relativi metadati e informazioni sulla versione. `desired` `reported`
- ***thingname***$aws/things/ shadow/update/documents: mostra i valori precedenti e correnti degli stati and e e i relativi metadati e informazioni sulla versione. `desired` `reported`

Poiché il documento $aws/things/***thingname***/contiene shadow/update/documents anche informazioni contenute negli altri due argomenti, possiamo esaminarlo per vedere le informazioni sullo stato. Lo stato precedente mostra il valore riportato impostato su `green`, i metadati e le informazioni sulla versione e lo stato corrente che mostra il valore riportato aggiornato a `yellow`.

```
{
"previous": {
  "state": {
    "desired": {
      "welcome": "aws-iot",
      "color": "green"
    },
    "reported": {
      "welcome": "aws-iot",
      "color": "green"
    }
  },
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297898
      }
    },
    "reported": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297898
      }
    }
  },
  "version": 10
},
"current": {
  "state": {
    "desired": {
```

```
      "welcome": "aws-iot",
      "color": "yellow"
    },
    "reported": {
      "welcome": "aws-iot",
      "color": "yellow"
    }
  },
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297904
      }
    },
    "reported": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297904
      }
    }
  },
  "version": 11
},
"timestamp": 1617297904
}
```


c. Ora, se si inserisce un altro valore `desired`, vengono visualizzate ulteriori modifiche ai valori `reported` e agli aggiornamenti dei messaggi ricevuti da questi argomenti. Anche il numero di versione aumenta di 1. Ad esempio, se si inserisce il valore `green`, lo stato precedente riporta il valore `yellow` e lo stato corrente riporta il valore `green`.

4. Modifica del documento Shadow per osservare gli eventi delta

   Per osservare le modifiche apportate all'argomento delta, modifica il documento Shadow nella console AWS IoT . Ad esempio, è possibile modificare il valore `desired` per il colore `red`. Per fare ciò, nella AWS IoT console, scegli Modifica, quindi imposta il `desired` valore su rosso nel file JSON, mantenendo il valore impostato su. `reported` `green` Prima di salvare la modifica, tieni il terminale aperto poiché vedrai il messaggio delta riportato nel terminale.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

L'app di esempio `shadow.py` risponde a questa modifica e visualizza i messaggi nel terminale che indica il delta. Nel client di test MQTT, gli argomenti `update` avranno ricevuto un messaggio che mostra le modifiche apportate ai valori `desired` e `reported`.

Vedi anche che l'argomento ***thingname***$aws/things//ha ricevuto un messaggio. shadow/update/delta Per visualizzare il messaggio, scegli questo argomento, elencato in Subscriptions (Sottoscrizioni).

```
{
"version": 13,
"timestamp": 1617318480,
"state": {
  "color": "red"
},
"metadata": {
  "color": {
    "timestamp": 1617318480
  }
}
}
```

Fase 3: Risoluzione degli errori con le interazioni Device Shadow

Quando esegui l'app di esempio Shadow, potresti riscontrare problemi con l'osservazione delle interazioni con il servizio Device Shadow.

Se il programma viene eseguito correttamente e richiede di inserire un valore `desired`, si dovrebbe essere in grado di osservare le interazioni Device Shadow utilizzando il documento Shadow e il client

di test MQTT come descritto in precedenza. Tuttavia, se non riesci a visualizzare le interazioni, ecco alcune cose da controllare:

- Controlla il nome dell'oggetto e la sua ombra nella console AWS IoT

  Se i messaggi non vengono visualizzati nel documento Shadow, controlla il comando e assicurati che corrisponda al nome dell'oggetto nella console AWS IoT . Puoi anche verificare se hai una shadow classica scegliendo la tua risorsa dell'oggetto e quindi scegliendo Shadows. Questo tutorial si concentra principalmente sulle interazioni con la shadow classica.

  È inoltre possibile confermare che il dispositivo utilizzato è connesso a Internet. Nella console AWS IoT , scegli l'oggetto che hai creato in precedenza, quindi scegli Interact (Interagisci). Nella pagina dei dettagli dell'oggetto, dovresti visualizzare un messaggio con scritto: `This thing already appears to be connected.`

- Controlla gli argomenti riservati MQTT che hai sottoscritto

  Se i messaggi non vengono visualizzati nel client di test MQTT, verifica se gli argomenti sottoscritti sono formattati correttamente. Gli argomenti MQTT Device Shadow hanno il formato $aws/things/ ***thingname*** /shadow/ e possono averlo `update` o `delete` seguirlo a seconda delle azioni che si desidera eseguire sull'shadow. `get` Questo tutorial utilizza l'argomento $aws/things/ ***thingname*** /shadow/ #, quindi assicurati di averlo inserito correttamente quando ti iscrivi all'argomento nella sezione Filtro argomento del client di test.

  Mentre inserisci il nome dell'argomento, assicurati che sia lo stesso del nome dell'*thingname*oggetto che hai creato in precedenza. AWS IoT È inoltre possibile sottoscrivere altri argomenti MQTT per verificare se un aggiornamento è stato eseguito correttamente. Ad esempio, puoi iscriverti all'argomento ***thingname***$aws/things/ per ricevere un messaggio ogni volta che una richiesta shadow/update/rejected di aggiornamento fallisce, in modo da poter risolvere i problemi di connessione. Per ulteriori informazioni sugli argomenti riservati, consulta the section called "Argomenti copie shadow" e Argomenti MQTT di Device Shadow.

Fase 4: Esamina i risultati e i passaggi successivi

In questo tutorial, hai appreso come:

- Utilizzare l'app di esempio `shadow.py` per specificare gli stati desiderati e aggiornare lo stato corrente di shadow.

- Modificare il documento Shadow per osservare gli eventi delta e come l'app di esempio `shadow.py` risponde ad essi.
- Utilizzare il client di test MQTT per effettuare la sottoscrizione ad argomenti shadow e osservare gli aggiornamenti quando si esegue il programma di esempio.

Fasi successive

Puoi iscriverti ad altri argomenti riservati MQTT per osservare gli aggiornamenti dell'applicazione shadow. Ad esempio, se ti iscrivi solo all'argomento ***`thingname`***$aws/things/ shadow/update/ accepted, quando un aggiornamento viene eseguito correttamente, vedrai solo le informazioni sullo stato corrente.

È inoltre possibile sottoscrivere altri argomenti shadow per eseguire il debug dei problemi o per ottenere ulteriori informazioni sulle interazioni Device Shadow e anche eseguire il debug di eventuali problemi con le interazioni Device Shadow. Per ulteriori informazioni, consultare the section called “Argomenti copie shadow” e Argomenti MQTT di Device Shadow.

Puoi anche scegliere di estendere la tua applicazione utilizzando shadow denominati o utilizzando hardware aggiuntivo collegato al Raspberry Pi LEDs e osservare le modifiche al loro stato utilizzando i messaggi inviati dal terminale.

Per ulteriori informazioni sul servizio Device Shadow e sull'utilizzo del servizio in dispositivi, app e servizi, consulta AWS IoT Servizio Device Shadow, Utilizzo delle copie shadow nei dispositivi e Utilizzo delle copie shadow in app e servizi.

Tutorial: creazione di un autorizzatore personalizzato per AWS IoT Core

Questo tutorial illustra i passaggi per creare, convalidare e utilizzare l'autenticazione ad hoc utilizzando il AWS CLI. Facoltativamente, utilizzando questo tutorial, è possibile utilizzare Postman per inviare dati ad AWS IoT Core utilizzando l'API HTTP Publish.

In questo tutorial viene illustrato come creare una funzione Lambda di esempio che implementa la logica di autorizzazione e autenticazione e un'autorizzazione personalizzata utilizzando la chiamata create-authorizer con la firma dei token abilitata. L'autorizzatore viene quindi convalidato utilizzando e infine è possibile inviare dati utilizzando l'test-invoke-authorizerAPI HTTP Publish a AWS IoT Core un argomento MQTT di test. La richiesta di esempio specificherà l'autorizzatore da richiamare utilizzando l'`x-amz-customauthorizer-name`intestazione e passerà le intestazioni e nella richiesta. token-key-name `x-amz-customauthorizer-signature`

Cosa imparerai in questo tutorial:

- Come creare una funzione Lambda per essere un gestore di autorizzazioni ad hoc
- Come creare un autorizzatore personalizzato utilizzando l'opzione con firma con token abilitata AWS CLI
- Come testare la tua autorizzazione ad hoc utilizzando il comando test-invoke-authorizer
- Come pubblicare un argomento MQTT utilizzando lo strumento Postman e convalidare la richiesta con la tua autorizzazione ad hoc

Questo tutorial dura circa 60 minuti.

In questo tutorial, apprenderai a:

- Fase 1: Creazione di una funzione Lambda per la propria autorizzazione ad hoc
- Fase 2: Creazione di una coppia di chiavi pubblica e privata per le tue autorizzazioni ad hoc
- Passaggio 3: Crea una risorsa di autorizzazione personalizzata e la relativa autorizzazione
- Fase 4: Verifica l'autorizzatore chiamando test-invoke-authorizer
- Fase 5: Test della pubblicazione del messaggio MQTT tramite Postman
- Fase 6: Visualizzazione dei messaggi nel client di test MQTT
- Fase 7: Esamina i risultati e i passaggi successivi
- Fase 8: Pulizia

Prima di iniziare questo tutorial, assicurati di disporre di:

- Configurare Account AWS

  Avrai bisogno della tua Account AWS AWS IoT console per completare questo tutorial.

  L'account che utilizzi per questo tutorial funziona meglio quando include almeno queste policy gestite da AWS :

  - `IAMFullAccess`
  - `AWSIoTFullAccess`
  - `AWSLambda_FullAccess`

> **Important**
>
> Le policy IAM utilizzate in questo tutorial sono più permissive di quelle che si dovrebbero seguire in un'implementazione di produzione. In un ambiente di produzione, assicurati che le policy dell'account e delle risorse concedano solo le autorizzazioni necessarie.
> Quando si creano policy IAM per la produzione, determinare l'accesso necessario agli utenti e ai ruoli e quindi progettare le policy che consentono loro di eseguire solo tali attività.
> Per ulteriori informazioni, consulta la sezione Best practice per la sicurezza in IAM

- Hai installato il AWS CLI

  Per informazioni su come installare AWS CLI, consulta Installazione della AWS CLI. Questo tutorial richiede una AWS CLI versione `aws-cli/2.1.3 Python/3.7.4 Darwin/18.7.0 exe/x86_64` o successiva.

- Strumenti OpenSSL

  Gli esempi di questo tutorial utilizzano LibreSSL 2.6.5. È possibile utilizzare anche gli strumenti OpenSSL v1.1.1i per questo tutorial.

- Aver rivisto la panoramica di AWS Lambda

  Se non l'hai mai usato AWS Lambda prima, AWS Lambdaconsulta la sezione Guida introduttiva a Lambda per conoscerne i termini e i concetti.

- Esaminato come creare richieste con lo strumento Postman

  Per ulteriori informazioni, consulta la sezione Building requests (Richieste di compilazione).

- Autorizzazioni ad hoc rimosse dal tutorial precedente

  Account AWS È possibile configurare solo un numero limitato di autorizzatori personalizzati alla volta. Per informazioni su come rimuovere un'autorizzazione ad hoc, consulta the section called “Fase 8: Pulizia”.

Fase 1: Creazione di una funzione Lambda per la propria autorizzazione ad hoc

L'autenticazione personalizzata AWS IoT Core utilizza le risorse di autorizzazione create dall'utente per autenticare e autorizzare i client. La funzione che creerai in questa sezione autentica e autorizza i client durante la connessione e l'accesso alle risorse. AWS IoT Core AWS IoT

La funzione Lambda; svolge le operazioni seguenti:

- Se proviene una richiesta da test-invoke-authorizer, restituisce una policy IAM con un'azione `Deny`.
- Se una richiesta proviene da Postman tramite HTTP e il `actionToken` parametro ha un valore di`allow`, restituisce una policy IAM con un'azione. `Allow` In alternativa, restituisce una policy IAM con un'azione `Deny`.

Creazione di una funzione Lambda per la propria autorizzazione ad hoc

1. Nella console Lambda, apri Functions (Funzioni).
2. Scegli Crea funzione.
3. Conferma che l'opzione Author from scratch (Crea da zero) sia selezionata.
4. In Basic information (Informazioni di base):
   a. In Function name (Nome funzione) immettere **`custom-auth-function`**.
   b. In Runtime, conferma Node.js 18.x
5. Scegli Crea funzione.

   Lambda crea una funzione in Node.js e un ruolo di esecuzione che concede alla funzione l'autorizzazione di caricare i log. La funzione Lambda assume il ruolo di esecuzione quando si richiama la funzione e utilizza il ruolo di esecuzione per creare credenziali per l' AWS SDK e leggere i dati dalle fonti degli eventi.
6. Per visualizzare il codice e la configurazione della funzione nell'AWS Cloud9editor, scegliete custom-auth-functionnella finestra di progettazione, quindi scegliete index.js nel pannello di navigazione dell'editor.

   Per i linguaggi di script come Node.js, Lambda include una funzione di base che restituisce una risposta positiva. Puoi utilizzare l'editor di AWS Cloud9 per modificare le funzioni purché il codice sorgente superi i 3 MB.
7. Sostituisci il codice index.js nell'editor con il seguente codice:

```
// A simple Lambda function for an authorizer. It demonstrates
// How to parse a CLI and Http password to generate a response.

export const handler = async (event, context, callback) => {

    //Http parameter to initiate allow/deny request
```

```
const HTTP_PARAM_NAME='actionToken';
const ALLOW_ACTION = 'Allow';
const DENY_ACTION = 'Deny';

//Event data passed to Lambda function
var event_str = JSON.stringify(event);
console.log('Complete event :'+ event_str);

//Read protocolData from the event json passed to Lambda function
var protocolData = event.protocolData;
console.log('protocolData value---> ' + protocolData);

//Get the dynamic account ID from function's ARN to be used
// as full resource for IAM policy
var ACCOUNT_ID = context.invokedFunctionArn.split(":")[4];
console.log("ACCOUNT_ID---"+ACCOUNT_ID);

//Get the dynamic region from function's ARN to be used
// as full resource for IAM policy
var REGION = context.invokedFunctionArn.split(":")[3];
console.log("REGION---"+REGION);

//protocolData data will be undefined if testing is done via CLI.
// This will help to test the set up.
if (protocolData === undefined) {

    //If CLI testing, pass deny action as this is for testing purpose only.
    console.log('Using the test-invoke-authorizer cli for testing only');
    callback(null, generateAuthResponse(DENY_ACTION,ACCOUNT_ID,REGION));

} else{

    //Http Testing from Postman
    //Get the query string from the request
    var queryString = event.protocolData.http.queryString;
    console.log('queryString values -- ' + queryString);
    /*         global URLSearchParams       */
    const params = new URLSearchParams(queryString);
    var action = params.get(HTTP_PARAM_NAME);

    if(action!=null && action.toLowerCase() === 'allow'){

        callback(null, generateAuthResponse(ALLOW_ACTION,ACCOUNT_ID,REGION));
```

```
        }else{

            callback(null, generateAuthResponse(DENY_ACTION,ACCOUNT_ID,REGION));

        }

    }

};

// Helper function to generate the authorization IAM response.
var generateAuthResponse = function(effect,ACCOUNT_ID,REGION) {

    var full_resource = "arn:aws:iot:"+ REGION + ":" + ACCOUNT_ID + ":*";
    console.log("full_resource---"+full_resource);

    var authResponse = {};
    authResponse.isAuthenticated = true;
    authResponse.principalId = 'principalId';

    var policyDocument = {};
    policyDocument.Version = '2012-10-17';
    policyDocument.Statement = [];
    var statement = {};
    statement.Action = 'iot:*';
    statement.Effect = effect;
    statement.Resource = full_resource;
    policyDocument.Statement[0] = statement;
    authResponse.policyDocuments = [policyDocument];
    authResponse.disconnectAfterInSeconds = 3600;
    authResponse.refreshAfterInSeconds = 600;

    console.log('custom auth policy function called from http');
    console.log('authResponse --> ' + JSON.stringify(authResponse));
    console.log(authResponse.policyDocuments[0]);

    return authResponse;
}
```

8. Seleziona Implementa.
9. Dopo aver visualizzato Changes deployed (Modifiche implementate) in alto nell'editor:

    a. Scorri fino alla sezione Function overview (Panoramica delle funzioni) in alto nell'editor.

b. Copia la Function ARN (Funzione ARN) e salvala per utilizzarla più avanti in questo tutorial.

10. Prova la tua funzione .

    a. Seleziona la scheda Test.
    b. Utilizzando le impostazioni di test di default, fai clic su Invoke (Richiama).
    c. Se il test ha avuto esito positivo, in Execution results (Risultati dell'esecuzione), apri la visualizzazione Details (Dettagli). Dovresti vedere il documento di policy restituito dalla funzione.

       Se il test non è riuscito o non viene visualizzato un documento di policy, esaminare il codice per trovare e correggere gli errori.

Fase 2: Creazione di una coppia di chiavi pubblica e privata per le tue autorizzazioni ad hoc

La tua autorizzazioni ad hoc richiede una chiave pubblica e privata per autenticarla. I comandi di questa sezione utilizzano gli strumenti OpenSSL per creare questa coppia di chiavi.

Creazione di una coppia di chiavi pubblica e privata per le tue autorizzazioni ad hoc

1. Creazione del file della chiave privata.

   ```
   openssl genrsa -out private-key.pem 4096
   ```

2. Verificare il file della chiave privata appena creato.

   ```
   openssl rsa -check -in private-key.pem -noout
   ```

   Se il comando non mostra errori, il file della chiave privata è valido.

3. Creazione del file della chiave pubblica.

   ```
   openssl rsa -in private-key.pem -pubout -out public-key.pem
   ```

4. Verificare il file della chiave pubblica.

   ```
   openssl pkey -inform PEM -pubin -in public-key.pem -noout
   ```

   Se il comando non mostra errori, il file della chiave pubblica è valido.

Passaggio 3: Crea una risorsa di autorizzazione personalizzata e la relativa autorizzazione

L'autorizzatore AWS IoT personalizzato è la risorsa che unisce tutti gli elementi creati nei passaggi precedenti. In questa sezione, creerai una risorsa di autorizzazione ad hoc e la autorizzerai ad eseguire la funzione Lambda creata in precedenza. È possibile creare una risorsa di autorizzazione personalizzata utilizzando la AWS IoT console AWS CLI, l'o l' AWS API.

Per questo tutorial, è sufficiente creare una sola autorizzazione ad hoc. Questa sezione descrive come creare utilizzando la AWS IoT console e il AWS CLI, in modo da poter utilizzare il metodo più conveniente per te. Non c'è differenza tra le risorse delle autorizzazioni ad hoc create da entrambi i metodi.

Crea una risorsa di autorizzazione personalizzata

Scegli una di queste opzioni per creare la risorsa dell'autorizzazione ad hoc

- Crea un autorizzatore personalizzato utilizzando la console AWS IoT
- Creazione di un'autorizzazione ad hoc tramite AWS CLI

Per creare un'autorizzazione ad hoc (console)

1. Apri la pagina dell'autorizzazione personalizzata della AWS IoT console e scegli Crea autorizzazione.
2. In Crea autorizzazioni:
    a. In Nome dell'autorizzatore, immetti **`my-new-authorizer`**.
    b. In Stato autorizzazione, seleziona Attiva.
    c. In Authorizer function (Funzione autorizzazione), seleziona la funzione Lambda creata in precedenza.
    d. In Token validation - optional (Convalida token - facoltativa):
        i. Attiva Convalida token.
        ii. In Nome della chiave dei token, immetti **`tokenKeyName`**.
        iii. Scegliere Add key (Aggiungi chiave).
        iv. In Nome chiave, immetti **`FirstKey`**.

v. In Chiave pubblica, immetti il contenuto del file `public-key.pem`. Assicurati di includere le righe del file con `-----BEGIN PUBLIC KEY-----` e `-----END PUBLIC KEY-----` e non aggiungere o rimuovere alcun feed di riga, ritorni a capo o altri caratteri dal contenuto del file. La stringa inserita dovrebbe essere simile a quanto segue.

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBzOk4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQOe5tyzz1MsKdmtFGxMqQ3qjEXAMPLEOmqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU5OSAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csalS/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N5ON1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZO8OzcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7l7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFl2yOBmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXSOfUCAwEAAQ==
-----END PUBLIC KEY-----
```

3. Scegli Create Authorizer (Crea autorizzazioni).
4. Se la risorsa di autorizzazione ad hoc è stata creata, vedrai l'elenco delle autorizzazioni ad hoc e la tua nuova autorizzazione ad hoc dovrebbe apparire nell'elenco e puoi continuare alla sezione successiva per testarla.

   Se viene visualizzato un errore, rivedere l'errore e provare a creare nuovamente l'autorizzazione ad hoc e ricontrollare le voci. Attenzione: ciascuna risorsa dell'autorizzazione ad hoc deve avere un nome univoco.

Per creare un'autorizzazione ad hoc (AWS CLI)

1. Sostituisci i tuoi valori con `authorizer-function-arn` e `token-signing-public-keys` e quindi esegui il seguente comando:

```
aws iot create-authorizer \
--authorizer-name "my-new-authorizer" \
--token-key-name "tokenKeyName" \
--status ACTIVE \
--no-signing-disabled \
```

```
--authorizer-function-arn "arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function" \
--token-signing-public-keys FirstKey="-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBzOk4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQOe5tyzz1MsKdmtFGxMqQ3qjEXAMPLEOmqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU5OSAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csalS/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N5ON1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZO8OzcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7l7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFl2yOBmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXSOfUCAwEAAQ==
-----END PUBLIC KEY-----"
```

Dove:

- Il valore `authorizer-function-arn` è l'Amazon Resource Name (ARN) della funzione Lambda creata per l'autorizzazione ad hoc.
- Il valore `token-signing-public-keys` include il nome della chiave, **`FirstKey`** e i contenuti del file `public-key.pem`. Assicurati di includere le righe del file con `-----BEGIN PUBLIC KEY-----` e `-----END PUBLIC KEY-----` e non aggiungere o rimuovere alcun feed di riga, ritorni a capo o altri caratteri dal contenuto del file.

  Nota: fai attenzione a inserire la chiave pubblica poiché qualsiasi modifica al valore della chiave pubblica la rende inutilizzabile.

2. Creando l'autorizzazione ad hoc, il comando restituisce il nome e l'ARN della nuova risorsa, come qui di seguito.

```
{
    "authorizerName": "my-new-authorizer",
    "authorizerArn": "arn:aws:iot:Region:57EXAMPLE833:authorizer/my-new-authorizer"
}
```

Salva il valore `authorizerArn` da utilizzare nella fase successiva.

Ricorda che ciascuna risorsa dell'autorizzazione ad hoc deve avere un nome univoco.

Autorizza le risorse delle autorizzazioni ad hoc

In questa sezione, verrà descritto come concedere alla risorsa dell'autorizzazione ad hoc appena creata l'autorizzazione per eseguire la funzione Lambda. Per concedere l'autorizzazione, è possibile utilizzare il comando dell'interfaccia a riga di comando add-permission.

Concedi l'autorizzazione alla tua funzione Lambda utilizzando il AWS CLI

1. Una volta inseriti i valori, inserisci il comando seguente. Attenzione: il valore `statement-id` deve essere univoco. Sostituisci *`Id-1234`* con un altro valore se hai eseguito questo tutorial prima o in occasione di un errore `ResourceConflictException`.

```
aws lambda add-permission  \
--function-name "custom-auth-function" \
--principal "iot.amazonaws.com" \
--action "lambda:InvokeFunction" \
--statement-id "Id-1234" \
--source-arn authorizerArn
```

2. Se il comando viene completato correttamente, restituisce un'istruzione di autorizzazione come questa. È possibile passare alla sezione successiva per testare l'autorizzazione ad hoc.

```
{
    "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
\":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
\",\"Resource\":\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-
auth-function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function\"}}}"
}
```

   Se il comando non ha esito positivo, restituisce un errore come questo. Dovrai esaminare e correggere l'errore prima di continuare.

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
mission on resource: arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-
function
```

Fase 4: Verifica l'autorizzatore chiamando test-invoke-authorizer

Con tutte le risorse definite, in questa sezione, chiamerai test-invoke-authorizer dalla riga di comando per testare il passaggio di autorizzazione.

Si noti che quando si richiama l'autorizzazione dalla riga di comando, `protocolData` non è definito, quindi l'autore restituirà sempre un documento DENY. Tuttavia, questo test conferma che l'autorizzazione ad hoc e la funzione Lambda siano configurate correttamente, anche se non esegue il test completo della funzione Lambda.

Per testare l'autorizzatore personalizzato e la relativa funzione Lambda utilizzando il AWS CLI

1. Nella directory che contiene il file `private-key.pem` creato in un passaggio precedente, eseguire il comando seguente.

   ```
   echo -n "tokenKeyValue" | openssl dgst -sha256 -sign private-key.pem | openssl
    base64 -A
   ```

   Questo comando crea una stringa di firma da utilizzare nel passaggio successivo. La stringa della firma ha un aspetto simile a questo:

   ```
   dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V
   +MMWu09YSA86+64Y3Gt4tOykpZqn9mn
   VB1wyxp+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
   +CPY0zrWt1jr9BikgGPDxWkjaeeh
   bQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsHNig1JePgnuOBvMGCEFE09jGjj
   szEHfgAUAQIWXiVGQj16BU1xKpTGSiTAwheLKUjITOEXAMPLECK3aHKYKY
   +d1vTvdthKtYHBq8MjhzJ0kggbt29V
   QJCb8RilN/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
   +tsDuX
   f3LzCwQQF/YSUy02u5XkWn
   +sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o+K
   EWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
   +flFeloZlAWQFH
   xRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
   ```

   Copia la stringa della firma da utilizzare nel passaggio successivo. Fai attenzione a non includere caratteri aggiuntivi o a non lasciarli fuori.

2. In questo comando, sostituire il valore `token-signature` con la stringa di firma del passaggio precedente ed esegui questo comando per testare l'autorizzazione.

```
aws iot test-invoke-authorizer \
--authorizer-name my-new-authorizer \
--token tokenKeyValue \
--token-signature dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr
+rbCvmu9Jl4KHAA9DG+V+MMWu09YSA86+64Y3Gt4tOykpZqn9mnVB1wyxp
+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsH
+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RilN/
P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
+tsDuXf3LzCwQQF/YSUy02u5XkWn
+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o
+KEWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+flFeloZlAWQFHxRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
```

Se il comando ha esito positivo, restituisce le informazioni generate dalla funzione di autorizzazione personalizzata, come questo esempio.

```
{
    "isAuthenticated": true,
    "principalId": "principalId",
    "policyDocuments": [
        "{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Action\":\"iot:*\",
\"Effect\":\"Deny\",\"Resource\":\"arn:aws:iot:Region:57EXAMPLE833:*\"}]}"
    ],
    "refreshAfterInSeconds": 600,
    "disconnectAfterInSeconds": 3600
}
```

Se il comando restituisce un errore, esamina l'errore e ricontrolla i comandi utilizzati in questa sezione.

Fase 5: Test della pubblicazione del messaggio MQTT tramite Postman

1. Per ottenere l'endpoint dati del dispositivo dalla riga di comando, chiama describe-endpoint come mostrato qui

```
aws iot describe-endpoint --output text --endpoint-type iot:Data-ATS
```

Salva questo indirizzo per utilizzarlo come *device_data_endpoint_address* in un passaggio successivo.

2. Apri una nuova finestra Postman e crea una nuova richiesta HTTP POST.

   a. Dal computer, apri l'app Postman.

   b. In Postman, dal menu File, seleziona New... (Nuovo...).

   c. Nella finestra di dialogo New (Nuovo) seleziona Request (Richiesta).

   d. In Salva richiesta,

      i. In Request name (Nome della richiesta) inserisci **Custom authorizer test request**.

      ii. In Select a collection or folder to save to: (Seleziona una raccolta o una cartella in cui salvare:) scegli o crea una raccolta in cui salvare questa richiesta.

      iii. Scegli Salva in ***collection_name***.

3. Crea la richiesta POST per testare la tua autorizzazione ad hoc.

   a. Nel selettore del metodo di richiesta accanto al campo URL, seleziona POST.

   b. Nel campo URL, crea l'URL per la tua richiesta utilizzando il seguente URL con il comando *device_data_endpoint_address* from the describe-endpoint del passaggio precedente.

      ```
      https://device_data_endpoint_address:443/topics/test/cust-auth/topic?
      qos=0&actionToken=allow
      ```

      Attenzione: questo URL include il parametro query `actionToken=allow` che dirà alla funzione Lambda di restituire un documento policy che consenta l'accesso a AWS IoT. Dopo aver inserito l'URL, i parametri della query vengono visualizzati anche nella scheda Params (Parametri) di Postman.

   c. Nella scheda Auth (Autenticazione), nel campo Type (Tipo), seleziona No Auth (Nessuna autenticazione).

   d. Nella scheda Intestazioni:

      i. Se c'è un tasto host selezionato, deselezionalo.

      ii. Nella parte inferiore dell'elenco delle intestazioni aggiungete queste nuove intestazioni e confermate che siano state selezionate. Sostituisci il **Host** valore con il tuo *device_data_endpoint_address* e il **x-amz-customauthorizer-signature**

valore con la stringa della firma che hai usato con il test-invoke-authorize comando nella sezione precedente.

Chiave	Valore
x-amz-customauthorizer-name	**my-new-authorizer**
Host	*device_data_endpoint_address*
tokenKeyName	**tokenKeyValue**

Chiave	Valore
x-amz-customauthorizer-signature	*dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V+MMWu09YSA86+64Y3Gt4tOykpZqn9mnVB1wyxp+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsHNig1JePgnuOBvMGCEFE09jGjjszEHfgAUAQIWXiVGQj16BU1xKpTGSiTAwheLKUjITOEXAMPLECK3aHKYKY+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RilN/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca+tsDuXf3LzCwQQF/YSUy02u5XkWn+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o+KEWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h+flFeloZlAWQFHxRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=*

e. Nella scheda Body (Corpo):

i. Nella casella di opzione formato dati, scegli Raw (Non elaborati).

ii. Nell'elenco dei tipi di dati, scegli JavaScript.

iii. Nel campo di testo, inserisci questo payload del messaggio JSON per il messaggio di prova:

```
{
```

```
    "data_mode": "test",
    "vibration": 200,
    "temperature": 40
}
```

4. Scegli Send (Invia) per inviare la richiesta.

   Se la richiesta è stata completata correttamente, verrà visualizzato:

   ```
   {
       "message": "OK",
       "traceId": "ff35c33f-409a-ea90-b06f-fbEXAMPLE25c"
   }
   ```

   La risposta corretta indica che l'autorizzazione personalizzata ha consentito la connessione a AWS IoT e che il messaggio di prova è stato recapitato al broker in AWS IoT Core.

   Se restituisce un errore, esamina il messaggio di errore, la*device_data_endpoint_address*, la stringa della firma e gli altri valori di intestazione.

Conserva questa richiesta in Postman per utilizzarla nella sezione successiva.

Fase 6: Visualizzazione dei messaggi nel client di test MQTT

Nel passaggio precedente, hai inviato messaggi simulati al dispositivo AWS IoT utilizzando Postman. La risposta riuscita ha indicato che l'autorizzazione ad hoc ha consentito la connessione a AWS IoT e che il messaggio di prova è stato consegnato al broker AWS IoT Core. In questa sezione, utilizzerai il client di test MQTT nella AWS IoT console per visualizzare il contenuto dei messaggi di quel messaggio come potrebbero fare altri dispositivi e servizi.

Per visualizzare i messaggi di prova autorizzati dalla tua autorizzazione ad hoc

1. Nella AWS IoT console, aprite il client di test MQTT.
2. Nella scheda Subscribe to a topic (Sottoscrivi un argomento), in Topic filter (Filtro argomenti), inserisci **`test/cust-auth/topic`**, ovvero l'argomento del messaggio utilizzato nell'esempio Postman della sezione precedente.
3. Scegli Abbonati.

   Tieni questa finestra visibile per la fase successiva.

4. In Postman, nella richiesta creata per la sezione precedente, seleziona Send (Invia).

   Controlla la risposta per assicurarti che sia stata elaborata correttamente. In caso contrario, risolvi l'errore come descritto nella sezione precedente.

5. Nel Client MQTT di test, dovresti vedere una nuova voce che mostra l'argomento del messaggio e, se espanso, il payload del messaggio dalla richiesta inviata da Postman.

   Se non si visualizzano i messaggi nel client MQTT di test, ecco alcune cose da controllare:

   - Assicurati che la tua richiesta Postman sia stata restituita correttamente. Se AWS IoT rifiuta la connessione e restituisce un errore, il messaggio contenuto nella richiesta non viene passato al broker di messaggi.
   - Assicurati che la Account AWS e Regione AWS usata per aprire la AWS IoT console sia la stessa che usi nell'URL di Postman.
   - Assicurati di utilizzare l'endpoint appropriato per l'autorizzatore personalizzato. L'endpoint IoT predefinito potrebbe non supportare l'utilizzo di autorizzazioni personalizzate con funzioni Lambda. Puoi invece utilizzare le configurazioni di dominio per definire un nuovo endpoint e quindi specificare quell'endpoint per l'autorizzatore personalizzato.
   - Assicurati di aver inserito correttamente l'argomento nel client MQTT di test. Il filtro non fa distinzione tra maiuscole e minuscole. In caso di dubbio, puoi anche iscriverti all'**#**argomento, che sottoscrive tutti i messaggi MQTT che passano attraverso il broker di messaggi Account AWS e Regione AWS utilizzati per aprire la console. AWS IoT

Fase 7: Esamina i risultati e i passaggi successivi

In questo tutorial:

- Hai creato una funzione Lambda come gestore di autorizzazioni ad hoc
- Hai creato un'autorizzazione ad hoc con la firma dei token abilitata
- Hai testato la tua autorizzazione ad hoc utilizzando il comando test-invoke-authorizer
- Hai pubblicato un argomento MQTT utilizzando Postman e convalidato la richiesta con la tua autorizzazione ad hoc
- Hai usato il client MQTT di test per visualizzare i messaggi inviati dal test Postman

Fasi successive

Dopo aver inviato alcuni messaggi da Postman per verificare che l'autorizzazione ad hoc funzioni, prova a sperimentare vedendo come i diversi aspetti di questo tutorial influiscono sui risultati. Ecco alcuni esempi per cominciare.

- Modificare la stringa di firma in modo che non sia più valida, in modo tale da poter vedere come vengono gestiti i tentativi di connessione non autorizzati. Dovresti ricevere una risposta di errore, come questa, e il messaggio non dovrebbe apparire nel client MQTT di test.

```
{
    "message": "Forbidden",
    "traceId": "15969756-a4a4-917c-b47a-5433e25b1356"
}
```

- Per ulteriori informazioni su come individuare gli errori che potrebbero verificarsi durante lo sviluppo e l'utilizzo AWS IoT delle regole, consulta. Monitoraggio AWS IoT

Fase 8: Pulizia

Se desideri ripetere questo tutorial, potresti dover rimuovere alcune delle tue autorizzazioni ad hoc. Account AWS Puoi avere solo un numero limitato di autorizzatori personalizzati configurati contemporaneamente e puoi ottenerne uno `LimitExceededException` quando provi ad aggiungerne uno nuovo senza rimuovere un autorizzatore personalizzato esistente.

Per rimuovere un'autorizzazione ad hoc (console)

1. Apri la pagina Autorizzatore personalizzato della AWS IoT console e, nell'elenco degli autorizzatori personalizzati, trova l'autorizzatore personalizzato da rimuovere.
2. Apri la pagina dei dettagli delle autorizzazioni ad hoc e, dal menu Actions (Azioni), seleziona Edit (Modifica).
3. Deseleziona l'opzione Activate authorizer (Attiva autorizzazione) e quindi seleziona Update (Aggiorna).

   Non puoi eliminare un'autorizzazione ad hoc mentre è attiva.
4. Apri la pagina dei dettagli delle autorizzazioni ad hoc, apri il menu Actions (Azioni), seleziona Delete (Elimina).

Per rimuovere un'autorizzazione ad hoc (AWS CLI)

1. Elenca le autorizzazioni ad hoc installate e trova il nome dell'autorizzazione ad hoc da eliminare.

```
aws iot list-authorizers
```

2. Imposta l'autorizzazione ad hoc in stato `inactive` eseguendo questo comando dopo la sostituzione di *`Custom_Auth_Name`* con il `authorizerName` dell'autorizzazione ad hoc da eliminare.

```
aws iot update-authorizer --status INACTIVE --authorizer-name Custom_Auth_Name
```

3. Elimina l'autorizzazione ad hoc eseguendo questo comando dopo la sostituzione di *`Custom_Auth_Name`* con il `authorizerName` dell'autorizzazione ad hoc da eliminare.

```
aws iot delete-authorizer --authorizer-name Custom_Auth_Name
```

Tutorial: monitoraggio dell'umidità del suolo con AWS IoT Raspberry Pi

Questo tutorial mostra come utilizzare un Raspberry Pi, un sensore di umidità e come AWS IoT monitorare il livello di umidità del suolo per una pianta da appartamento o un giardino. Il Raspberry Pi esegue un codice che legge il livello di umidità e la temperatura dal sensore e quindi invia i dati a. AWS IoT Crei una regola AWS IoT che invia un'e-mail a un indirizzo abbonato a un argomento di Amazon SNS quando il livello di umidità scende al di sotto di una soglia.

> **Note**
>
> Questo tutorial potrebbe non essere aggiornato. Alcuni riferimenti potrebbero essere stati sostituiti dalla pubblicazione di questo argomento.

Indice

- Prerequisiti
- Configurazione AWS IoT
  - Fase 1: Creare la AWS IoT politica
  - Fase 2: Creare l' AWS IoT oggetto, il certificato e la chiave privata
  - Fase 3: Creazione e sottoscrizione a un argomento Amazon SNS

  - Passaggio 4: Crea una AWS IoT regola per inviare un'email
- Configurazione del Raspberry Pi e del sensore di umidità

Prerequisiti

Per completare questo tutorial, è necessario quanto segue:

- Un. Account AWS
- Un utente IAM con autorizzazioni da amministratore.
- Un computer di sviluppo che esegue Windows, macOS, Linux o Unix per accedere alla console AWS IoT.
- Un Raspberry Pi 3B o 4B con l'ultimo sistema operativo Raspberry Pi. Per le istruzioni di installazione, consulta Installazione di un sistema operativo sul sito Web di Raspberry Pi.
- Un monitor, una tastiera, un mouse e una rete Wi-Fi o una connessione Ethernet per il Raspberry Pi.
- Un sensore di umidità compatibile con Raspberry Pi. Il sensore utilizzato in questo tutorial è un sensore di umidità capacitivo Adafruit STEMMA I2C con connettore femmina su un'estremità e JST a 4 pin all'estremità opposta.

Configurazione AWS IoT

Per completare questo tutorial, è necessario creare le seguenti risorse. Per connettere un dispositivo AWS IoT, crei un oggetto IoT, un certificato del dispositivo e una AWS IoT policy.

- Qualsiasi AWS IoT cosa.

  Un oggetto rappresenta un dispositivo fisico (in questo caso, il Raspberry Pi) e contiene metadati statici relativi al dispositivo.
- Un certificato del dispositivo.

  Tutti i dispositivi devono disporre di un certificato del dispositivo per connettersi a AWS IoT ed eseguire l'autenticazione.
- Una AWS IoT politica.

  A ogni certificato del dispositivo sono associate una o più AWS IoT politiche. Queste politiche determinano a quali AWS IoT risorse può accedere il dispositivo.

- Un certificato CA AWS IoT principale.

  I dispositivi e gli altri client utilizzano un certificato CA AWS IoT principale per autenticare il AWS IoT server con cui comunicano. Per ulteriori informazioni, consulta Autenticazione del server.
- Una AWS IoT regola.

  Una regola contiene una query e una o più operazioni di regola. La query estrae i dati dai messaggi del dispositivo per determinare se i dati del messaggio devono essere elaborati. L'operazione della regola specifica cosa fare se i dati corrispondono alla query.
- Un argomento Amazon SNS e una sottoscrizione di un argomento.

  La regola ascolta i dati di umidità dal Raspberry Pi. Se il valore è inferiore a una soglia, invia un messaggio all'argomento Amazon SNS. Amazon SNS invia tale messaggio a tutti gli indirizzi e-mail che hanno effettuato la sottoscrizione all'argomento.

Fase 1: Creare la AWS IoT politica

Crea una AWS IoT policy che consenta al tuo Raspberry Pi di connettersi e inviare messaggi a AWS IoT.

1. Se nella console AWS IoT è presente un pulsante Get started (Inizia), selezionarlo. In caso contrario, nel riquadro di navigazione del servizio espandere Security (Sicurezza), quindi selezionare Policies (Policy).
2. Se viene visualizzata la finestra di dialogoYou don't have any policies yet (Non hai ancora policy), selezionare Create a policy (Crea una policy). In caso contrario, scegliere Create (Crea).
3. Inserisci un nome per la AWS IoT politica (ad esempio,**`MoistureSensorPolicy`**).
4. Nella sezione Add statements (Aggiungi istruzioni), sostituire la policy esistente con il seguente JSON. Sostituisci «*`region`*e» *`account`* con il tuo Account AWS numero Regione AWS e.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:Connect",
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/RaspberryPi"
        },
```

```
        {
            "Effect": "Allow",
            "Action": "iot:Publish",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/update",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/delete",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/get"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/delete/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/delete/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/delete/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/delete/rejected"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:thing/RaspberryPi"
        }
    ]
}
```

5. Scegli Create (Crea).

Fase 2: Creare l' AWS IoT oggetto, il certificato e la chiave privata

Crea un elemento nel AWS IoT registro per rappresentare il tuo Raspberry Pi.

1. Nel riquadro di navigazione della console AWS IoT scegliere Manage (Gestisci), quindi scegliere Things (Oggetti).
2. Se è visibile la finestra di dialogo You don't have any things yet (Non hai ancora oggetti), selezionare Register a thing (Registra un oggetto). In caso contrario, scegliere Create (Crea).
3. Nella pagina Creazione di AWS IoT oggetti, scegli Crea una singola cosa.
4. Nella pagina Add your device to the device registry (Aggiungi il tuo dispositivo al registro dei dispositivi), immettere un nome per l'oggetto IoT, ad esempio **`RaspberryPi`**, quindi scegliere Next (Avanti). Non puoi modificare il nome di un oggetto dopo averlo creato. Per cambiare il nome di un oggetto, devi creare un nuovo oggetto, dargli il nuovo nome e quindi eliminare il vecchio oggetto.
5. Nella pagina Add a certificate for your thing (Aggiungi un certificato per l'oggetto), scegli Create certificate (Crea certificato).
6. Scegliere i collegamenti Download (Scarica) per scaricare il certificato, la chiave privata e il certificato CA root.

   > **Important**
   >
   > Questa è l'unico momento in cui è possibile scaricare il certificato e la chiave privata.

7. Scegli Activate (Attiva) per attivare il certificato. Il certificato deve essere attivo affinché un dispositivo possa connettersi a AWS IoT.

8. Scegliere Attach a policy (Collega policy).
9. Per Aggiungi una politica per il tuo oggetto, scegli MoistureSensorPolicy, quindi scegli Registra oggetto.

Fase 3: Creazione e sottoscrizione a un argomento Amazon SNS

Creazione e sottoscrizione di un argomento Amazon SNS.

1. Dalla console AWS SNS, nel riquadro di navigazione, scegli Topics (Argomenti) e quindi scegli Create topic (Crea argomento).
2. Scegli Tipo come Standard e inserisci un nome per l'argomento (ad esempio,**`MoistureSensorTopic`**).
3. Immettere un nome visualizzato per l'argomento, ad esempio **`Moisture Sensor Topic`**. Questo è il nome visualizzato per l'argomento nella console Amazon SNS.
4. Scegli Create topic (Crea argomento).
5. Nella pagina dei dettagli dell'argomento Amazon SNS, scegli Create Subscription (Crea sottoscrizione).
6. Per Protocollo, scegli E-mail.
7. Per Endpoint, immettere il proprio indirizzo e-mail.
8. Scegli Create Subscription (Crea sottoscrizione).
9. Aprire il client e-mail e cercare un messaggio con l'oggetto **`MoistureSensorTopic`**. Aprire l'e-mail e selezionare il collegamento Confirm subscription (Conferma sottoscrizione).

> **Important**
>
> Non riceverai avvisi e-mail da questo argomento Amazon SNS finché non confermi la sottoscrizione.

Dovresti ricevere un messaggio di posta elettronica con il testo che hai digitato.

Passaggio 4: Crea una AWS IoT regola per inviare un'email

Una AWS IoT regola definisce una query e una o più azioni da eseguire quando si riceve un messaggio da un dispositivo. Il motore AWS IoT delle regole ascolta i messaggi inviati dai dispositivi

e utilizza i dati contenuti nei messaggi per determinare se è necessario intraprendere un'azione. Per ulteriori informazioni, consulta Regole per AWS IoT.

In questo tutorial, il Raspberry Pi pubblica messaggi su `aws/things/RaspberryPi/shadow/update`. Si tratta di un argomento MQTT interno utilizzato dai dispositivi e dal servizio Thing Shadow. Il Raspberry Pi pubblica messaggi nel formato seguente:

```
{
    "reported": {
        "moisture" : moisture-reading,
        "temp" : temperature-reading
    }
}
```

È necessario creare una query che estrae i dati relativi a umidità e temperatura dal messaggio in entrata. È necessario inoltre creare un'azione Amazon SNS che acquisisce i dati e li invia agli iscritti dell'argomento se la lettura dell'umidità è inferiore a un valore di soglia.

Creazione di una regola Amazon SNS.

1. Nella AWS IoT console, scegli Routing dei messaggi, quindi scegli Regole. Se viene visualizzata la finestra di dialogoYou don't have any rules yet (Non hai ancora regole), selezionare Create a rule (Crea una regola). Altrimenti, scegli Crea regola.
2. Nella pagina delle proprietà della regola, inserisci un nome per la regola, ad esempio**`MoistureSensorRule`**, e fornisci una breve descrizione della regola, ad esempio**`Sends an alert when soil moisture level readings are too low`**.
3. Scegliete Avanti e configurate l'istruzione SQL. Scegli la versione SQL come 23/03/2016 e inserisci la seguente AWS IoT istruzione di query SQL:

   ```
   SELECT * FROM '$aws/things/RaspberryPi/shadow/update/accepted' WHERE
    state.reported.moisture < 400
   ```

   Questa istruzione attiva l'operazione della regola quando la lettura `moisture` è inferiore a `400`.

> **Note**
>
> Potrebbe essere necessario utilizzare un valore diverso. Quando il codice è in esecuzione sul Raspberry Pi, è possibile visualizzare i valori ottenuti dal sensore toccando il sensore, immergendolo in acqua o posizionandolo in una fioriera.

4. Scegli Avanti e allega le azioni delle regole. Per l'azione 1, scegli Simple Notification Service. La descrizione di questa azione della regola è Invia un messaggio come notifica push SNS.
5. Per l'argomento SNS, scegli l'argomento in Fase 3: Creazione e sottoscrizione a un argomento Amazon SNS cui hai creato e lascia il formato del messaggio come RAW. MoistureSensorTopic Per Ruolo IAM, scegli Crea un nuovo ruolo. Inserisci un nome per il ruolo, ad esempio**`LowMoistureTopicRole`**, e quindi scegli Crea ruolo.
6. Scegli Avanti per rivedere, quindi scegli Crea per creare la regola.

Configurazione del Raspberry Pi e del sensore di umidità

Inserire la scheda micro SD nel Raspberry Pi, connettere il monitor, la tastiera, il mouse e, se non si utilizza la rete Wi-Fi, il cavo Ethernet. Non collegare il cavo di alimentazione per il momento.

Collegare il cavo jumper JST al sensore di umidità. L'estremità opposta del jumper presenta quattro fili:

- Verde: I2C SCL
- Bianco: I2C SDA
- Rosso: alimentazione (3,5 V)
- Nero: terra

Tieni il Raspberry Pi con il connettore Ethernet a destra. In questo orientamento, ci sono due file di pin GPIO nella parte superiore. Collega i fili dal sensore di umidità alla fila inferiore di pin nel seguente ordine. Partendo dal pin più a sinistra, collega il rosso (alimentazione), bianco (SDA) e verde (SCL). Salta un pin, quindi collega il file nero (terra). Per ulteriori informazioni, consulta la pagina relativa al cablaggio per i computer Python.

Collega il cavo di alimentazione al Raspberry Pi e collega l'estremità opposta a una presa di corrente per accenderlo.

Configurare il Raspberry Pi

1. In Welcome to Raspberry Pi (Benvenuto in Raspberry Pi), scegliere Next (Avanti).
2. Scegliere il paese, la lingua, il fuso orario e il layout della tastiera. Scegli Next (Successivo).
3. Immettere una password per il Raspberry Pi, quindi scegliere Next (Avanti).
4. Scegliere la rete Wi-Fi, quindi scegliere Next (Avanti). Se non si utilizza una rete Wi-Fi, scegliere Skip (Ignora).
5. Scegliere Next (Avanti) per verificare la presenza di aggiornamenti software. Al termine degli aggiornamenti, scegliere Restart (Riavvia) per riavviare il Raspberry Pi.

Dopo l'avvio del Raspberry Pi, abilitare l'interfaccia I2C.

1. Nell'angolo in alto a sinistra del desktop Raspbian, fare clic sull'icona Raspberry, scegliere Preferences (Preferenze), quindi scegliere Raspberry Pi Configuration (Configurazione Raspberry Pi).
2. Nella scheda Interfaces (Interfacce), per I2C, scegliere Enable (Abilita).
3. Scegli OK.

Le librerie per il sensore di umidità Adafruit STEMMA sono state scritte per. CircuitPython Per eseguirle su un Raspberry Pi, è necessario installare la versione più recente di Python 3.

1. Eseguire i comandi seguenti da un prompt dei comandi per aggiornare il software Raspberry Pi:

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get upgrade
   ```
2. Eseguire il comando seguente per aggiornare l'installazione di Python 3:

   ```
   sudo pip3 install --upgrade setuptools
   ```
3. Eseguire il comando seguente per installare le librerie GPIO di Raspberry Pi:

   ```
   pip3 install RPI.GPIO
   ```
4. Eseguire il comando seguente per installare le librerie Adafruit Blinka:

   ```
   pip3 install adafruit-blinka
   ```

   Per ulteriori informazioni, consulta Installazione delle CircuitPython librerie su Raspberry Pi.

5. Eseguire il comando seguente per installare le librerie Adafruit Seesaw:

   ```
   sudo pip3 install adafruit-circuitpython-seesaw
   ```

6. Esegui il seguente comando per installare AWS IoT Device SDK per Python:

   ```
   pip3 install AWSIoTPythonSDK
   ```

Il Raspberry Pi ora dispone di tutte le librerie necessarie. Crea un file denominato **`moistureSensor.py`** e copia il seguente codice Python nel file:

```
from adafruit_seesaw.seesaw import Seesaw
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient
from board import SCL, SDA

import logging
import time
import json
import argparse
import busio

# Shadow JSON schema:
#
# {
#   "state": {
#       "desired":{
#           "moisture":<INT VALUE>,
#           "temp":<INT VALUE>
#       }
#   }
# }

# Function called when a shadow is updated
def customShadowCallback_Update(payload, responseStatus, token):

    # Display status and data from update request
    if responseStatus == "timeout":
        print("Update request " + token + " time out!")

    if responseStatus == "accepted":
        payloadDict = json.loads(payload)
        print("~~~~~~~~~~~~~~~~~~~~~~~")
```

```
        print("Update request with token: " + token + " accepted!")
        print("moisture: " + str(payloadDict["state"]["reported"]["moisture"]))
        print("temperature: " + str(payloadDict["state"]["reported"]["temp"]))
        print("~~~~~~~~~~~~~~~~~~~~~~~\n\n")

    if responseStatus == "rejected":
        print("Update request " + token + " rejected!")

# Function called when a shadow is deleted
def customShadowCallback_Delete(payload, responseStatus, token):

     # Display status and data from delete request
    if responseStatus == "timeout":
        print("Delete request " + token + " time out!")

    if responseStatus == "accepted":
        print("~~~~~~~~~~~~~~~~~~~~~~~")
        print("Delete request with token: " + token + " accepted!")
        print("~~~~~~~~~~~~~~~~~~~~~~~\n\n")

    if responseStatus == "rejected":
        print("Delete request " + token + " rejected!")


# Read in command-line parameters
def parseArgs():

    parser = argparse.ArgumentParser()
    parser.add_argument("-e", "--endpoint", action="store", required=True, dest="host",
 help="Your device data endpoint")
    parser.add_argument("-r", "--rootCA", action="store", required=True,
 dest="rootCAPath", help="Root CA file path")
    parser.add_argument("-c", "--cert", action="store", dest="certificatePath",
 help="Certificate file path")
    parser.add_argument("-k", "--key", action="store", dest="privateKeyPath",
 help="Private key file path")
    parser.add_argument("-p", "--port", action="store", dest="port", type=int,
 help="Port number override")
    parser.add_argument("-n", "--thingName", action="store", dest="thingName",
 default="Bot", help="Targeted thing name")
    parser.add_argument("-id", "--clientId", action="store", dest="clientId",
 default="basicShadowUpdater", help="Targeted client id")

    args = parser.parse_args()
```

```
    return args


# Configure logging
# AWSIoTMQTTShadowClient writes data to the log
def configureLogging():

    logger = logging.getLogger("AWSIoTPythonSDK.core")
    logger.setLevel(logging.DEBUG)
    streamHandler = logging.StreamHandler()
    formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -
 %(message)s')
    streamHandler.setFormatter(formatter)
    logger.addHandler(streamHandler)


# Parse command line arguments
args = parseArgs()

if not args.certificatePath or not args.privateKeyPath:
    parser.error("Missing credentials for authentication.")
    exit(2)

# If no --port argument is passed, default to 8883
if not args.port:
    args.port = 8883


# Init AWSIoTMQTTShadowClient
myAWSIoTMQTTShadowClient = None
myAWSIoTMQTTShadowClient = AWSIoTMQTTShadowClient(args.clientId)
myAWSIoTMQTTShadowClient.configureEndpoint(args.host, args.port)
myAWSIoTMQTTShadowClient.configureCredentials(args.rootCAPath, args.privateKeyPath,
 args.certificatePath)

# AWSIoTMQTTShadowClient connection configuration
myAWSIoTMQTTShadowClient.configureAutoReconnectBackoffTime(1, 32, 20)
myAWSIoTMQTTShadowClient.configureConnectDisconnectTimeout(10) # 10 sec
myAWSIoTMQTTShadowClient.configureMQTTOperationTimeout(5) # 5 sec

# Initialize Raspberry Pi's I2C interface
i2c_bus = busio.I2C(SCL, SDA)

# Intialize SeeSaw, Adafruit's Circuit Python library
```

```
ss = Seesaw(i2c_bus, addr=0x36)

# Connect to AWS IoT
myAWSIoTMQTTShadowClient.connect()

# Create a device shadow handler, use this to update and delete shadow document
deviceShadowHandler =
 myAWSIoTMQTTShadowClient.createShadowHandlerWithName(args.thingName, True)

# Delete current shadow JSON doc
deviceShadowHandler.shadowDelete(customShadowCallback_Delete, 5)

# Read data from moisture sensor and update shadow
while True:

    # read moisture level through capacitive touch pad
    moistureLevel = ss.moisture_read()

    # read temperature from the temperature sensor
    temp = ss.get_temp()

    # Display moisture and temp readings
    print("Moisture Level: {}".format(moistureLevel))
    print("Temperature: {}".format(temp))

    # Create message payload
    payload = {"state":{"reported":{"moisture":str(moistureLevel),"temp":str(temp)}}}

    # Update shadow
    deviceShadowHandler.shadowUpdate(json.dumps(payload), customShadowCallback_Update,
 5)
    time.sleep(1)
```

Salva il file nella posizione che preferisci. Esegui `moistureSensor.py` dalla riga di comando con i parametri seguenti:

endpoint

Il tuo endpoint personalizzato AWS IoT . Per ulteriori informazioni, consulta API REST del servizio Device Shadow.

rootCA

Il percorso completo del certificato CA AWS IoT root.

cert

Il percorso completo del certificato AWS IoT del dispositivo.

Chiave

Il percorso completo della chiave privata AWS IoT del certificato del dispositivo.

thingName

Nome dell'oggetto, in questo caso `RaspberryPi`.

clientId

ID client MQTT. Utilizza `RaspberryPi`.

La riga di comando dovrebbe essere simile alla seguente:

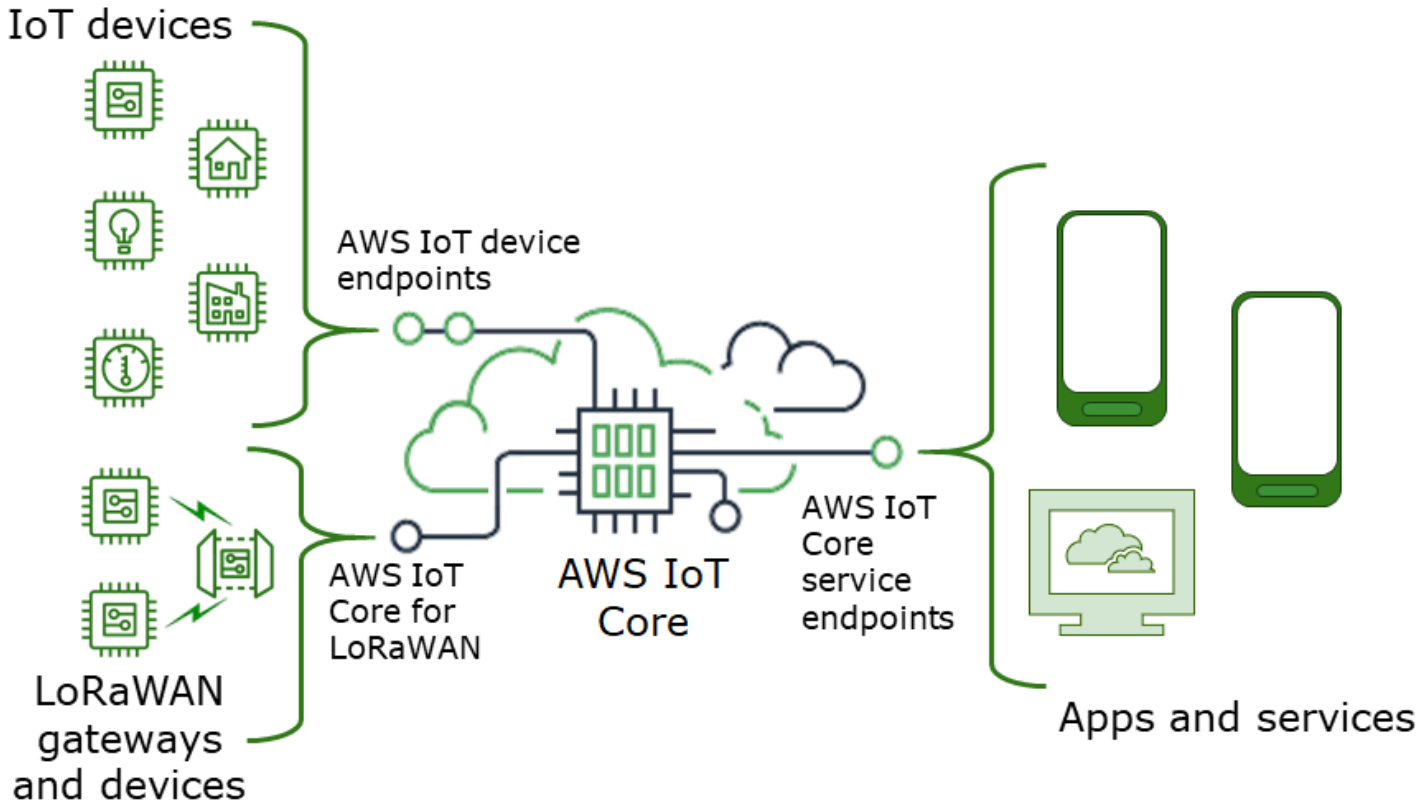
```
python3 moistureSensor.py --endpoint your-endpoint --rootCA ~/certs/
AmazonRootCA1.pem --cert ~/certs/raspberrypi-certificate.pem.crt --key
~/certs/raspberrypi-private.pem.key --thingName RaspberryPi --clientId
RaspberryPi
```

Prova a toccare il sensore, a inserirlo in una fioriera o a immergerlo in un bicchiere d'acqua per vedere come risponde a vari livelli di umidità. Se necessario, puoi modificare il valore di soglia in `MoistureSensorRule`. Quando la lettura del sensore di umidità scende al di sotto del valore specificato nell'istruzione di query SQL della regola, AWS IoT pubblica un messaggio sull'argomento Amazon SNS. Riceverai un messaggio e-mail contenente i dati relativi all'umidità e alla temperatura.

Dopo aver verificato la ricezione dei messaggi e-mail da Amazon SNS, premi CTRL+C per arrestare il programma Python. È improbabile che il programma Python invii un numero di messaggi tale da comportare un addebito, ma è una best practice interrompere il programma al termine delle operazioni.

Connect a AWS IoT Core

AWS IoT Core supporta connessioni con dispositivi IoT, gateway wireless, servizi e app. I dispositivi si connettono AWS IoT Core in modo da poter inviare e ricevere dati da AWS IoT servizi e altri dispositivi. Le app e altri servizi si connettono anche AWS IoT Core per controllare e gestire i dispositivi IoT ed elaborare i dati dalla tua soluzione IoT. Questa sezione descrive come scegliere il modo migliore per connettersi e comunicare AWS IoT Core per ogni aspetto della soluzione IoT.



Esistono diversi modi per interagire con. AWS IoT Le app e i servizi possono utilizzare Endpoint del piano di controllo AWS IoT Core e i dispositivi possono connettersi AWS IoT Core utilizzando AWS IoT endpoint del dispositivo o AWS IoT Core per le regioni e gli endpoint LoRa WAN.

Endpoint del piano di controllo AWS IoT Core

Gli endpoint AWS IoT Core- control plane forniscono l'accesso alle funzioni che controllano e gestiscono la AWS IoT soluzione.

- Endpoints

Gli endpoint del piano di controllo AWS IoT Core e del piano di controllo AWS IoT Core Device Advisor sono specifici per la regione e sono elencati nella sezione Endpoint e quote di AWS IoT Core. I formati degli endpoint sono i seguenti.

Scopo dell'endpoint	Formato dell'endpoint	Serve
piano di controllo AWS IoT Core	AWS IoT Endpoint Control Plane	AWS IoT API Control Plane
AWS IoT Core Device Advisor - piano di controllo	`api.iotdeviceadvisor.` *`aws-region`* `.amazonaws.com`	AWS IoT Core API del piano di controllo di Device Advisor

- IPv4 endpoint: gli IPv4 endpoint supportano solo il IPv4 traffico e sono disponibili per tutte le regioni.

  IPv4 gli endpoint utilizzano la seguente convenzione di denominazione:

  ```
  iot.aws-region.amazonaws.com
  ```

  Ad esempio, il nome dell' IPv4 endpoint per la regione us-east-1 è. `iot.us-east-1.amazonaws.com`
- Endpoint dual-stack (IPv4 e IPv6): gli endpoint dual-stack supportano sia il traffico che il traffico. IPv4 IPv6 Quando viene effettuata una richiesta a un endpoint dual-stack, l'URL dell'endpoint si risolve in un indirizzo o in un IPv6 indirizzo, a seconda del protocollo utilizzato dalla rete e dal client. IPv4

  Gli endpoint dual-stack utilizzano la seguente convenzione di denominazione:

  ```
  iot.aws-region.api.aws
  ```

  Ad esempio, il nome dell'endpoint dual-stack per la regione us-east-1 è. `iot.us-east-1.api.aws`
- SDKs e strumenti

AWS SDKsForniscono supporto specifico per lingua per e altri servizi. AWS IoT Core APIs APIs AWS The AWS Mobile SDKs offre agli sviluppatori di app un supporto specifico della piattaforma per l' AWS IoT Core API e altri servizi sui dispositivi mobili. AWS

AWS CLIFornisce l'accesso da riga di comando alle funzioni fornite dagli endpoint del servizio. AWS IoT AWS Tools for PowerShell fornisce strumenti per gestire AWS servizi e risorse nell'ambiente di scripting. PowerShell

- Autenticazione

  Gli endpoint del servizio utilizzano utenti e AWS credenziali IAM per autenticare gli utenti.

- Ulteriori informazioni

  Per ulteriori informazioni e collegamenti ai riferimenti SDK, consulta the section called “Connect agli endpoint AWS IoT Core del servizio”.

AWS IoT endpoint del dispositivo

Gli endpoint del AWS IoT dispositivo supportano la comunicazione tra i dispositivi IoT e AWS IoT.

- Endpoints

  Supporto AWS IoT Core e funzioni degli endpoint del dispositivo. AWS IoT Device Management Sono specifici per te Account AWS e puoi vedere cosa sono usando il describe-endpoint comando.

Scopo dell'endpoint	Formato dell'endpoint	Serve
Piano dati AWS IoT Core	Per informazioni, consulta ???.	AWS IoT API Data Plane
Dati dei processi AWS IoT Device Management	Per informazioni, consulta ???.	AWS IoT API Jobs Data Plane
AWS IoT Device Advisor - piano dati	Per informazioni, consulta ???.	Non applicabile

Scopo dell'endpoint	Formato dell'endpoint	Serve
Tunneling sicuro AWS IoT Device Management	`api.tunneling.iot.` *`aws-region`* `.amazonaws.com`	AWS IoT API di tunneling sicuro

Se utilizzi endpoint dual-stack (IPv4 and IPv6) per operazioni sul piano dati, usa il tipo di endpoint. `iot:Data-ATS` `iot:Jobs`può essere usato solo per. IPv4 Per ulteriori informazioni su questi endpoint e sulle funzioni che supportano, consulta the section called “AWS IoT dati del dispositivo e endpoint di servizio”.

- SDKs

  Il AWS IoT dispositivo SDKs fornisce un supporto specifico per la lingua per i protocolli Message Queueing Telemetry Transport (MQTT) e WebSocket Secure (WSS), con cui i dispositivi comunicano. AWS IoTAWS Mobile SDKsforniscono inoltre supporto per le comunicazioni con dispositivi MQTT e per altri servizi su dispositivi mobili. AWS IoT APIs APIs AWS

- Autenticazione

  Gli endpoint del dispositivo utilizzano certificati X.509 o utenti AWS IAM con credenziali per autenticare gli utenti.

- Ulteriori informazioni

  Per ulteriori informazioni e collegamenti ai riferimenti SDK, consulta the section called “AWS IoT Device SDKs”.

AWS IoT Core per gateway e dispositivi WAN LoRa

AWS IoT Core per LoRa WAN collega gateway e dispositivi wireless a. AWS IoT Core

- Endpoints

  AWS IoT Core for LoRa WAN gestisce le connessioni del gateway agli endpoint specifici dell'account e della regione AWS IoT Core . I gateway possono connettersi all'endpoint CUPS (Configuration and Update Server) dell'account fornito dalla rete WAN. AWS IoT Core LoRa

Scopo dell'endpoint	Formato dell'endpoint	Serve
Server di configurazione e aggiornamento (CUPS)	*account-specific-prefix* .cups.lorawan. *aws-region* .amazonaws.com:443	Comunicazione gateway con il server di configurazione e aggiornamento fornito da for WAN AWS IoT Core LoRa
LoRaServer di rete WAN (LNS)	*account-specific-prefix* .gateway.lorawan. *aws-region* .amazonaws.com:443	Comunicazione gateway con il server di rete LoRa WAN fornito da AWS IoT Core for LoRa WAN

- SDKs

  L'API AWS IoT wireless su cui è basata AWS IoT Core la LoRa rete WAN è supportata dall' AWS SDK. Per ulteriori informazioni, consulta AWS SDKs and Toolkits.

- Autenticazione

  AWS IoT Core per le comunicazioni con dispositivi LoRa WAN, utilizza i certificati X.509 per proteggere le comunicazioni con. AWS IoT

- Ulteriori informazioni

  Per ulteriori informazioni sulla configurazione e la connessione dei dispositivi wireless, consulta Regioni ed AWS IoT Core endpoint LoRa WAN.

Connect agli endpoint AWS IoT Core del servizio

Puoi accedere alle funzionalità del AWS IoT Core piano di controllo utilizzando l' AWS CLI AWS SDK nella tua lingua preferita o chiamando direttamente l'API REST. Ti consigliamo di utilizzare AWS CLI o un AWS SDK con cui interagire AWS IoT Core perché incorporano le migliori pratiche per le chiamate AWS ai servizi. La chiamata APIs diretta a REST è un'opzione, ma è necessario fornire le credenziali di sicurezza necessarie per consentire l'accesso all'API.

> **Note**
>
> I dispositivi IoT devono utilizzare AWS IoT Device SDKs. SDKs I dispositivi sono ottimizzati per l'uso sui dispositivi, supportano la comunicazione MQTT e supportano i dispositivi AWS

IoT APIs più utilizzati dai dispositivi. AWS IoT Per ulteriori informazioni sul Dispositivo SDKs e sulle funzionalità che offre, vedereAWS IoT Device SDKs.
I dispositivi mobili devono utilizzare AWS Mobile SDKs. The Mobile SDKs fornisce supporto per AWS IoT APIs le comunicazioni con dispositivi MQTT e altri AWS servizi sui dispositivi mobili. APIs Per ulteriori informazioni sul dispositivo mobile SDKs e sulle funzionalità che offre, consultaAWS Mobile SDKs.

Puoi utilizzare AWS Amplify strumenti e risorse nelle applicazioni web e mobili per connetterti più facilmente a AWS IoT Core. Per ulteriori informazioni sulla connessione AWS IoT Core tramite Amplify, PubSubconsulta la documentazione di Amplify.

Le sezioni seguenti descrivono gli strumenti SDKs che è possibile utilizzare per sviluppare e interagire con AWS IoT altri servizi. AWS Per l'elenco completo degli AWS strumenti e dei kit di sviluppo disponibili per creare e gestire app AWS, consulta Tools to Build on AWS.

AWS CLI per AWS IoT Core

AWS CLI Fornisce l'accesso da riga di comando a. AWS APIs

- Installazione

  Per informazioni su come installare AWS CLI, vedere Installazione di. AWS CLI
- Autenticazione

  AWS CLI Utilizza le credenziali del tuo Account AWS.
- Documentazione di riferimento

  Per informazioni sui AWS CLI comandi per questi AWS IoT Core servizi, vedi:
  - AWS CLI Riferimento ai comandi per IoT
  - AWS CLI Command Reference per dati IoT
  - AWS CLI Command Reference per i dati dei lavori IoT
  - AWS CLI Command Reference per il tunneling sicuro dell'IoT

Per gli strumenti per la gestione di AWS servizi e risorse nell'ambiente di PowerShell scripting, vedi AWS Tools for. PowerShell

AWS SDKs

Con AWS SDKs, le app e i dispositivi compatibili possono effettuare chiamate AWS IoT APIs e accedere APIs ad altri AWS servizi. Questa sezione fornisce collegamenti alla AWS SDKs e alla documentazione di riferimento APIs delle API per i AWS IoT Core servizi.

AWS SDKs Supportano questi AWS IoT Core APIs

- AWS IoT
- AWS IoT Piano dati
- AWS IoT Jobs Data Plane
- AWS IoT Tunneling sicuro
- AWS IoT Wireless

C++

Per installare il AWS SDK per C++ e usarlo per connetterti a AWS IoT:

1. Segui le istruzioni in Guida introduttiva all'uso dell' AWS SDK for C++

   Nelle seguenti istruzioni viene descritto come:

   - Installare e creare l'SDK dai file sorgente
   - Fornire le credenziali per utilizzare l'SDK con l' Account AWS
   - Inizializzare e chiudere l'SDK nella tua app o servizio
   - Crea un CMake progetto per creare la tua app o il tuo servizio
2. Creare ed eseguire un'app di esempio. Per le app di esempio che utilizzano SDK AWS per C++, consulta Codici di esempio AWS SDK per C++ SDK per C++.

Documentazione per i AWS IoT Core servizi che AWS SDK per C++ supporta

- AWS Documentazione di riferimento: :IoTClient»
- Documentazione di riferimento Aws: :Io TData Plane: TData PlaneClient :Io
- Documentazione di riferimento Aws: :Io TJobsDataPlane: :Io TJobs DataPlaneClient
- Documentazione di riferimento di Aws: :Io TSecure Tunneling: :Io TSecure TunnelingClient

Go

Per installare il AWS SDK per Go e usarlo per connetterti a AWS IoT:

1. Segui le istruzioni riportate in Getting Started with the AWS SDK per Go

   Nelle seguenti istruzioni viene descritto come:

   - Installa il AWS SDK per Go
   - I messaggi vengono archiviati in base al limite dell'account, i messaggi che superano tale limite vengono eliminati.
   - Poiché la velocità di consegna dei messaggi archiviati è limitata, saranno necessari diversi secondi per consegnare tutti i messaggi archiviati se una sessione ha più di 10 messaggi archiviati da consegnare dopo la riconnessione.
2. Creare ed eseguire un'app di esempio. Quando scade il tempo di scadenza di una sessione persistente.

Documentazione per i AWS IoT Core servizi che AWS SDK per Go supporta

- Documentazione di riferimento di IoT
- Documentazione di riferimento di Io TData Plane
- Documentazione TJobs DataPlane di riferimento di Io
- Documentazione di riferimento di Io TSecure Tunneling

Java

Per installare il AWS SDK per Java e usarlo per connetterti a AWS IoT:

1. Segui le istruzioni riportate in Guida introduttiva a AWS SDK for Java 2.x

   Nelle seguenti istruzioni viene descritto come:

   - Iscriviti AWS e crea un utente IAM
   - Scaricare l'SDK
   - Configura AWS credenziali e regione
   - Utilizzare gli SDK con Apache Maven
   - Utilizzare gli SDK con Gradle

2. Creare ed eseguire un'app di esempio utilizzando uno dei Codici di esempio AWS SDK for Java 2.x.
3. Esaminare la Documentazione di riferimento delle API SDK

Documentazione per i AWS IoT Core servizi che supporta AWS SDK per Java

- IotClient documentazione di riferimento
- IotDataPlaneClient documentazione di riferimento
- IotJobsDataPlaneClient documentazione di riferimento
- documentazione TSecure TunnelingClient di riferimento Io

JavaScript

Per installarlo AWS SDK per JavaScript e utilizzarlo per connettersi a AWS IoT:

1. Segui le istruzioni in Configurare AWS SDK per JavaScript. Queste istruzioni si applicano all'utilizzo di file AWS SDK per JavaScript nel browser e con Node.JS. Assicurarsi di seguire le istruzioni applicabili all'installazione.

   Nelle seguenti istruzioni viene descritto come:

   - Verificare i prerequisiti
   - Installa l'SDK per JavaScript
   - Carica l'SDK per JavaScript
2. Creare ed eseguire un'app di esempio per iniziare a utilizzare l'SDK come descritto dall'opzione introduttiva per il tuo ambiente.
   - Inizia a usare l'AWS SDK per JavaScript nel browser oppure
   - Inizia a usare l'AWS SDK per JavaScript in Node.js

Documentazione per i AWS IoT Core servizi che supporta AWS SDK per JavaScript

- `AWS.Iot reference documentation`
- `AWS.IotData reference documentation`
- `AWS.IotJobsDataPlane reference documentation`
- `AWS.IotSecureTunneling reference documentation`

.NET

Per installare il AWS SDK per .NET e usarlo per connetterti a AWS IoT:

1. Segui le istruzioni riportate in Configurazione dell' AWS SDK per .NET ambiente
2. Segui le istruzioni riportate nella sezione Configurazione del AWS SDK per .NET progetto

   Nelle seguenti istruzioni viene descritto come:

   - Iniziare un nuovo progetto
   - Ottieni e configura le AWS credenziali
   - Installa i pacchetti AWS SDK
3. Crea ed esegui uno dei programmi di esempio in Working with AWS services in the AWS SDK for .NET
4. Esaminare la Documentazione di riferimento delle API SDK

Documentazione per i AWS IoT Core servizi supportati AWS SDK per .NET

- Documentazione di riferimento Amazon.iot.model
- Amazon. IotData.Documentazione di riferimento sul modello
- Documentazione di riferimento Amazon.Io TJobs DataPlane .Model
- Documentazione di riferimento Amazon.Io Tunneling.Model TSecure

PHP

Per installare il AWS SDK per PHP e usarlo per connetterti a AWS IoT:

1. Segui le istruzioni riportate in Guida introduttiva alla versione 3 AWS SDK per PHP

   Nelle seguenti istruzioni viene descritto come:

   - Verificare i prerequisiti
   - Installazione dell'SDK
   - Applicare l'SDK a uno script PHP
2. Creare ed eseguire un'app di esempio utilizzando uno dei Esempi di codice di AWS SDK per PHP Versione 3

Documentazione per i AWS IoT Core servizi che AWS SDK per PHP supporta

- Documentazione TClient di riferimento Io
- Documentazione TData PlaneClient di riferimento Io
- Documentazione TJobs DataPlaneClient di riferimento Io
- Documentazione TSecure TunnelingClient di riferimento Io

Python

Per installare AWS SDK per Python (Boto3) e usarlo per connetterti a AWS IoT:

1. Segui le istruzioni riportate nella Guida introduttiva ad AWS SDK per Python (Boto3)

   Nelle seguenti istruzioni viene descritto come:

   - Installazione dell'SDK
   - Configurare l'SDK
   - Usare l'SDK nel tuo codice
2. Creare ed eseguire un programma di esempio che utilizza AWS SDK per Python (Boto3)

   Questo programma visualizza le opzioni di registrazione attualmente configurate dell'account. Dopo aver installato e configurato l'SDK per il tuo account, dovresti essere in grado di eseguire questo programma.

   ```
   import boto3
   import json

   # initialize client
   iot = boto3.client('iot')

   # get current logging levels, format them as JSON, and write them to stdout
   response = iot.get_v2_logging_options()
   print(json.dumps(response, indent=4))
   ```

   Per ulteriori informazioni sulle funzioni utilizzate in questo esempio, consulta the section called “Configurare la registrazione AWS IoT”.

Documentazione per i AWS IoT Core servizi che AWS SDK per Python (Boto3) supporta

- Documentazione di riferimento di IoT
- Documentazione di riferimento di Io TData Plane
- Documentazione TJobs DataPlane di riferimento di Io
- Documentazione di riferimento di Io TSecure Tunneling

Ruby

Per installare il AWS SDK per Ruby e usarlo per connetterti a AWS IoT:

- Segui le istruzioni riportate in Guida introduttiva a AWS SDK per Ruby

  Nelle seguenti istruzioni viene descritto come:
  - Installazione dell'SDK
  - Configurare l'SDK
- Creare ed eseguire il Hello World Tutorial

Documentazione per i AWS IoT Core servizi supportati dall' AWS SDK for Ruby

- Documentazione di riferimento di Aws::IoT::Client
- Documentazione di riferimento di Aws: :Io TData Plane: :Client
- Documentazione di riferimento Aws: :Io TJobsDataPlane: :Client
- Documentazione di riferimento di Aws: :Io TSecure Tunneling: :Client

AWS Mobile SDKs

The AWS Mobile SDKs fornisce agli sviluppatori di app mobili un supporto specifico per la piattaforma APIs dei servizi AWS IoT Core , la comunicazione dei dispositivi IoT tramite MQTT e altri servizi. APIs AWS

Android

AWS Mobile SDK per Android

AWS Mobile SDK per Android Contiene una libreria, esempi e documentazione per gli sviluppatori con cui creare applicazioni mobili connesse utilizzando. AWS Questo SDK include anche il

supporto per le comunicazioni con i dispositivi MQTT e la chiamata ai APIs servizi. AWS IoT Core Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS SDK per dispositivi mobili per Android su GitHub
- AWS SDK mobile per Android Readme
- AWS Esempi di SDK for Android per dispositivi mobili
- AWS Riferimento all'API SDK for Android
- AWSIoTClient Documentazione di riferimento della classe

iOS

AWS Mobile SDK for iOS

AWS Mobile SDK for iOS È un kit di sviluppo software open source, distribuito con una licenza Apache Open Source. L'SDK for iOS fornisce una libreria, esempi di codice e documentazione per aiutare gli sviluppatori a creare applicazioni AWS mobili connesse utilizzando. Questo SDK include anche il supporto per le comunicazioni con i dispositivi MQTT e la chiamata ai APIs servizi. AWS IoT Core Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS Mobile SDK for iOS su GitHub
- AWS Readme SDK for iOS
- AWS Esempi di SDK for iOS
- AWS IoT Documenti di riferimento per le classi nell' AWS SDK for iOS

RESTO APIs dei servizi AWS IoT Core

Il resto APIs dei AWS IoT Core servizi può essere richiamato direttamente utilizzando le richieste HTTP.

- URL dell'endpoint

  Gli endpoint del servizio che espongono il REST APIs dei AWS IoT Core servizi variano in base alla regione e sono elencati in AWS IoT Core Endpoints and Quotas. È necessario utilizzare l'endpoint per la regione che dispone delle risorse a cui si desidera accedere, poiché AWS IoT AWS IoT le risorse sono specifiche della regione.
- Autenticazione

Il RESTO APIs dei AWS IoT Core servizi utilizza le credenziali AWS IAM per l'autenticazione. Per ulteriori informazioni, consulta Firmare le richieste AWS API nella Guida AWS generale.

- Guida di riferimento alle API

  Per informazioni sulle funzioni specifiche fornite dal RESTO APIs dei AWS IoT Core servizi, consulta:

  - Riferimento API per IoT.
  - Riferimento API per dati IoT.
  - Riferimento API per i dati dei processi IoT.
  - Riferimento API per tunneling sicuro IoT.

Connect i dispositivi a AWS IoT

I dispositivi si connettono a AWS IoT e altri servizi tramite AWS IoT Core. Tramite AWS IoT Core, i dispositivi inviano e ricevono messaggi utilizzando endpoint specifici per il tuo account. the section called “AWS IoT Device SDKs” supportano le comunicazioni dei dispositivi utilizzando i protocolli MQTT e WSS. Per ulteriori informazioni sui protocolli utilizzabili dai dispositivi, consulta the section called “Protocolli di dispositivo di comunicazione”.

Broker di messaggi

AWS IoT gestisce la comunicazione tra dispositivi tramite un broker di messaggi. I dispositivi e i client pubblicano messaggi sul broker di messaggi e si iscrivono anche ai messaggi pubblicati dal broker di messaggi. I messaggi sono identificati da un'applicazione definita Argomento. Quando il broker di messaggi riceve un messaggio pubblicato da un dispositivo o da un client, lo pubblica nuovamente nei dispositivi e nei client che hanno effettuato la sottoscrizione all'argomento del messaggio. Il broker di messaggi inoltra anche i messaggi al motore AWS IoT delle regole, che può agire sul contenuto del messaggio.

AWS IoT sicurezza dei messaggi

Connessioni dei dispositivi da AWS IoT utilizzare the section called “Certificati client X.509” e AWS firma V4 per l'autenticazione. Le comunicazioni tra dispositivi sono protette dalla versione TLS 1.3 e AWS IoT richiedono ai dispositivi di inviare l'estensione SNI (Server Name Indication) quando si connettono. Per ulteriori informazioni, consulta Transport Security in. AWS IoT

AWS IoT dati del dispositivo e endpoint di servizio

> **Important**
>
> Puoi archiviare o memorizzare nella cache gli endpoint nel tuo dispositivo. Ciò significa che non sarà necessario interrogare l'API `DescribeEndpoint` ogni volta che viene connesso un nuovo dispositivo. Gli endpoint non cambieranno dopo averli AWS IoT Core creati per il tuo account.

Ogni account ha diversi endpoint del dispositivo che sono unici per l'account e supportano funzioni IoT specifiche. Gli endpoint di dati dei AWS IoT dispositivi supportano un publish/subscribe protocollo progettato per le esigenze di comunicazione dei dispositivi IoT; tuttavia, anche altri client, come app e servizi, possono utilizzare questa interfaccia se la loro applicazione richiede le funzionalità specializzate fornite da questi endpoint. Gli endpoint di assistenza ai AWS IoT dispositivi supportano l'accesso incentrato sui dispositivi ai servizi di sicurezza e gestione.

Per conoscere l'endpoint dei dati del dispositivo del tuo account, puoi trovarlo nella pagina Impostazioni della tua console. AWS IoT Core

Per conoscere l'endpoint del dispositivo dell'account per uno scopo specifico, incluso l'endpoint dei dati del dispositivo, usa il comando CLI describe-endpoint mostrato qui, o il comando API REST `DescribeEndpoint` e fornire il valore del parametro *`endpointType`* dalla tabella seguente.

```
aws iot describe-endpoint --endpoint-type endpointType
```

Questo comando restituisce un file *`iot-endpoint`* nel seguente formato:*`account-specific-prefix`*`.iot.`*`aws-region`*`.amazonaws.com`.

Ogni cliente ha un `iot:Data-ATS` e un endpoint `iot:Data`. Ogni endpoint utilizza un certificato X.509 per autenticare il client. Si consiglia vivamente ai clienti di utilizzare il tipo di endpoint `iot:Data-ATS` più recente per evitare problemi legati alla diffidenza diffusa nei confronti delle autorità di certificazione Symantec. Forniamo l'`iot:Data`endpoint per i dispositivi per recuperare dati da vecchi endpoint che utilizzano VeriSign certificati per la compatibilità con le versioni precedenti. Per ulteriori informazioni, consulta Autenticazione del server.

AWS IoT endpoint per dispositivi

Scopo dell'endpoint	***endpointType*** value	Description
Operazioni del piano dati AWS IoT Core	`iot:Data-ATS`	Utilizzato per inviare e ricevere dati da e verso i component i Broker di messaggi, Device Shadow e Motore di regole, di AWS IoT. `iot:Data-ATS` restituisce un endpoint di dati firmati con ATS.
operazioni del piano dati AWS IoT Core(legacy)	`iot:Data`	`iot:Data`restituisce un endpoint di dati VeriSign firmato fornito per la compatibi lità con le versioni precedent i. MQTT 5 non è supportat o su endpoint Symantec (`iot:Data`).
AWS IoT Core accesso tramite credenziali	`iot:CredentialProv ider`	Utilizzato per scambiare il certificato X.509 incorporato di un dispositivo con credenzia li temporanee per connettersi direttamente con altri servizi AWS . Per ulteriori informazi oni sulla connessione ad altri AWS servizi, vedere Autorizza zione delle chiamate dirette ai AWS servizi.
Operazioni di dati dei processi AWS IoT Device Management	`iot:Jobs`	Utilizzato per consentire ai dispositivi di interagire con il servizio AWS IoT Jobs utilizzando il Jobs Device HTTPS APIs.

Scopo dell'endpoint	*endpointType* value	Description
		`iot:Jobs`può essere utilizzato IPv4 solo per. Se utilizzi endpoint dual-stack (IPv4 and IPv6), usa il tipo di endpoint. `iot:Data-ATS`
AWS IoT Operazioni Device Advisor	`iot:DeviceAdvisor`	Tipo di endpoint di prova utilizzato per testare i dispositivi con Device Advisor. Per ulteriori informazioni, consulta ???.
AWS IoT Core data beta (anteprima)	`iot:Data-Beta`	Un tipo di endpoint riservato alle versioni beta. Per informazioni sul suo utilizzo corrente, consulta ???.

Puoi anche utilizzare il tuo nome di dominio completo (FQDN), ad esempio*example.com*, e il certificato del server associato a cui connettere i dispositivi utilizzando. AWS IoT the section called "Configurazioni del dominio"

AWS IoT Device SDKs

Il AWS IoT dispositivo ti SDKs aiuta a connettere i tuoi dispositivi IoT AWS IoT Core e supporta i protocolli MQTT e MQTT su WSS.

Il AWS IoT dispositivo SDKs si differenzia dal fatto che SDKs supporta le esigenze di comunicazione specializzate dei dispositivi IoT, ma non supporta tutti i servizi supportati da AWS SDKs. AWS SDKs AWS IoT I AWS IoT dispositivi SDKs sono compatibili con quelli AWS SDKs che supportano tutti i AWS servizi; tuttavia, utilizzano metodi di autenticazione diversi e si connettono a diversi endpoint, il che potrebbe rendere AWS SDKs impraticabile l'utilizzo su un dispositivo IoT.

Dispositivi mobili

the section called "AWS Mobile SDKs"Supportano sia le comunicazioni dei dispositivi MQTT, alcuni AWS IoT servizi APIs, sia quelle APIs di altri servizi. AWS Se stai sviluppando su un dispositivo

mobile supportato, consulta il suo SDK per verificare se è l'opzione migliore per lo sviluppo della tua soluzione IoT.

C++

AWS IoT SDK per dispositivi C++

Il AWS IoT C++ Device SDK consente agli sviluppatori di creare applicazioni AWS connesse utilizzando e i APIs servizi. AWS IoT Core In particolare, questo SDK è stato progettato per i dispositivi che non hanno vincoli di risorse e che richiedono caratteristiche avanzate come l'accodamento dei messaggi, il supporto per il multithreading e le più recenti caratteristiche di linguaggio. Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT Device SDK C++ v2 attivo GitHub
- AWS IoT Readme del dispositivo SDK C++ v2
- AWS IoT Esempi di Device SDK C++ v2
- AWS IoT Documentazione sull'API Device SDK C++ v2

Python

AWS IoT Device SDK per Python

Il AWS IoT Device SDK for Python consente agli sviluppatori di scrivere script Python per utilizzare i propri dispositivi per accedere alla piattaforma tramite MQTT o MQTT tramite AWS IoT il protocollo Secure (WSS). WebSocket Collegando i propri dispositivi ai AWS IoT Core servizi, gli utenti possono lavorare in sicurezza con il broker APIs di messaggi, le regole e il servizio Device Shadow che AWS IoT Core fornisce e con altri AWS servizi come AWS Lambda Amazon Kinesis e Amazon S3 e altro ancora.

- AWS IoT Device SDK per Python v2 attivo GitHub
- AWS IoT Device SDK per Python v2 Readme
- AWS IoT Device SDK per esempi in Python v2
- AWS IoT Documentazione dell'API Device SDK per Python v2

JavaScript

AWS IoT Device SDK per JavaScript

Il AWS IoT Device SDK for JavaScript consente agli sviluppatori di scrivere JavaScript applicazioni che accedono APIs a o AWS IoT Core utilizzando MQTT o MQTT tramite il protocollo. WebSocket Questo SDK può essere usato nelle applicazioni di tipo browser e negli ambienti Node.js. Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT Device SDK per la versione 2 attivo JavaScript GitHub
- AWS IoT SDK del dispositivo per Readme v2 JavaScript
- AWS IoT Device SDK per esempi per la versione 2 JavaScript
- AWS IoT Documentazione relativa all'API Device SDK per la versione 2 JavaScript

Java

AWS IoT SDK per dispositivi per Java

Il AWS IoT Device SDK for Java consente agli sviluppatori Java di accedervi APIs tramite MQTT o MQTT tramite AWS IoT Core il protocollo. WebSocket L'SDK supporta il servizio Device Shadow. Puoi accedere alle copie shadow tramite i metodi HTTP, tra cui GET, UPDATE e DELETE. L'SDK supporta anche un modello di accesso semplificato alle copie shadow che permette agli sviluppatori di scambiare i dati con le copie shadow semplicemente utilizzando i metodi getter e setter, senza dover serializzare o deserializzare documenti JSON. Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT SDK del dispositivo per Java v2 attivo GitHub
- AWS IoT SDK del dispositivo per Java v2 Readme
- AWS IoT Esempi di Device SDK for Java v2
- AWS IoT Documentazione sull'API Device SDK for Java v2

Embedded C

AWS IoT Device SDK per Embedded C

> **Important**
>
> Questo SDK è destinato all'uso da parte di sviluppatori di software integrati esperti.

Il SDK per dispositivi AWS IoT per Embedded C (C-SDK) è una raccolta di file sorgente C con licenza open source MIT che può essere utilizzata in applicazioni integrate per connettere in modo sicuro i dispositivi IoT a AWS IoT Core. Include le librerie MQTT, JSON Parser e AWS IoT Device Shadow e altre. Viene distribuito come codice sorgente ed è pensato per essere integrato nel firmware del cliente con il codice dell'applicazione, altre librerie ed eventualmente un Real Time Operating System (RTOS).

SDK per dispositivi AWS IoT per Embedded C È generalmente destinato a dispositivi con risorse limitate che richiedono un runtime ottimizzato in linguaggio C. È possibile utilizzare l'SDK su qualsiasi sistema operativo e ospitarlo su qualsiasi tipo di processore (ad esempio, MCUs e). MPUs Se il tuo dispositivo dispone di memoria e risorse di elaborazione sufficienti, ti consigliamo di utilizzare uno degli altri AWS IoT dispositivi e dispositivi mobili SDKs, ad esempio AWS IoT Device SDK for C++, JavaScript Java o Python.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT Device SDK per Embedded C su GitHub
- AWS IoT SDK del dispositivo per il file Readme C incorporato
- AWS IoT SDK del dispositivo per esempi C incorporati

Protocolli di dispositivo di comunicazione

AWS IoT Core supporta dispositivi e client che utilizzano i protocolli MQTT e MQTT over WebSocket Secure (WSS) per pubblicare e sottoscrivere messaggi e dispositivi e client che utilizzano il protocollo HTTPS per pubblicare messaggi. Tutti i protocolli supportano e. IPv4 IPv6 In questa sezione vengono descritte le diverse opzioni di connessione per dispositivi e client.

Versioni del protocollo TLS

AWS IoT Core utilizza la versione TLS 1.2 e la versione TLS 1.3 per crittografare tutte le comunicazioni. È possibile configurare versioni aggiuntive delle policy TLS per l'endpoint configurando le impostazioni TLS nelle configurazioni del dominio. Quando si collegano i dispositivi AWS IoT Core, i client possono inviare l'estensione Server Name Indication (SNI), necessaria per funzionalità come la registrazione di più account, endpoint configurabili, domini personalizzati ed endpoint VPC. Per ulteriori informazioni, vedere Protezione del trasporto in AWS IoT.

AWS IoT Device SDKs supportano MQTT e MQTT su WSS e supportano i requisiti di sicurezza delle connessioni client. Si consiglia di utilizzare il AWS IoT Device SDKs per connettere i client a AWS IoT.

Protocolli, mappature delle porte e autenticazione

Il modo in cui un dispositivo o un client si connette al broker di messaggi è configurabile utilizzando un tipo di autenticazione. Per impostazione predefinita o quando non viene inviata alcuna estensione SNI, il metodo di autenticazione si basa sul protocollo dell'applicazione, sulla porta e sull'estensione TLS Application Layer Protocol Negotiation (ALPN) utilizzati dai dispositivi. La tabella seguente elenca l'autenticazione prevista in base a porta, porta e ALPN.

Mappature tra protocolli, autenticazione e porte

Protocollo	Operazioni supportate	Autenticazione	Porta	Nome del protocollo ALPN
MQTT su WebSocket	Pubblicazione, sottoscrizione	Signature Version 4	443	N/D
MQTT finito WebSocket	Pubblicazione, sottoscrizione	Autenticazione personalizzata	443	N/D
MQTT	Pubblicazione, sottoscrizione	Certificato client X.509	443†	`x-amzn-mqtt-ca`
MQTT	Pubblicazione, sottoscrizione	Certificato client X.509	8883	N/D
MQTT	Pubblicazione, sottoscrizione	Autenticazione personalizzata	443†	`mqtt`
HTTPS	Solo pubblicazione	Signature Version 4	443	N/D
HTTPS	Solo pubblicazione	Certificato client X.509	443†	`x-amzn-http-ca`
HTTPS	Solo pubblicazione	Certificato client X.509	8443	N/D

Protocollo	Operazioni supportate	Autenticazione	Porta	Nome del protocollo ALPN
HTTPS	Solo pubblicazione	Autenticazione personalizzata	443	N/D

Application Layer Protocol Negotiation (ALPN)

† Quando si utilizzano configurazioni endpoint predefinite, i client che si connettono alla porta 443 con l'autenticazione del certificato client X.509 devono implementare l'estensione TLS Application Layer Protocol Negotiation (ALPN) e utilizzare il nome del protocollo ALPN elencato nell'ALPN inviato dal client come parte del messaggio. ProtocolNameList `ClientHello`
Sulla porta 443, l'endpoint IoT:Data-ATS supporta ALPN HTTP, ma l'endpoint IoT:Jobs no. x-amzn-http-ca
Sulla porta 8443 HTTPS e sulla porta 443 MQTT con ALPN, l'autenticazione personalizzata non può essere utilizzata. x-amzn-mqtt-ca

I client si connettono agli endpoint dei propri dispositivi. Account AWS Consulta the section called “AWS IoT dati del dispositivo e endpoint di servizio” per informazioni su come trovare gli endpoint del dispositivo dell'account.

Note

AWS SDKs non richiedono l'intero URL. Richiedono solo il nome host dell'endpoint, ad `pubsub.py`esempio per AWS IoT Device SDK for Python on. GitHub Il passaggio dell'intero URL come indicato nella tabella seguente può generare un errore come nome host non valido.

Connessione a AWS IoT Core

Protocollo	Endpoint o URL
MQTT	*iot-endpoint*

Protocollo	Endpoint o URL
MQTT su WSS	`wss://iot-endpoint /mqtt`
HTTPS	`https://iot-endpoint /topics`

Scelta di un protocollo applicativo per la comunicazione del dispositivo

Per la maggior parte delle comunicazioni tra dispositivi IoT attraverso gli endpoint del dispositivo, ti consigliamo di utilizzare i protocolli Secure MQTT o MQTT over WebSocket Secure (WSS); tuttavia, gli endpoint del dispositivo supportano anche HTTPS.

La tabella seguente confronta il modo in cui vengono AWS IoT Core utilizzati i due protocolli di alto livello (MQTT e HTTPS) per la comunicazione tra dispositivi.

AWS IoT protocolli dei dispositivi (MQTT e HTTPS) side-by-side

Funzionalità	MQTT	HTTPS
Supporto pubblicazione/sottoscrizione	Pubblicazione e sottoscrizione	Solo pubblicazione
Supporto SDK	AWS Il dispositivo SDKs supporta i protocolli MQTT e WSS	Nessun supporto SDK, ma è possibile utilizzare metodi specifici della lingua per effettuare richieste HTTPS
Supporto Qualità del servizio	QoS MQTT livelli 0 e 1	QoS è supportato passando un parametro di stringa di query `?qos=qos` dove il valore può essere 0 o 1. È possibile aggiungere questa stringa di query per pubblicare un messaggio con il valore QoS desiderato.
È possibile ricevere messaggi mancanti mentre il dispositivo era offline	Sì	No

Funzionalità	MQTT	HTTPS
Supporto del campo `clientId`	Sì	No
Rilevamento disconnessione dispositivo	Sì	No
Sicurezza delle comunicazioni	Sì. Consulta ???	Sì. Consulta ???
Definizioni degli argomenti	Applicazione definita	Applicazione definita
Formato dei dati del messaggio	Applicazione definita	Applicazione definita
Sovraccarico del protocollo	Più basso	Più alto
Consumo energetico	Più basso	Più alto

Scelta del tipo di autenticazione per la comunicazione del dispositivo

Puoi configurare il tipo di autenticazione per il tuo endpoint IoT utilizzando endpoint configurabili. In alternativa, utilizza la configurazione predefinita e determina in che modo i dispositivi si autenticano con la combinazione di protocollo applicativo, porta e estensione TLS ALPN. Il tipo di autenticazione scelto determina il modo in cui i dispositivi si autenticheranno durante la connessione a. AWS IoT Core Esistono cinque tipi di autenticazione:

certificato X.509

Autentica i dispositivi utilizzando i certificati client X.509, che AWS IoT Core convalidano l'autenticazione del dispositivo. Questo tipo di autenticazione funziona con i protocolli Secure MQTT (MQTT over TLS) e HTTPS.

Certificato X.509 con autorizzatore personalizzato

Autentica i dispositivi utilizzando certificati client X.509 ed esegui azioni di autenticazione aggiuntive utilizzando un autorizzatore personalizzato, che riceverà le informazioni sul certificato client X.509. Questo tipo di autenticazione funziona con i protocolli Secure MQTT (MQTT over TLS) e HTTPS. Questo tipo di autenticazione è possibile solo utilizzando endpoint configurabili con autenticazione personalizzata X.509. Non è disponibile alcuna opzione ALPN.

AWS Signature versione 4 (SigV4)

Autentica i dispositivi utilizzando Cognito o il tuo servizio di backend, supportando la federazione sociale e aziendale. Questo tipo di autenticazione funziona con i protocolli MQTT over WebSocket Secure (WSS) e HTTPS.

Autorizzatore personalizzato

Autentica i dispositivi configurando una funzione Lambda per elaborare le informazioni di autenticazione personalizzate inviate a. AWS IoT Core Questo tipo di autenticazione funziona con i protocolli Secure MQTT (MQTT over TLS), HTTPS e MQTT over Secure (WSS). WebSocket

Impostazione predefinita

Autentica i dispositivi in base all'estensione ALPN (Port and/or Application Layer Protocol Negotiation) utilizzata dai dispositivi. Alcune opzioni di autenticazione aggiuntive non sono supportate. Per ulteriori informazioni, consulta ???.

La tabella seguente mostra tutte le combinazioni supportate di tipi di autenticazione e protocolli applicativi.

Combinazioni supportate di tipi di autenticazione e protocolli applicativi

Tipo di autenticazione	MQTT sicuro (MQTT su TLS)	MQTT su WebSocket Secure (WSS)	HTTPS	Predefinita
certificato X.509	✓		✓	
Certificato X.509 con autorizzatore personalizzato	✓		✓	
AWS Signature versione 4 (SigV4)		✓	✓	
Autorizzatore personalizzato	✓	✓	✓	

Tipo di autenticazione	MQTT sicuro (MQTT su TLS)	MQTT su WebSocket Secure (WSS)	HTTPS	Predefinita
Impostazione predefinita				✓

Limiti di durata della connessione

Le connessioni HTTPS non durano più a lungo del tempo necessario per ricevere e rispondere alle richieste.

La durata della connessione MQTT dipende dalla caratteristica di autenticazione utilizzata. La tabella seguente elenca la durata massima della connessione nelle condizioni ideali per ciascuna caratteristica.

Durata della connessione MQTT per caratteristica di autenticazione

Funzionalità	Durata massima *
Certificato client X.509	1—2 settimane
Autenticazione personalizzata	1—2 settimane
Signature Version 4	Fino a 24 ore

* Non garantito

Con i certificati X.509 e l'autenticazione personalizzata, la durata della connessione non ha limiti rigidi, ma può durare pochi minuti. Per vari motivi si possono verificare interruzioni di connessione. L'elenco seguente contiene alcuni dei motivi più comuni.

- Interruzioni di disponibilità Wi-Fi
- Interruzioni della connessione fornitore di servizi Internet (ISP)
- Patch di servizi
- Implementazioni di servizi
- Scalabilità automatica del servizio

- Host del servizio non disponibile
- Problemi e aggiornamenti del bilanciamento del carico
- Errori lato client

I dispositivi devono implementare strategie per rilevare le disconnessioni e la riconnessione. Per informazioni sulla disconnessione di eventi e indicazioni su come gestirli, consulta ??? in ???.

MQTT

MQTT (Message Queuing Telemetry Transport) è un protocollo di messaggistica leggero e ampiamente adottato, progettato per dispositivi vincolati. Il supporto AWS IoT Core per MQTT si basa sulla specifica MQTT v3.1.1 e la specifica MQTT v5.0, con alcune differenze come documentato in the section called “AWS IoT differenze rispetto alle specifiche MQTT”. Essendo la versione più recente dello standard, MQTT 5 introduce diverse funzionalità chiave che rendono un sistema basato su MQTT più affidabile, inclusi nuovi miglioramenti della scalabilità, segnalazione degli errori migliorata con risposte codice motivo, timer di scadenza dei messaggi e delle sessioni e intestazioni messaggi utente personalizzate. Per ulteriori informazioni sulle funzionalità supportate da MQTT 5, consultate Funzionalità AWS IoT Core supportate da MQTT 5. AWS IoT Core supporta anche la comunicazione tra versioni MQTT (MQTT 3 e MQTT 5). Un autore MQTT 3 può inviare un messaggio MQTT 3 a un sottoscrittore MQTT 5 che riceverà un messaggio di pubblicazione MQTT 5, e viceversa.

AWS IoT Core supporta connessioni di dispositivi che utilizzano il protocollo MQTT e il protocollo MQTT over WSS e che sono identificate da un ID client. AWS IoT Device SDKs supportano entrambi i protocolli e sono i modi consigliati per connettere i dispositivi a AWS IoT Core. Il AWS IoT dispositivo SDKs supporta le funzioni necessarie ai dispositivi e ai client per connettersi e accedere ai servizi. AWS IoT Il dispositivo SDKs supporta i protocolli di autenticazione richiesti dai AWS IoT servizi e i requisiti di ID di connessione richiesti dal protocollo MQTT e dai protocolli MQTT su WSS. Per informazioni su come connettersi AWS IoT utilizzando il AWS dispositivo SDKs e collegamenti ad esempi delle lingue AWS IoT supportate, vedere. the section called “Connessione con MQTT tramite il dispositivo AWS IoT SDKs” Per ulteriori informazioni sull’autenticazione e sulla mappatura delle porte per i messaggi MQTT, consulta ???.

Sebbene consigliamo di utilizzare il AWS IoT Dispositivo SDKs a cui connettersi AWS IoT, non sono necessari. Se non si utilizza il AWS IoT Dispositivo SDKs, tuttavia, è necessario fornire la necessaria sicurezza di connessione e comunicazione. I client devono inoltre inviare la Estensione TLS di Server Name Indication per impedire che le connessioni vengano rifiutate. I tentativi di connessione

che non includono l'SNI vengono rifiutati. Per ulteriori informazioni, vedere Transport Security in AWS IoT. I client che utilizzano utenti e AWS credenziali IAM per autenticare i client devono fornire l'autenticazione Signature Version 4 corretta.

Dopo aver connesso i client, puoi monitorare e gestire le loro connessioni client MQTT utilizzando. APIs Per ulteriori informazioni, consulta the section called “Gestione delle connessioni MQTT”.

In questo argomento:

- Connessione con MQTT tramite il dispositivo AWS IoT SDKs
- Opzioni di qualità del servizio MQTT (QoS)
- Sessioni persistenti MQTT
- Messaggi conservati da MQTT
- Messaggi MQTT Last Will and Testament (LWT)
- Utilizzo degli Attributi Connect
- Caratteristiche supportate da MQTT 5
- Proprietà MQTT 5
- Codici motivo MQTT
- AWS IoT differenze rispetto alle specifiche MQTT
- Gestione delle connessioni MQTT

Connessione con MQTT tramite il dispositivo AWS IoT SDKs

Questa sezione contiene collegamenti al AWS IoT dispositivo SDKs e al codice sorgente di programmi di esempio che illustrano come collegare un dispositivo a. AWS IoT Le app di esempio collegate qui mostrano come connettersi AWS IoT utilizzando il protocollo MQTT e MQTT tramite WSS.

> **Note**
>
> The AWS IoT Device SDKs ha rilasciato un client MQTT 5.

C++

Utilizzo del AWS IoT C++ Device SDK per connettere i dispositivi

- Codice sorgente di un'app di esempio che mostra un esempio di connessione MQTT in C++

- AWS IoT SDK del dispositivo per C++ v2 attivo GitHub

Python

Utilizzo del AWS IoT Device SDK per Python per connettere i dispositivi

- Codice sorgente di un'app di esempio che mostra un esempio di connessione MQTT in Python
- AWS IoT Device SDK v2 per Python attivo GitHub

JavaScript

Utilizzo del AWS IoT Device SDK per connettere i dispositivi JavaScript

- Codice sorgente di un'app di esempio che mostra un esempio di connessione MQTT in JavaScript
- AWS IoT Device SDK per JavaScript la versione 2 attivo GitHub

Java

Utilizzo del AWS IoT Device SDK for Java per connettere i dispositivi

> **Note**
>
> Il AWS IoT Device SDK for Java v2 ora supporta lo sviluppo Android. Per ulteriori informazioni, consulta AWS IoT Device SDK for Android.

- Codice sorgente di un'app di esempio che mostra un esempio di connessione MQTT in Java
- AWS IoT SDK del dispositivo per Java v2 attivo GitHub

Embedded C

Utilizzo del AWS IoT Device SDK for Embedded C per connettere i dispositivi

> **Important**
>
> Questo SDK è destinato all'uso da parte di sviluppatori esperti di software integrato.

- Codice sorgente di un'app di esempio che mostra un esempio di connessione MQTT in Embedded C
- AWS IoT Device SDK per Embedded C attivo GitHub

Opzioni di qualità del servizio MQTT (QoS)

AWS IoT e il AWS IoT dispositivo SDKs supporta i livelli di qualità del servizio (QoS) 0 MQTT e. 1 Il protocollo MQTT definisce un terzo livello di QoS, 2 livello, AWS IoT ma non lo supporta. Solo il protocollo MQTT supporta la caratteristica QoS. HTTPS supporta QoS passando un parametro di stringa di query `?qos=qos` dove il valore può essere 0 o 1.

Questa tabella descrive come ogni livello QoS influisce sui messaggi pubblicati da e per il broker messaggi.

Con un livello QoS di...	Il messaggio è...	Commenti
QoS livello 0	Inviato zero o più volte	Questo livello deve essere utilizzato per i messaggi inviati tramite collegamenti di comunicazione affidabili o che possono essere persi senza problemi.
Livello QoS 1	Inviato almeno una volta e poi ripetutamente fino a quando non si riceve una risposta `PUBACK`	Il messaggio non viene considerato completo fino a quando il mittente non riceve una risposta `PUBACK` per indicare la riuscita della consegna.

Sessioni persistenti MQTT

Le sessioni persistenti archiviano le sottoscrizioni e i messaggi di un client, con una qualità del servizio (QoS) pari a 1, che non sono stati riconosciuti dal client. Quando il dispositivo si riconnette a una sessione persistente, la sessione riprende, le relative sottoscrizioni vengono reintegrate e i messaggi sottoscritti non riconosciuti ricevuti e archiviati prima della riconnessione vengono inviati al client.

L'elaborazione dei messaggi archiviati viene registrata in CloudWatch metriche e registri. CloudWatch Per informazioni sulle voci scritte in CloudWatch and CloudWatch Logs, vedere e. Parametri del broker di messaggi Voce di log Queued

Creazione di una sessione persistente

In MQTT3, è possibile creare una sessione persistente MQTT inviando un messaggio `CONNECT` e impostando il flag `cleanSession` su `0`. Se non esiste alcuna sessione per il client che invia il messaggio `CONNECT`, viene creata una nuova sessione persistente. Se esiste una sessione per il client, il client riprende la sessione esistente. Per creare una sessione pulita, è possibile inviare un messaggio `CONNECT` e impostare il flag `cleanSession` su `1`. Il broker non memorizzerà alcuno stato della sessione quando il client si disconnette.

In MQTT 5, è possibile gestire le sessioni persistenti impostando il flag `Clean Start` e `Session Expiry Interval`. Avvio pulito controlla l'inizio della sessione di connessione e la fine della sessione precedente. Quando si imposta `Clean Start = 1`, viene creata una nuova sessione e una sessione precedente viene terminata, se esiste. Quando si imposta `Clean Start= 0`, la sessione di connessione riprende una sessione precedente, se esiste. L'intervallo di scadenza della sessione controlla la fine della sessione di connessione. L'intervallo di scadenza della sessione specifica il tempo, in secondi (numero intero a 4 byte), di persistenza di una sessione dopo la disconnessione. L'impostazione `Session Expiry interval=0` causa la terminazione immediata della sessione dopo la disconnessione. Se l'intervallo di scadenza della sessione non è specificato nel messaggio CONNECT, il valore predefinito è 0.

Avvio pulito e scadenza della sessione MQTT 5

Valore proprietà	Description
`Clean Start= 1`	Crea una nuova sessione e termina una sessione precedente, se una esiste.
`Clean Start= 0`	Ripristina una sessione, se esiste una sessione precedente.
`Session Expiry Interval> 0`	Mantiene una sessione.
`Session Expiry interval= 0`	Non mantiene una sessione.

In MQTT 5, se si imposta `Clean Start` = `1` e `Session Expiry Interval` = `0`, questo è l'equivalente di una sessione pulita MQTT 3. Se si imposta `Clean Start` = `0` e `Session Expiry Interval`> `0`, è l'equivalente di una sessione persistente MQTT 3.

> **Note**
>
> Le sessioni persistenti tra versioni MQTT (MQTT 3 e MQTT 5) non sono supportate. Una sessione persistente MQTT 3 non può essere ripristinata come una sessione MQTT 5, e viceversa.

Operazioni durante una sessione persistente

I dispositivi utilizzano l'attributo `sessionPresent` nel messaggio `CONNACK` (Connection Acknowledged) per determinare l'eventuale presenza di una sessione persistente. Se `sessionPresent` è `1`, è presente una sessione persistente e tutti i messaggi archiviati per il client vengono distribuiti al client dopo che il client riceve il `CONNACK`, come descritto in Traffico dei messaggi dopo la riconnessione a una sessione persistente. Se `sessionPresent` è impostato su `1`, non è necessario che il client esegua di nuovo la sottoscrizione. Se `sessionPresent` è impostato su `0`, non è presente alcuna sessione persistente e il client deve eseguire nuovamente la sottoscrizione nei filtri degli argomenti.

Dopo che il client si unisce ad una sessione persistente, può pubblicare messaggi e sottoscrivere filtri argomento senza ulteriori flag in ogni operazione.

Traffico dei messaggi dopo la riconnessione a una sessione persistente

Una sessione persistente rappresenta una connessione continua tra un cliente e un broker di messaggi MQTT. Quando un client si connette a un broker di messaggi utilizzando una sessione persistente, il broker di messaggi salva tutte le sottoscrizioni che il client effettua durante la connessione. Quando il client si disconnette, il broker di messaggi archivia i messaggi QoS 1 non riconosciuti e i nuovi messaggi QoS 1 pubblicati in argomenti di cui il client ha eseguito la sottoscrizione. I messaggi vengono archiviati in base al limite dell'account. I messaggi che superano il limite verranno eliminati. Per ulteriori informazioni sui limiti dei messaggi persistenti, consulta Endpoint e quote di AWS IoT Core. Quando il client si riconnette alla sessione persistente, tutte le sottoscrizioni vengono ripristinate e tutti i messaggi archiviati vengono inviati al client a una velocità massima di 10 messaggi al secondo. In MQTT 5, se un QoS 1 in uscita con Intervallo di scadenza dei messaggi scade quando un client è offline, dopo la ripresa della connessione, il client non riceverà il messaggio scaduto.

Dopo la riconnessione, i messaggi archiviati vengono inviati al client, a una velocità limitata di 10 messaggi archiviati al secondo, insieme a qualsiasi traffico di messaggi corrente fino a quando il limite di `Publish requests per second per connection` viene raggiunto. Poiché la velocità di distribuzione dei messaggi archiviati è limitata, ci vorranno alcuni secondi per distribuire tutti i messaggi archiviati se una sessione ha più di 10 messaggi archiviati da distribuire dopo la riconnessione.

Per gli abbonati condivisi, i messaggi verranno messi in coda se almeno un abbonato di un gruppo utilizza una sessione persistente e nessun abbonato è online per ricevere il messaggio QoS 1. La rimozione dalla coda dei messaggi viene effettuata a una velocità massima di 20 messaggi al secondo per abbonato attivo in un gruppo. Per ulteriori informazioni, consulta Accodamento dei messaggi relativi alle sottoscrizioni condivise.

Termine di una sessione persistente

Le sessioni persistenti possono terminare nei modi seguenti:

- Il tempo di scadenza della sessione persistente è trascorso. Il timer di scadenza della sessione persistente inizia quando il broker messaggi rileva che un client si è disconnesso, tramite la disconnessione del client o il timeout della connessione.
- Il client invia un messaggio `CONNECT` che imposta il flag `cleanSession` su `1`.
- Si disconnette manualmente il client e si cancella la sessione utilizzando `DeleteConnection` l'API. Per ulteriori informazioni, consulta the section called “Gestione delle connessioni MQTT”.

In MQTT 3, il valore predefinito del tempo di scadenza delle sessioni persistenti è un'ora e si applica a tutte le sessioni nell'account.

In MQTT 5, è possibile impostare Intervallo di scadenza della sessione per ogni sessione sui pacchetti CONNECT e DISCONNECT.

Per intervallo di scadenza della sessione sul pacchetto DISCONNECT:

- Se l'intervallo di scadenza della sessione corrente è impostato su 0, non è possibile impostare un intervallo di scadenza della sessione su un valore maggiore di 0 sul pacchetto DISCONNECT.
- Se l'intervallo di scadenza della sessione corrente è maggiore di 0 e si imposta l'intervallo di scadenza della sessione su 0 nel pacchetto DISCONNECT, la sessione verrà terminata su DISCONNECT.

- In caso contrario, l'Intervallo di scadenza della sessione sul pacchetto DISCONNECT aggiornerà l'Intervallo di scadenza della sessione corrente.

> **Note**
>
> I messaggi archiviati in attesa di essere inviati al client al termine di una sessione vengono eliminati; tuttavia, vengono comunque fatturati alla tariffa standard di messaggistica, anche se non è stato possibile inviarli. Per ulteriori informazioni sui prezzi delle prenotazioni, consulta Prezzi di AWS IoT Core. È possibile configurare l'intervallo del tempo di scadenza.

Riconnessione dopo la scadenza di una sessione persistente

Se un client non si riconnette alla sessione persistente prima della scadenza, la sessione termina e i messaggi archiviati vengono eliminati. Quando un client si riconnette dopo la scadenza della sessione e imposta il flag `cleanSession` su `0`, il servizio crea una nuova sessione persistente. Gli abbonamenti o i messaggi della sessione precedente non sono disponibili per questa sessione perché sono stati scartati alla scadenza della sessione precedente.

Addebito dei messaggi di sessione persistente

I messaggi vengono addebitati all'utente Account AWS quando il broker di messaggi invia un messaggio a un cliente o a una sessione persistente offline. Quando un dispositivo offline con una sessione persistente si riconnette e riprende la sessione, i messaggi archiviati vengono recapitati al dispositivo e vengono nuovamente addebitati sul tuo account. Per ulteriori informazioni sui prezzi dei messaggi, consulta AWS IoT Core pricing - Messagging (Prezzi di Amazon IoT Core - Messaggistica).

Si può aumentare il tempo di scadenza predefinito di un'ora della sessione persistente utilizzando il processo di aumento del limite standard. Si noti che l'aumento del tempo di scadenza della sessione potrebbe aumentare gli addebiti dei messaggi. Infatti, il tempo aggiuntivo potrebbe consentire l'archiviazione di più messaggi per il dispositivo offline e tali messaggi aggiuntivi verranno addebitati all'account alla tariffa di messaggistica standard. Il tempo di scadenza della sessione è approssimativo e una sessione potrebbe persistere fino a 30 minuti più a lungo rispetto al limite dell'account; tuttavia, una sessione non può essere inferiore al limite dell'account. Per ulteriori informazioni sui limiti di sessione, consulta Service Quotas di AWS.

Messaggi conservati da MQTT

AWS IoT Core supporta il `RETAIN` flag descritto nel protocollo MQTT. Quando un client imposta il `RETAIN` flag su un messaggio MQTT che pubblica, AWS IoT Core salva il messaggio. Può quindi essere inviato a nuovi sottoscrittori, recuperato chiamando l'operazione `GetRetainedMessage` e visualizzato nella console AWS IoT.

Esempi di utilizzo di messaggi conservati da MQTT

- Come messaggio di configurazione iniziale

  I messaggi conservati MQTT vengono inviati a un client dopo che il client si è sottoscritto a un argomento. Se desiderate che tutti i client che sottoscrivono un argomento ricevano il messaggio MQTT conservato subito dopo la sottoscrizione, potete pubblicare un messaggio di configurazione con il flag impostato. `RETAIN` I client sottoscrittori ricevono anche aggiornamenti di tale configurazione ogni volta che viene pubblicato un nuovo messaggio di configurazione.
- Come ultimo messaggio di stato noto

  I dispositivi possono impostare il `RETAIN` flag sui messaggi con lo stato attuale in modo da salvarli. AWS IoT Core Quando le applicazioni si connettono o si riconnettono, possono iscriversi a questo argomento e visualizzare l'ultimo stato segnalato subito dopo la sottoscrizione all'argomento del messaggio selezionato. In questo modo possono evitare di dover attendere il prossimo messaggio dal dispositivo per vedere lo stato corrente.

In questa sezione:

- Le attività più comuni con MQTT hanno mantenuto i messaggi in AWS IoT Core
- Messaggi di fatturazione e conservati
- Confronto tra i messaggi conservati MQTT e le sessioni persistenti MQTT
- MQTT ha mantenuto i messaggi e i Device Shadows AWS IoT

Le attività più comuni con MQTT hanno mantenuto i messaggi in AWS IoT Core

AWS IoT Core salva i messaggi MQTT con il `RETAIN` flag impostato. Questi messaggi conservati vengono inviati a tutti i client che hanno sottoscritto l'argomento, come un normale messaggio MQTT, e vengono archiviati anche per essere inviati ai nuovi sottoscrittori all'argomento.

I messaggi conservati MQTT richiedono azioni di policy specifiche per autorizzare i client ad accedervi. Per esempi di utilizzo di policy dei messaggi conservati, consulta Esempi di policy per i messaggi conservati.

Questa sezione descrive le operazioni comuni che comportano messaggi conservati.

- Creazione di un messaggio conservato

  Il client determina se un messaggio viene conservato quando pubblica un messaggio MQTT. I client possono impostare il `RETAIN` flag quando pubblicano un messaggio utilizzando un Device SDK. Le applicazioni e i servizi possono impostare il `RETAIN` flag quando utilizzano l'`Publish`azione per pubblicare un messaggio MQTT.

  Viene conservato un solo messaggio per nome argomento. Un nuovo messaggio con il flag RETAIN impostato pubblicato su un argomento sostituisce qualsiasi messaggio conservato esistente inviato all'argomento in precedenza.

  > **Note**
  >
  > Non è possibile pubblicare su un argomento riservato con il `RETAIN` flag impostato.

- Iscrizione di un argomento del messaggio conservato

  I clienti si iscrivono agli argomenti dei messaggi conservati come qualsiasi altro argomento del messaggio MQTT. Il contrassegno è impostato per i messaggi conservati ricevuti mediante sottoscrizione a un argomento riservato`RETAIN`.

  I messaggi conservati vengono eliminati AWS IoT Core quando un client pubblica un messaggio conservato con un payload di messaggi da 0 byte nell'argomento del messaggio conservato. I client che hanno sottoscritto l'argomento del messaggio conservato riceveranno anche il messaggio da 0 byte.

  La sottoscrizione a un filtro argomento jolly che include un argomento del messaggio conservato consente al client di ricevere messaggi successivi pubblicati sull'argomento del messaggio conservato, ma non recapita il messaggio conservato al momento della sottoscrizione.

> **Note**
>
> Per ricevere un messaggio conservato al momento della sottoscrizione, il filtro degli argomenti nella richiesta di iscrizione deve corrispondere esattamente all'argomento del messaggio salvato.

Per i messaggi conservati ricevuti al momento della sottoscrizione a un argomento del messaggio mantenuto è impostato il flag. `RETAIN` I messaggi conservati ricevuti da un cliente abbonato dopo l'abbonamento non lo hanno.

- Recupero di un messaggio conservato

  I messaggi conservati vengono recapitati automaticamente ai client quando si sottoscrivono all'argomento con il messaggio conservato. Affinché un cliente riceva il messaggio conservato al momento dell'abbonamento, deve sottoscrivere il nome esatto dell'argomento del messaggio conservato. La sottoscrizione a un filtro argomento jolly che include un argomento del messaggio conservato consente al client di ricevere messaggi successivi pubblicati sull'argomento del messaggio conservato, ma non recapita il messaggio conservato al momento della sottoscrizione.

  I servizi e le app possono elencare e recuperare i messaggi conservati chiamando `ListRetainedMessages` e `GetRetainedMessage`.

  A un client non viene impedito di pubblicare messaggi su un argomento del messaggio mantenuto senza impostare il contrassegno. `RETAIN` Ciò potrebbe causare risultati imprevisti, ad esempio il messaggio conservato che non corrisponde al messaggio ricevuto sottoscrivendo l'argomento.

  Con MQTT 5, se in un messaggio conservato l'intervallo di scadenza del messaggio è impostato e il messaggio conservato scade, un nuovo sottoscrittore che effettua la sottoscrizione a quell'argomento non riceverà il messaggio conservato al termine della sottoscrizione.

- Elenco di argomenti di messaggio conservati

  È possibile elencare i messaggi conservati chiamando `ListRetainedMessages` e i messaggi conservati possono essere visualizzati nella console AWS IoT.

- Ricevere i dettagli del messaggio conservati

  È possibile ottenere i dettagli del messaggio conservati chiamando `GetRetainedMessage` e possono essere visualizzati nella console AWS IoT.

- Conservazione di un messaggio Will

  I messaggi Will MQTT che vengono creati quando un dispositivo si connette può essere conservato impostando flag `Will Retain` nel campo `Connect Flag bits`.

- Eliminazione di un messaggio conservato

  I dispositivi, le applicazioni e i servizi possono eliminare un messaggio conservato pubblicando un messaggio con il `RETAIN` flag impostato e un payload di messaggi vuoto (0 byte) sul nome dell'argomento del messaggio salvato da eliminare. Tali messaggi eliminano il messaggio conservato AWS IoT Core, vengono inviati ai client abbonati all'argomento, ma non vengono conservati da. AWS IoT Core

  I messaggi conservati possono anche essere eliminati in modo interattivo accedendo al messaggio conservato nella console AWS IoT. Messaggi conservati che vengono eliminati utilizzando la console AWS IoTinviano anche un messaggio a 0 byte ai client che hanno sottoscritto l'argomento del messaggio conservato.

  I messaggi conservati non possono essere ripristinati dopo che sono stati eliminati. Un client dovrebbe pubblicare un nuovo messaggio conservato per sostituire il messaggio eliminato.

- Debugging e risoluzione dei problemi dei messaggi conservati

  La console AWS IoT fornisce diversi strumenti per aiutarti a risolvere i problemi dei messaggi conservati:

  - La pagina Messaggi conservati

    La pagina Messaggi conservati nella console AWS IoT fornisce un elenco impaginato dei messaggi conservati che sono stati memorizzati dal tuo Account nella regione corrente. In questa pagina, è possibile:

    - Visualizzare i dettagli di ogni messaggio conservato, ad esempio il payload del messaggio, il QoS, l'ora in cui è stato ricevuto.
    - Aggiornare il contenuto di un messaggio conservato.
    - Eliminare un messaggio conservato.

  - Il client di test MQTT

    La pagina Client di test MQTT nella console AWS IoT può iscriversi e pubblicare argomenti MQTT. L'opzione di pubblicazione consente di impostare il flag RETAIN sui messaggi pubblicati per simulare il comportamento dei dispositivi. È inoltre possibile utilizzare il client di test MQTT

per monitorare i messaggi provenienti dai client connessi gestiti tramite l'interfaccia delle connessioni client. Per ulteriori informazioni sulla gestione delle connessioni client, vederethe section called “Gestione delle connessioni MQTT”.

Alcuni risultati imprevisti potrebbero essere il risultato di questi aspetti del modo in cui vengono implementati i messaggi conservati in AWS IoT Core.

- Limiti di messaggi conservati

  Quando un account ha archiviato il numero massimo di messaggi conservati, AWS IoT Core restituisce una risposta limitata ai messaggi pubblicati con il set RETAIN e carica più di 0 byte fino a quando alcuni messaggi conservati non vengono eliminati e il numero di messaggi conservati non scende al di sotto del limite.

- Ordine di consegna dei messaggi conservati

  La sequenza di messaggi conservati e di recapito del messaggio sottoscritto non è garantita.

Messaggi di fatturazione e conservati

La pubblicazione di messaggi con il `RETAIN` flag impostato da un client, utilizzando la AWS IoT console o chiamando `Publish`comporta costi di messaggistica aggiuntivi descritti in Pricing - Messaging.AWS IoT Core

Il recupero dei messaggi conservati da un client, utilizzando la AWS IoT console o tramite chiamata `GetRetainedMessage`comporta costi di messaggistica oltre ai normali costi di utilizzo dell'API. I costi aggiuntivi sono descritti nella sezione Prezzi di AWS IoT Core - Messaggistica.

I messaggi Will MQTT che vengono pubblicati quando un dispositivo si disconnette inaspettatamente comportano i costi di messaggistica descritti in Prezzi di AWS IoT Core - Messaggistica.

Per ulteriori informazioni sui prezzi dei messaggi, consulta la sezione Prezzi di AWS IoT Core - Messaggistica.

Confronto tra i messaggi conservati MQTT e le sessioni persistenti MQTT

I messaggi conservati e le sessioni persistenti sono caratteristiche standard di MQTT che consentono ai dispositivi di ricevere messaggi pubblicati mentre erano offline. I messaggi conservati possono essere pubblicati da sessioni persistenti. In questa sezione vengono descritti gli aspetti chiave di queste caratteristiche e come funzionano insieme.

	Messaggi conservati	Sessioni persistenti
Caratteristiche principali	I messaggi conservati possono essere utilizzati per configurare o notificare grandi gruppi di dispositivi dopo la connessione. I messaggi conservati possono essere utilizzati anche quando si desidera che i dispositivi ricevano solo l'ultimo messaggio pubblicat o su un argomento dopo una riconnessione.	Le sessioni persistenti sono utili per i dispositivi con connettività intermittente e potrebbero mancare diversi messaggi importanti. I dispositivi possono connetter si a una sessione persistente per ricevere messaggi inviati mentre sono offline.
Esempi	I messaggi conservati possono fornire informazi oni sulla configurazione dei dispositivi sul loro ambiente quando vengono online. La configurazione iniziale potrebbe includere un elenco di altri argomenti di messaggio a cui deve sottoscrivere o informazioni su come configurare il fuso orario locale.	I dispositivi che si connetton o su una rete cellulare con connettività intermittente potrebbero utilizzare sessioni persistenti per evitare di perdere messaggi importanti inviati mentre un dispositivo è fuori copertura di rete o deve spegnere la radio cellulare.
Messaggi ricevuti durante la sottoscrizione iniziale a un argomento	Dopo aver sottoscritto un argomento con un messaggio conservato, viene ricevuto il messaggio conservato più recente.	Dopo aver effettuato la sottoscrizione a un argomento senza un messaggio conservato, non viene ricevuto alcun messaggio finché non viene pubblicato sull'argo mento.

	Messaggi conservati	Sessioni persistenti
Argomenti iscritti dopo la riconnessione	Senza una sessione persistente, il client deve iscriversi agli argomenti dopo la riconnessione.	Gli argomenti sottoscritti vengono ripristinati dopo la riconnessione.
Messaggi ricevuti dopo la riconnessione	Dopo aver sottoscritto un argomento con un messaggio conservato, viene ricevuto il messaggio conservato più recente.	Tutti i messaggi pubblicati con QOS = 1 e sottoscritti con un QOS =1 mentre il dispositivo è stato disconnesso vengono inviati dopo la riconnessione del dispositivo.
Scadenza della sessione/dati	In MQTT 3, i messaggi conservati non scadono. Sono archiviati fino a quando non vengono sostituiti o eliminati. In MQTT 5, i messaggi conservati scadono dopo l'intervallo di scadenza dei messaggi impostato. Per ulteriori informazioni, consulta Scadenza dei messaggi.	Le sessioni persistenti scadono se il client non si riconnette entro il periodo di timeout. Dopo la scadenza di una sessione persistente, vengono eliminati gli abbonamenti e i messaggi salvati del client pubblicati con un QOS = 1 e sottoscritti con un QOS =1 mentre il dispositivo è stato disconnesso. I messaggi scaduti non verranno distribuiti. Per ulteriori informazioni sulle scadenze di sessione con sessioni persistenti, consulta the section called “Sessioni persistenti MQTT”.

Per informazioni sulle sessioni persistenti, consulta the section called “Sessioni persistenti MQTT”.

Con i messaggi mantenuti, il client di pubblicazione determina se un messaggio deve essere conservato e recapitato a un dispositivo dopo la connessione, indipendentemente dal fatto che

abbia avuto una sessione precedente o meno. La scelta di archiviare un messaggio viene effettuata dall'editore e il messaggio archiviato viene recapitato a tutti i client attuali e futuri che sottoscrivono abbonamenti QoS 0 o QoS 1. I messaggi conservati conservano un solo messaggio su un determinato argomento alla volta.

Quando un account ha archiviato il numero massimo di messaggi conservati, AWS IoT Core restituisce una risposta limitata ai messaggi pubblicati con impostazione RETAIN e payload superiori a 0 byte fino a quando alcuni messaggi conservati non vengono eliminati e il conteggio dei messaggi conservati scende al di sotto del limite.

MQTT ha mantenuto i messaggi e i Device Shadows AWS IoT

I messaggi conservati e i Device Shadow conservano entrambi i dati da un dispositivo, ma si comportano in modo diverso e hanno scopi diversi. In questa sezione vengono descritti somiglianze e differenze.

	Messaggi conservati	Device Shadow
Il payload del messaggio ha una struttura o uno schema predefinito	Come definito dall'impl ementazione. MQTT non specifica una struttura o uno schema per il payload dei messaggi.	AWS IoT supporta una struttura di dati specifica.
L'aggiornamento del payload del messaggio genera messaggi di evento	La pubblicazione di un messaggio conservato invia il messaggio ai client sottoscri tti, ma non genera ulteriori messaggi di aggiornamento.	L'aggiornamento di un Device Shadow produce messaggi di aggiornamento che descrivono la modifica.
Gli aggiornamenti dei messaggi sono numerati	I messaggi conservati non vengono numerati automatic amente.	I documenti Device Shadow hanno numeri di versione e timestamp automatici.
Il payload dei messaggi è collegato a una risorsa oggetto	I messaggi conservati non sono allegati a una risorsa oggetto.	I Device Shadow Device Shadow sono collegati a una risorsa oggetto.

	Messaggi conservati	Device Shadow
Aggiornamento di singoli elementi del payload del messaggio	I singoli elementi del messaggio non possono essere modificati senza aggiornare l'intero payload del messaggio.	I singoli elementi di un documento Device Shadow possono essere aggiornati senza dover aggiornare l'intero documento Device Shadow.
Il client riceve i dati dei messaggi al momento dell'sott oscrizione	Il client riceve automatic amente un messaggio conservato dopo aver sottoscri tto un argomento con un messaggio conservato.	I client possono iscriversi agli aggiornamenti Device Shadow, ma devono richieder e deliberatamente lo stato corrente.
Indicizzazione e ricercabilità	I messaggi conservati non sono indicizzati per la ricerca.	L'indicizzazione del parco istanze indicizza i dati di Device Shadow per la ricerca e l'aggregazione.

Messaggi MQTT Last Will and Testament (LWT)

Last Will and Testament (LWT) è una funzionalità di MQTT. Con LWT, i clienti possono specificare un messaggio che il broker pubblicherà su un argomento definito dal client e invierà a tutti i client che hanno sottoscritto l'argomento quando si verifica una disconnessione non iniziata. Il messaggio specificato dai client è chiamato messaggio LWT o messaggio Will e l'argomento definito dai client viene definito argomento Will. Puoi specificare un messaggio LWT quando un dispositivo si connette al broker. Questi messaggi possono essere conservati impostando il flag `Will Retain` nel campo `Connect Flag bits` durante la connessione. Ad esempio, se il flag `Will Retain` è impostato su 1, un messaggio Will verrà archiviato nel broker nell'argomento Will associato. Per ulteriori informazioni, consulta Messaggi Will.

Quando gestisci le connessioni client, puoi controllare se i messaggi LWT vengono pubblicati quando disconnetti un client. Per ulteriori informazioni, consulta the section called "Gestione delle connessioni MQTT".

Il broker memorizzerà i messaggi Will fino a quando non si verificherà una disconnessione non avviata. Quando ciò accade, il broker pubblicherà i messaggi a tutti i clienti che si sono iscritti all'argomento Will per notificare la disconnessione. Se il client si disconnette dal broker con una

disconnessione avviata dal client utilizzando il messaggio MQTT DISCONNECT, il broker non pubblicherà i messaggi LWT memorizzati. In tutti gli altri casi, i messaggi LWT verranno inviati. A causa della natura asincrona dell'elaborazione delle disconnessioni, non è garantito che i messaggi LWT vengano inviati in ordine durante la riconnessione. Si consiglia di utilizzare gli eventi del ciclo di vita per migliorare l'accuratezza del rilevamento dello stato di connettività, poiché questi eventi forniscono attributi come timestamp e numeri di versione per gestire gli eventi. out-of-order Per un elenco completo degli scenari di disconnessione in cui il broker invierà i messaggi LWT, vediEventi di connessione/disconnessione.

Utilizzo degli Attributi Connect

`ConnectAttributes` consentono di specificare quali attributi si desidera utilizzare nel messaggio di connessione delle policy IAM, ad esempio `PersistentConnect` e `LastWill`. Con `ConnectAttributes`, puoi creare criteri che non consentono ai dispositivi di accedere alle nuove funzionalità per impostazione predefinita, cosa che può essere utile in caso di compromissione di un dispositivo.

`connectAttributes` supporta le caratteristiche seguenti:

`PersistentConnect`

Utilizzo del `PersistentConnect` per salvare tutte le sottoscrizioni che il client effettua durante la connessione quando essa viene interrotta tra il client e il broker.

`LastWill`

Utilizzo del `LastWill` per pubblicare un messaggio nel `LastWillTopic` quando un client si disconnette in modo imprevisto.

Per impostazione predefinita, la policy dispone di una connessione non persistente e non esistono attributi passati per questa connessione. È necessario specificare una connessione permanente nella policy IAM se si desidera averne una.

Quando si gestiscono le connessioni client, è possibile visualizzare gli attributi di connessione e la configurazione della sessione per i client connessi. Per ulteriori informazioni, consulta the section called “Gestione delle connessioni MQTT”.

Per esempi di `ConnectAttributes`, consulta Esempi di policy di connessione.

Caratteristiche supportate da MQTT 5

AWS IoT Core il supporto per MQTT 5 si basa sulla specifica MQTT v5.0 con alcune differenze, come documentato in. the section called “AWS IoT differenze rispetto alle specifiche MQTT”

AWS IoT Core supporta le seguenti funzionalità MQTT 5:

- Abbonamenti condivisi
- Avvio pulito e scadenza della sessione
- Codice motivo su tutti ACKs
- Alias degli argomenti
- Scadenza del messaggio
- Altre caratteristiche MQTT 5

Abbonamenti condivisi

AWS IoT Core supporta sottoscrizioni condivise sia per MQTT 3.1.1 che per MQTT 5. Gli abbonamenti condivisi consentono a più client di condividere un abbonamento a un argomento e solo un client riceverà i messaggi pubblicati su quell'argomento utilizzando una distribuzione casuale. Le sottoscrizioni condivise possono bilanciare efficacemente il carico dei messaggi MQTT tra più abbonati. Ad esempio, si supponga di avere 1.000 dispositivi che pubblicano sullo stesso argomento e 10 applicazioni di back-end che elaborano tali messaggi. In tal caso, le applicazioni di backend possono sottoscrivere lo stesso argomento e ognuna riceverà in modo casuale i messaggi pubblicati dai dispositivi sull'argomento condiviso. Questo consente effettivamente di "condividere" il carico di tali messaggi. Le sottoscrizioni condivise consentono inoltre una migliore resilienza. Quando un'applicazione di back-end si disconnette, il broker distribuisce il carico agli abbonati rimanenti nel gruppo. Quando tutti gli abbonati si disconnettono, i messaggi vengono messi in coda.

Le funzionalità di accodamento dei messaggi sono disponibili per gli abbonamenti condivisi su entrambe le connessioni MQTT 3.1.1 e MQTT 5 per migliorare l'affidabilità della consegna dei messaggi.

Per utilizzare gli abbonamenti condivisi, i clienti sottoscrivono il filtro degli argomenti di un abbonamento condiviso nel modo seguente:

```
$share/{ShareName}/{TopicFilter}
```

- `$share` è una stringa letterale per indicare il filtro di argomenti di una Sottoscrizione condivisa, che deve iniziare con `$share`.
- `{ShareName}` è una stringa di caratteri per specificare il nome condiviso utilizzato da un gruppo di abbonati. Il filtro per argomenti di un abbonamento condiviso deve contenere un `ShareName` ed essere seguito dal `/` carattere. Il `{ShareName}` non deve includere i seguenti caratteri: `/`, + o#. La dimensione massima per `{ShareName}` è di 128 caratteri UTF-8.
- `{TopicFilter}`segue la stessa sintassi del filtro degli argomenti di un abbonamento non condiviso. La dimensione massima per `{TopicFilter}` è di 256 caratteri UTF-8.
- Le due barre obbligatorie (`/`) per `$share/{ShareName}/{TopicFilter}` non sono incluse nel limite Numero massimo di barre negli argomenti e nel filtro argomenti.

Le sottoscrizioni uguali `{ShareName}/{TopicFilter}` appartengono allo stesso gruppo di sottoscrizioni condiviso. Puoi creare più gruppi di abbonamenti condivisi e non superare il limite di sottoscrizioni condivise per gruppo. Per ulteriori informazioni, consulta Endpoint e quote di AWS IoT Core nella Documentazione generale di riferimento di AWS .

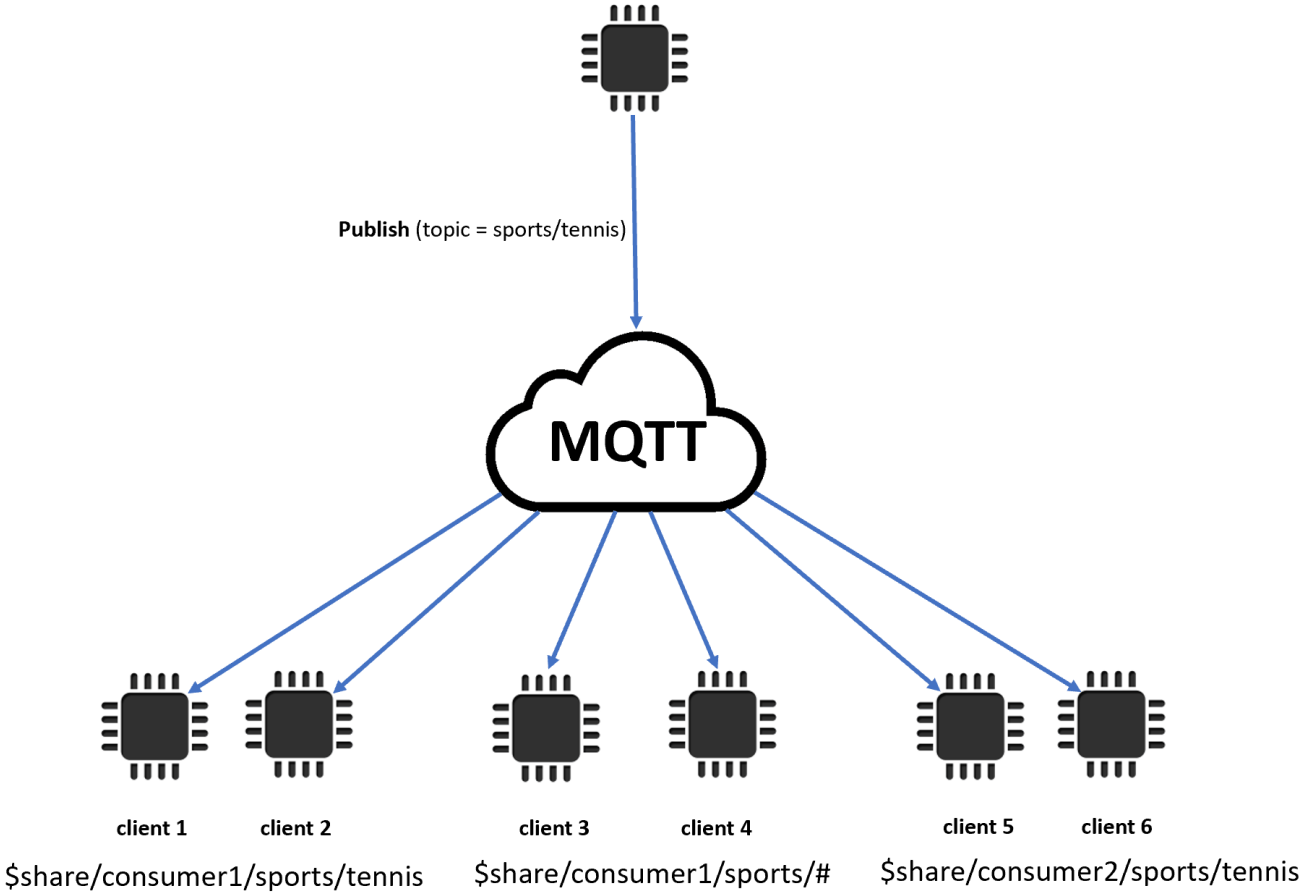
Le tabelle seguenti confrontano le sottoscrizioni non condivise e le sottoscrizioni condivise:

Subscription	Description	Esempi di filtro di argomenti
Sottoscrizioni non condivise	Ogni client crea una sottoscrizione separata per ricevere i messaggi pubblicati. Quando un messaggio viene pubblicato su un argomento, tutti gli abbonati a tale argomento ricevono una copia del messaggio.	`sports/tennis` `sports/#`
Abbonamenti condivisi	Più client possono condividere una sottoscri zione a un argomento e solo un client riceverà i messaggi pubblicati su tale argomento utilizzando una distribuzione casuale.	`$share/co nsumer/sp orts/tennis` `$share/ consumer/ sports/#`

Flusso di sottoscrizioni non condivise	Flusso di sottoscrizioni condivise		

Note importanti per l'utilizzo degli abbonamenti condivisi

- Se il gruppo di sottoscrittori condiviso è composto da sottoscrittori di sessione persistenti, quando tutti gli abbonati del gruppo condiviso sono disconnessi o se alcuni sottoscrittori violano il limite di richieste di pubblicazione al secondo per connessione, tutti i messaggi QoS 1 non riconosciuti e i messaggi QoS 1 non consegnati pubblicati su un gruppo di sottoscrizioni condiviso verranno messi in coda. Per ulteriori informazioni, consulta Accodamento dei messaggi relativi alle sottoscrizioni condivise.
- I messaggi QoS 0 pubblicati su un gruppo di sottoscrizioni condiviso verranno eliminati in caso di errore.
- Le sottoscrizioni condivise non ricevono messaggi conservati quando si sottoscrivono modelli di argomenti come parte di un gruppo di abbonati condiviso. I messaggi pubblicati su argomenti con sottoscrittori condivisi e con il `RETAIN` flag impostato vengono recapitati agli abbonati condivisi come qualsiasi altro messaggio di pubblicazione.
- Quando le sottoscrizioni condivise contengono caratteri jolly (# o +), potrebbero esserci più sottoscrizioni condivise corrispondenti a un argomento. In tal caso, il broker di messaggi copia il messaggio di pubblicazione e lo invia a un client casuale in ogni abbonamento condiviso corrispondente. Il comportamento jolly degli abbonamenti condivisi può essere spiegato nel diagramma seguente.



In questo esempio, ci sono tre sottoscrizioni condivise corrispondenti all'argomento MQTT di pubblicazione. `sports/tennis` Il broker di messaggi copia il messaggio pubblicato e lo invia a un client casuale in ciascun gruppo corrispondente.

Client 1 e client 2 condividono la sottoscrizione: `$share/consumer1/sports/tennis`

Client 3 e client 4 condividono la sottoscrizione: `$share/consumer1/sports/#`

Client 5 e client 6 condividono la sottoscrizione: `$share/consumer2/sports/tennis`

Per ulteriori informazioni sui limiti delle sottoscrizioni condivise, consulta AWS IoT Core endpoint e quote nel General Reference.AWS Per testare gli abbonamenti condivisi utilizzando il client AWS IoT MQTT nella console, vedere.AWS IoT ??? È inoltre possibile visualizzare a quali argomenti sono abbonati i client connessi, incluse le sottoscrizioni condivise, utilizzando le funzionalità di gestione della connessione client. Per ulteriori informazioni, consulta the section called “Gestione

delle connessioni MQTT". Per ulteriori informazioni sugli abbonamenti condivisi, vedere Sottoscrizioni condivise della specifica 2.0. MQTTv5

Abbonamenti condivisi, accodamento dei messaggi

Per migliorare l'affidabilità della consegna dei messaggi, gli abbonamenti condivisi includono funzionalità di accodamento dei messaggi che archiviano i messaggi quando non sono disponibili abbonati online. Quando un gruppo di sottoscrizioni condiviso contiene almeno un membro con una sessione persistente, la funzionalità di accodamento è abilitata per il gruppo. Quando si distribuiscono messaggi, i membri online vengono selezionati come destinatari. I messaggi QoS 1 vengono messi in coda quando nessun membro viene trovato online o quando gli abbonati superano il limite. `Publish requests per second per connection` I messaggi in coda vengono recapitati quando i membri esistenti riprendono le sessioni permanenti o nuovi membri si uniscono al gruppo. I messaggi in coda vengono recapitati fino a 20 messaggi in coda al secondo per abbonato attivo del gruppo, insieme a tutti gli altri messaggi recapitati all'abbonato in base agli abbonamenti.

Per impostazione predefinita, la conservazione dei messaggi in coda segue la quota. `Persistent Session expiry period` Tuttavia, se nel messaggio di pubblicazione in entrata è impostato un MEI (Message Expiry Interval), il MEI ha la precedenza. Quando è presente, il MEI determina il periodo di conservazione dei messaggi, indipendentemente dal periodo di scadenza della sessione persistente.

Le velocità di coda dei messaggi sono limitate in base alla `Queued messages per second per account`quota e il numero di messaggi è limitato dalla quota. `Maximum number of queued messages per shared subscription group` Per visualizzare e gestire le quote, accedi alla console Service Quotas.

Puoi monitorare la coda CloudWatch effettuando una ricerca nello spazio dei `ApproximateQueueDepth` `AWS/Usage` nomi oppure puoi utilizzare il seguente comando CLI per elencare le metriche associate alla profondità della coda di ogni gruppo di sottoscrizioni condiviso.

```
aws cloudwatch list-metrics --namespace "AWS/Usage" --dimensions
 Name=Resource,Value='ApproximateQueueDepth'
```

Quando questi limiti vengono superati, vengono conservati solo i messaggi che erano già in coda prima di raggiungere il limite. I nuovi messaggi in arrivo che supererebbero i limiti vengono eliminati. Il sistema non sostituisce i vecchi messaggi in coda con quelli più recenti.

L'accodamento dei messaggi viene registrato in metriche e registri. CloudWatch CloudWatch Per informazioni sulle voci scritte in CloudWatch and CloudWatch Logs, vedere e. Parametri del broker

di messaggi Voce di log Queued I messaggi in coda vengono comunque fatturati alla tariffa di messaggistica standard. Per ulteriori informazioni sui prezzi delle prenotazioni, consulta Prezzi di AWS IoT Core.

Ciclo di vita della sessione in un gruppo di sottoscrizioni condiviso

Quando una sessione pulita si iscrive a un gruppo, diventa un membro online del gruppo. Quando annulla l'iscrizione o si disconnette, la sessione pulita abbandona il gruppo.

Quando una sessione persistente si iscrive a un gruppo, diventa un membro online del gruppo. Quando si disconnette, rimane comunque nel gruppo, ma ne diventa un membro offline. Quando si riconnette, diventa nuovamente un membro online. La sessione persistente lascia il gruppo quando annulla esplicitamente l'iscrizione o quando scade dopo la disconnessione.

Avvio pulito e scadenza della sessione

È possibile utilizzare Avvio pulito e Scadenza della sessione per gestire le sessioni persistenti con maggiore flessibilità. Un flag Avvio pulito indica se la sessione deve iniziare senza utilizzare una sessione esistente. Un intervallo di scadenza della sessione indica per quanto tempo mantenere la sessione dopo una disconnessione. L'intervallo di scadenza della sessione può essere modificato al momento della disconnessione. Per ulteriori informazioni, consulta the section called “Sessioni persistenti MQTT”.

Codice motivo su tutti ACKs

È possibile eseguire il debug o elaborare i messaggi di errore più facilmente utilizzando i codici motivo. I codici motivo vengono restituiti dal broker di messaggi in base al tipo di interazione con il broker (sottoscrizione, pubblicazione, conferma). Per ulteriori informazioni, consulta Codici motivo MQTT. Per un elenco completo dei codici motivo MQTT, consulta Specifiche MQTT v5.

Alias degli argomenti

È possibile sostituire un nome argomento con un alias argomento, che è un numero intero a due byte. L'utilizzo degli alias degli argomenti può ottimizzare la trasmissione dei nomi degli argomenti per ridurre potenzialmente i costi dei dati sui servizi di dati misurati. AWS IoT Core ha un limite predefinito di 8 alias tematici. Per ulteriori informazioni, consulta Endpoint e quote di AWS IoT Core nella Documentazione generale di riferimento di AWS .

Scadenza del messaggio

È possibile aggiungere valori di scadenza del messaggio ai messaggi pubblicati. Questi valori rappresentano l'intervallo di scadenza dei messaggi (MEI) in secondi. Se i messaggi non sono stati

inviati ai sottoscrittori entro tale intervallo, il messaggio scadrà e verrà rimosso. Se non si imposta il valore di scadenza del messaggio, il messaggio non scadrà.

In uscita, il sottoscrittore riceverà un messaggio con il tempo rimanente nell'intervallo di scadenza. Ad esempio, se per un messaggio di pubblicazione in entrata è impostata una scadenza del messaggio di 30 secondi e il messaggio viene instradato al sottoscrittore dopo 20 secondi, il campo di scadenza del messaggio verrà aggiornato a 10. È possibile che il messaggio ricevuto dal sottoscrittore abbia un MEI aggiornato pari a 0. Questo perché non appena il tempo rimanente è pari o inferiore a 999 ms, verrà aggiornato a 0.

In AWS IoT Core, il MEI minimo è 1. Se l'intervallo è impostato su 0 dal lato client, verrà modificato su 1. L'intervallo massimo di scadenza del messaggio è 604800 (7 giorni). Qualsiasi valore superiore a questo verrà modificato nel valore massimo.

Nella comunicazione tra versioni, il comportamento della scadenza del messaggio viene deciso dalla versione MQTT del messaggio di pubblicazione in entrata. Ad esempio, un messaggio con scadenza inviato da una sessione connessa tramite MQTT5 può scadere per i dispositivi abbonati alle sessioni. MQTT3 Nella tabella seguente viene elencato in che modo la scadenza del messaggio supporta i seguenti tipi di messaggi di pubblicazione:

Tipo di messaggio di pubblicazione	Message Expiry Interval
Pubblicazione regolare	Se un server non riesce a recapitare il messaggio entro il tempo specificato, il messaggio scaduto verrà rimosso e non verrà ricevuto dal sottoscrittore. Ciò include situazioni come quando un dispositivo non effettua il back-up dei messaggi QoS 1.
Mantenimento	Se un messaggio conservato scade e un nuovo client effettua la sottoscrizione all'argomento, il client non riceverà il messaggio al momento della sottoscri zione.
Conclusivo	L'intervallo per i messaggi conclusivi inizia dopo che il client si disconnette e il server tenta di distribuire il messaggio conclusivo ai suoi sottoscrittori.
Messaggi in coda	Se un QoS 1 in uscita con intervallo di scadenza dei messaggi scade quando un client è offline, dopo la ripresa della sessione persistente, il client non riceverà il messaggio scaduto.

Altre caratteristiche MQTT 5

Disconnessione server

Quando si verifica una disconnessione, il server può inviare in modo proattivo al client un messaggio DISCONNECT per segnalare la chiusura della connessione con un codice motivo della disconnessione.

Richiesta/Risposta

Gli autori possono richiedere che il destinatario invii una risposta a un argomento specificato dall'autore al momento della ricezione.

Dimensione massima del pacchetto

Client e server possono specificare in modo indipendente la dimensione massima dei pacchetti supportati.

Formato payload e tipo di contenuto

È possibile specificare il formato payload (binario, testo) e il tipo di contenuto quando viene pubblicato un messaggio. Questi dati vengono inoltrati al destinatario del messaggio.

Proprietà MQTT 5

Le proprietà MQTT 5 sono importanti aggiunte allo standard MQTT per supportare nuove funzionalità MQTT 5 come Session Expiry e il pattern. Request/Response In AWS IoT Core, è possibile creare regole in grado di inoltrare le proprietà nei messaggi in uscita o utilizzare HTTP Publish per pubblicare messaggi MQTT con alcune delle nuove proprietà.

Nella tabella seguente sono elencate tutte le proprietà MQTT 5 supportate. AWS IoT Core

Proprietà	Description	Tipo di input	Pacchetto
Payload Format Indicator	Un valore booleano che indica se il payload è formattato come UTF-8.	Byte	PUBLISH, CONNECT
Content Type	Una stringa UTF-8 che descrive il contenuto del payload.	Stringa UTF-8	PUBLISH, CONNECT

Proprietà	Description	Tipo di input	Pacchetto
Response Topic	Una stringa UTF-8 che descrive l'argomento su cui il destinatario deve pubblicare come parte del flusso di richiesta-risposta. L'argomento non deve contenere caratteri jolly.	Stringa UTF-8	PUBLISH, CONNECT
Correlation Data	Dati binari utilizzati dal mittente del messaggio di richiesta per identificare a quale richiesta è destinato il messaggio di risposta.	Binario	PUBLISH, CONNECT
User Property	Una coppia di stringhe UTF-8. Questa proprietà può essere visualizzata più volte in un pacchetto. I destinatari riceveranno le coppie chiave-valore nello stesso ordine in cui vengono inviate.	Una coppia di stringhe UTF-8	CONNECT, PUBLISH, Will Properties, SUBSCRIBE, DISCONNECT, UNSUBSCRIBE
Message Expiry Interval	Un numero intero a 4 byte che rappresenta l'intervallo di scadenza del messaggio in secondi. Se assente, il messaggio non scade.	Numero intero a 4 byte	PUBLISH, CONNECT
Session Expiry Interval	Un numero intero a 4 byte che rappresenta l'intervallo di scadenza della sessione in secondi. AWS IoT Core supporta un massimo di 7 giorni, con un massimo predefinito di un'ora. Se il valore impostato supera il valore massimo del tuo account, AWS IoT Core restituirà il valore corretto nel CONNACK.	Numero intero a 4 byte	CONNECT, CONNACK, DISCONNECT

Proprietà	Description	Tipo di input	Pacchetto
Assigned Client Identifier	Un ID client casuale generato da AWS IoT Core quando un ID client non è specificato dai dispositivi. L'ID cliente casuale deve essere un nuovo identificatore del cliente che non viene utilizzato da nessun'altra sessione attualmente gestita dal broker.	Stringa UTF-8	CONNACK
Server Keep Alive	Un numero intero a 2 byte che rappresenta il tempo keep alive assegnato dal server. Il server eseguirà la disconnessione del client se il client è inattivo per un periodo superiore al tempo keep alive.	Numero intero a 2 byte	CONNACK
Request Problem Information	Un valore booleano che indica se la Reason String o User Properties vengono inviate in caso di errori.	Byte	CONNECT
Receive Maximum	Un numero intero a 2 byte che rappresenta il numero massimo di pacchetti PUBLISH QOS > 0 che possono essere inviati senza ricevere un PUBACK.	Numero intero a 2 byte	CONNECT, CONNACK
Topic Alias Maximum	Questo valore indica il valore massimo che verrà accettato come un alias argomento. Il valore predefinito è 0.	Numero intero a 2 byte	CONNECT, CONNACK
Maximum QoS	Il valore massimo di QoS supportato. AWS IoT Core L'impostazione predefinita è 1. AWS IoT Core non supporta QoS 2.	Byte	CONNACK
Retain Available	Un valore booleano che indica se il broker di messaggi supporta i AWS IoT Core messaggi conservati. Il valore di default è 1.	Byte	CONNACK

Proprietà	Description	Tipo di input	Pacchetto
Maximum Packet Size	La dimensione massima del pacchetto che AWS IoT Core accetta e invia. Non può superare i 128 KB.	Numer intero a 4 byte	CONNECT, CONNACK
Wildcard Subscription Available	Un valore booleano che indica se il broker di AWS IoT Core messaggi supporta Wildcard Subscription Available. Il valore di default è 1.	Byte	CONNACK
Subscript ion Identifier Available	Un valore booleano che indica se il broker di AWS IoT Core messaggi supporta Subscript ion Identifier Available. Il valore predefinito è 0.	Byte	CONNACK

Codici motivo MQTT

MQTT 5 introduce una migliore segnalazione degli errori con risposte al codice motivo. AWS IoT Core può restituire i codici causali inclusi, a titolo esemplificativo ma non esaustivo, i seguenti raggruppati per pacchetti. Per un elenco completo dei codici motivo supportati da MQTT 5, consulta Specifiche MQTT v5.

Codici motivo CONNACK

Valore	Hex	Nome del codice motivo	Description
0	0x00	Completato	La connessione è accettata.
128	0x80	Unspecified error (Errore non specificato)	Il server non desidera rivelare il motivo dell'errore o nessuno degli altri codici motivo è valido.
133	0x85	Client Identifier not valid (Identifi	L'identificatore client è una stringa valida ma non è consentito dal server.

Valore	Hex	Nome del codice motivo	Description
		catore client non valido)	
134	0x86	Bad User Name or Password (Nome utente o password errati)	Il server non accetta il nome utente o la password specifica ti dal client.
135	0x87	Not authorized (Non autorizzato)	Il client non è autorizzato a connettersi.
144	0x90	Topic Name invalid (Nome argomento non valido)	Il formato del nome argomento conclusivo è corretto ma non è accettato dal server.
151	0x97	Quota superata	È stato superato un limite imposto di implementazione o amministrativo.
155	0x9B	QoS not supported (QoS non supportato)	Il server non supporta il QoS impostato in Will QoS.

Codici motivo PUBACK

Valore	Hex	Nome del codice motivo	Description
0	0x00	Completato	Il messaggio è accettato. La pubblicazione del messaggio QoS 1 avanza.
128	0x80	Unspecified error (Errore non specifica to)	Il ricevitore non accetta la pubblicazione, ma non vuole rivelare il motivo o questo non corrisponde a uno degli altri valori.

Valore	Hex	Nome del codice motivo	Description
135	0x87	Not authorized (Non autorizzato)	Il PUBLISH non è autorizzato.
144	0x90	Topic Name invalid (Nome argomento non valido)	Il nome dell'argomento non è malformato, ma non è accettato dal client o dal server.
145	0x91	Packet identifier in use (Identificatore pacchetto in uso)	L'identificatore pacchetto è già in uso. Ciò potrebbe indicare una mancata corrispondenza nello stato della sessione tra il client e il server.
151	0x97	Quota superata	È stato superato un limite imposto di implementazione o amministrativo.

Codici motivo DISCONNECT

Valore	Hex	Nome del codice motivo	Description
129	0x81	Malformed Packet (Pacchetto non conforme)	Il pacchetto ricevuto non è conforme a questa specifica.
130	0x82	Protocol Error (Errore di protocollo)	È stato ricevuto un pacchetto imprevisto o non in ordine.
135	0x87	Not authorized (Non autorizzato)	La richiesta non è autorizzata.

Valore	Hex	Nome del codice motivo	Description
139	0x8B	Server shutting down (Arresto del server)	Il server è in fase di arresto.
141	0x8D	Keep Alive timeout (Timeout Keep Alive)	La connessione è stata chiusa perché non è stato ricevuto alcun pacchetto per 1,5 volte il tempo Keep Alive.
142	0x8E	Session taken over (Sessione acquisita)	Un'altra connessione che utilizza lo stesso ClientID è stata connessa, causando la chiusura di questa connessione.
143	0x8F	Topic Filter invalid (Filtro argomenti non valido)	Il formato del filtro argomenti è corretto ma non è accettato dal server.
144	0x90	Topic Name invalid (Nome argomento non valido)	Il formato del nome argomento è corretto ma non è accettato da questo client o dal server.
147	0x93	Receive Maximum exceeded (Ricezion e massima superata)	Il client o il server hanno ricevuto più della pubblicazione Receive Maximum (Ricezione massima) per la quale non hanno inviato PUBACK o PUBCOMP.
148	0x94	Topic Alias invalid (Alias argomento non valido)	Il client o il server hanno ricevuto un pacchetto PUBLISH contenente un alias argomento maggiore dell'alia s argomento massimo che ha inviato nel pacchetto CONNECT o CONNACK.

Valore	Hex	Nome del codice motivo	Description
151	0x97	Quota superata	È stato superato un limite imposto di implementazione o amministrativo.
152	0x98	Administrative action (Operazione amministrativa)	La connessione viene chiusa a causa di un'operazione amministrativa.
155	0x9B	QoS not supported (QoS non supportato)	Il client ha specificato un QoS maggiore del QoS specificato in un QoS massimo in CONNACK.
161	0xA1	Subscription Identifiers not supported (Identificatori sottoscrizione non supportati)	Il server non supporta gli identificatori sottoscrizione; la sottoscrizione non è accettata.

Codici motivo SUBACK

Valore	Hex	Nome del codice motivo	Description
0	0x00	Granted QoS 0 (QoS concesso 0)	La sottoscrizione è accettata e il QoS massimo inviato sarà QoS 0. Potrebbe essere un QoS inferiore a quello richiesto.
1	0x01	Granted QoS 1 (QoS concesso 1)	La sottoscrizione è accettata e il QoS massimo inviato sarà QoS 1. Potrebbe essere un QoS inferiore a quello richiesto.

Valore	Hex	Nome del codice motivo	Description
128	0x80	Unspecified error (Errore non specificato)	La sottoscrizione non è accettata e il server non desidera rivelare il motivo o nessuno degli altri codici motivo è valido.
135	0x87	Not authorized (Non autorizzato)	Il client non è autorizzato a effettuare questa sottoscrizione.
143	0x8F	Topic Filter invalid (Filtro argomenti non valido)	Il formato del filtro argomenti è corretto ma non è consentito per questo client.
145	0x91	Packet Identifier in use (Identificatore pacchetto in uso)	L'identificatore pacchetto specificato è già in uso.
151	0x97	Quota superata	È stato superato un limite imposto di implementazione o amministrativo.

Codici motivo UNSUBACK

Valore	Hex	Nome del codice motivo	Description
0	0x00	Completato	La sottoscrizione viene eliminata.
128	0x80	Unspecified error (Errore non specificato)	L'annullamento della sottoscrizione non può essere completata e il server non desidera rivelare il motivo o nessuno degli altri codici motivo è valido.
143	0x8F	Topic Filter invalid (Filtro	Il formato del filtro argomenti è corretto ma non è consentito per questo client.

Valore	Hex	Nome del codice motivo	Description
		argomenti non valido)	
145	0x91	Packet Identifier in use (Identificatore pacchetto in uso)	L'identificatore pacchetto specificato è già in uso.

AWS IoT differenze rispetto alle specifiche MQTT

L'implementazione del broker di messaggi è basata sulla specifica MQTT v3.1.1 e la specifica MQTT v5.0 ma è diversa dalle specifiche per quanto riguarda le considerazioni seguenti:

- AWS IoT non supporta i seguenti pacchetti per MQTT 3: PUBREC, PUBREL e PUBCOMP.
- AWS IoT non supporta i seguenti pacchetti per MQTT 5: PUBREC, PUBREL, PUBCOMP e AUTH.
- AWS IoT non supporta il reindirizzamento del server MQTT 5.
- AWS IoT supporta solo i livelli di qualità del servizio (QoS) MQTT 0 e 1. AWS IoT non supporta la pubblicazione o l'abbonamento con QoS di livello 2. Il broker di messaggi non invia un messaggio PUBACK o SUBACK quando è richiesto un livello QoS 2.
- In AWS IoT, la sottoscrizione a un argomento con livello QoS 0 significa che un messaggio viene recapitato zero o più volte. È possibile che un messaggio venga recapitato più di una volta. I messaggi recapitati più di una volta potrebbero essere inviati con un ID pacchetto diverso. In questi casi, il flag DUP non è impostato.
- Nel rispondere a una richiesta di connessione, il broker di messaggi invia un messaggio CONNACK. Questo messaggio contiene un flag per indicare se la connessione sta riprendendo una sessione precedente.
- Prima di inviare pacchetti di controllo aggiuntivi o una richiesta di disconnessione, il client deve attendere che broker di messaggi di AWS IoT riceva il messaggio CONNACK.
- Quando un client sottoscrive un argomento, può verificarsi un ritardo tra il momento in cui il broker di messaggi invia un messaggio SUBACK e il momento in cui il client inizia a ricevere nuovi messaggi corrispondenti.
- Quando un client utilizza il carattere jolly # nel filtro argomento per sottoscrivere un argomento, tutte le stringhe al livello e al di sotto del livello nella gerarchia degli argomenti vengono

abbinate. Tuttavia, l'argomento principale non è corrispondente. Ad esempio, una sottoscrizione all'argomento `sensor/#` riceve messaggi pubblicati sugli argomenti `sensor/`, `sensor/temperature`, `sensor/temperature/room1`, ma non messaggi pubblicati su `sensor`. Per ulteriori informazioni sull'utilizzo di caratteri jolly, vedi Filtri per i nomi degli argomenti.

- Il broker di messaggi usa l'ID client per identificare ogni client. L'ID client viene passato dal client al broker di messaggi come parte del payload MQTT. Due client con lo stesso ID client non possono essere connessi contemporaneamente al broker di messaggi. Quando un client si connette al broker di messaggi tramite un ID client usato da un altro client, viene accettata la connessione del nuovo client e il client connesso in precedenza viene disconnesso. È inoltre possibile disconnettere manualmente i client utilizzando. APIs Per ulteriori informazioni, consulta the section called “Gestione delle connessioni MQTT”.
- In rari casi, il broker di messaggi può inviare di nuovo lo stesso messaggio PUBLISH logico con un ID pacchetto diverso.
- La sottoscrizione ai filtri degli argomenti che contengono un carattere jolly non può ricevere messaggi conservati. Per ricevere un messaggio conservato, la richiesta di sottoscrizione deve contenere un filtro argomento che corrisponda esattamente all'argomento del messaggio conservato.
- Il broker di messaggi non garantisce l'ordine di ricezione dei messaggi e delle conferme.
- AWS IoT può avere limiti diversi dalle specifiche. Per ulteriori informazioni, consulta quote e limiti del broker di messaggistica e del protocollo AWS IoT Core nella Guida di riferimento di AWS IoT .
- Il flag MQTT DUP non è supportato.

Gestione delle connessioni MQTT

AWS IoT Core fornisce APIs assistenza nella gestione delle connessioni MQTT, inclusa la possibilità di disconnettere i client e gestirne le sessioni. Queste funzionalità offrono un maggiore controllo sulla flotta di AWS IoT clienti e aiutano a risolvere i problemi di connessione.

DeleteConnection API

Utilizzate l'`DeleteConnection`API per disconnettere i dispositivi MQTT AWS IoT Core specificando il relativo client. IDs Quando disconnetti un client, lo AWS IoT Core disconnette dal broker di messaggi e, facoltativamente, puoi pulire lo stato della sessione e sopprimere il AWS IoT Core messaggio Last Will and Testament (LWT).

Quando chiami l'`DeleteConnection`API, AWS IoT Core esegue diverse azioni per garantire una disconnessione pulita. AWS IoT Core invia innanzitutto un messaggio di disconnessione MQTT al client per terminare la sessione MQTT. Il servizio chiude quindi il socket sottostante. TCP/TLS

Il broker di messaggi invia un `DISCONNECT` pacchetto al dispositivo e pubblica un evento del ciclo di vita con il motivo della disconnessione. `API_INITIATED_DISCONNECT` Ciò consente di identificare quando una disconnessione è stata avviata tramite l'API anziché dal client o a causa di problemi di rete. È possibile monitorare questi eventi per scopi di visibilità, risoluzione dei problemi e controllo. Ad esempio, è possibile utilizzare AWS IoT regole per elaborare questi eventi per tenere traccia di quando e perché i client sono stati disconnessi.

Se si imposta il `cleanSession` parametro su`true`, AWS IoT Core rimuove lo stato della sessione del client, inclusi tutti gli abbonamenti e i messaggi in coda. Se si pulisce una sessione, la sessione persistente viene interrotta. Se il client era una sessione persistente e il `preventWillMessage` parametro è impostato su`false`, il servizio invia il messaggio LWT, se disponibile, utile durante le operazioni di manutenzione pianificate.

Quando si chiama l'`DeleteConnection`API, il processo di disconnessione inizia immediatamente, ma il momento esatto in cui il client riconosce la disconnessione può variare in base alle condizioni della rete e all'implementazione del client. La maggior parte dei client rileva la disconnessione entro pochi secondi, ma in alcuni casi con una connettività di rete scadente, il client potrebbe impiegare più tempo a riconoscere che è stata disconnessa.

> **Note**
>
> Forzando la disconnessione, un client deve autenticarsi nuovamente e autorizzarsi nuovamente con un nuovo stato di sessione. La chiamata API di per sé non impedisce la riconnessione dei client. Se lo si desidera, è inoltre necessario modificare le credenziali o la politica di un client prima di emettere la `DeleteConnection` chiamata API.

Per informazioni sui prezzi, consultare Prezzi di AWS IoT Core.

Casi d’uso

L'`DeleteConnection`API è utile per gestire i client che si comportano male e che presentano comportamenti problematici o consumano risorse eccessive. Forzando la disconnessione, puoi assicurarti che i client ristabiliscano le connessioni con l'autenticazione e l'autorizzazione corrette, il che può aiutare a risolvere i problemi di consumo di risorse.

Anche gli scenari di reindirizzamento dei client traggono vantaggio da questa API. Quando devi reindirizzare i client su diversi endpoint oppure Regioni AWS puoi disconnetterli a livello di codice e farli ricollegare a un altro endpoint modificando le impostazioni DNS. AWS IoT Core Questa API può aiutare a risolvere connessioni bloccate o eliminare stati di sessione problematici che potrebbero impedire il normale funzionamento.

Parametri dell'API

L'`DeleteConnection`API accetta i seguenti parametri:

ClientID (obbligatorio)

L'identificatore univoco del client MQTT da disconnettere. Questo è specificato nel percorso URL. L'ID client non può iniziare con il simbolo del dollaro ($).

> **Note**
>
> Il client MQTT IDs può contenere caratteri non validi nelle richieste HTTP. Quando si utilizza l'`DeleteConnection`API, è necessario codificare l'URL (codifica in percentuale) tutti i caratteri dell'ID client validi in MQTT ma non in HTTP. Ciò include caratteri speciali come spazi, barre (/) e caratteri UTF-8. Ad esempio, uno spazio diventa %20, una barra in avanti diventa %2F e il carattere UTF-8 ü diventa %C 3% BC. La codifica corretta garantisce che il client IDs MQTT venga trasmesso correttamente nella chiamata API basata su HTTP.

CleanSession (opzionale)

Specifica se rimuovere lo stato della sessione del client durante la disconnessione. Imposta su per `true` eliminare tutte le informazioni sulla sessione, incluse le sottoscrizioni e i messaggi in coda. Impostato per `false` preservare lo stato della sessione. Per impostazione predefinita, è impostato su `false` (mantiene lo stato della sessione). Per le sessioni pulite questo parametro verrà ignorato.

preventWillMessage (facoltativo)

Controlla se AWS IoT Core invia il messaggio Last Will and Testament (LWT), se disponibile, al momento della disconnessione. Impostato per impedire l'invio del `true` messaggio LWT. Impostato per consentire l'`false`invio. Per impostazione predefinita, è impostato su `false` (invia LWT se disponibile).

Sintassi dell'API

L'`DeleteConnection`API utilizza il seguente formato di richiesta HTTP:

```
DELETE /connections/<clientId>?
cleanSession=<cleanSession>&preventWillMessage=<preventWillMessage> HTTP/1.1
```

Richieste di esempio:

```
// Basic disconnect (preserves session, allows LWT message)
DELETE /connections/myDevice123 HTTP/1.1

// Disconnect and clear session
DELETE /connections/myDevice123?cleanSession=TRUE HTTP/1.1

// Disconnect, clear session, and prevent LWT message
DELETE /connections/myDevice123?cleanSession=TRUE&preventWillMessage=TRUE HTTP/1.1
```

Le richieste riuscite restituiscono HTTP 200 OK senza corpo di risposta.

> **Note**
>
> Il nome del servizio utilizzato da AWS Signature Version 4 per firmare le richieste è: iotdevicegateway. Puoi trovare il tuo endpoint usando il comando. `aws iot describe-endpoint --endpoint-type iot:Data-ATS`

Autorizzazioni richieste

Per utilizzare l'`DeleteConnection`API, è necessaria la seguente autorizzazione IAM:

```
iot:DeleteConnection
```

Puoi estendere questa autorizzazione a un cliente specifico IDs utilizzando politiche basate sulle risorse. Esempio:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": "iot:DeleteConnection",
            "Resource": "arn:aws:iot:region:account:client/myDevice*"
        }
    ]
}
```

Considerazioni importanti

I client disconnessi possono tentare immediatamente di riconnettersi a meno che non sia stata implementata una logica aggiuntiva per impedire la riconnessione. L'operazione di disconnessione termina solo la connessione corrente e non impedisce la riconnessione di una connessione. Se devi impedire la riconnessione, prendi in considerazione l'implementazione della logica lato client o la disabilitazione delle credenziali di un dispositivo.

I limiti di velocità si applicano all'API come parte della limitazione di velocità standard dell'API. AWS IoT Core Quando pianifichi operazioni di disconnessione in blocco, assicurati di tenere conto di questi limiti e di implementare una logica di ripetizione dei tentativi e strategie di batch appropriate per evitare limitazioni. Per ulteriori informazioni, consulta Endpoint e quote per AWS IoT Core.

Risposte agli errori

L'`DeleteConnection`API può restituire le seguenti risposte di errore:

InvalidRequestException

La richiesta non è valida. Ciò può verificarsi se il formato dell'ID client non è valido, contiene un prefisso con il simbolo del dollaro ($) o se mancano i parametri obbligatori.

ResourceNotFoundException

L'ID client specificato non esiste o non è attualmente connesso e non dispone di una sessione persistente.

UnauthorizedException

Non hai le autorizzazioni necessarie per eseguire questa operazione. Assicurati di disporre dell'`iot:DeleteConnection`autorizzazione richiesta.

ForbiddenException

Il chiamante non è autorizzato a effettuare la richiesta. Ciò potrebbe verificarsi a causa di autorizzazioni IAM insufficienti o di restrizioni politiche basate sulle risorse.

ThrottlingException

La velocità supera il limite. Riduci la frequenza delle chiamate API e implementa una logica di riprova appropriata con backoff esponenziale.

InternalFailureException

Si è verificato un errore imprevisto. Riprova la richiesta dopo un breve ritardo.

ServiceUnavailableException

Il servizio è temporaneamente non disponibile. Riprova la richiesta dopo un breve ritardo.

Pubblicazione HTTPS

I client possono pubblicare messaggi effettuando richieste all'API REST utilizzando i protocolli HTTP 1.0 o 1.1. Per l'autenticazione e i mapping delle porte utilizzati dalle richieste HTTP, consulta ???.

> **Note**
>
> A differenza di MQTT, HTTPS non supporta un valore `clientId`. `clientId` è disponibile quando si utilizza MQTT, ma non è disponibile quando si utilizza HTTPS.

URL del messaggio HTTPS

I dispositivi e i client pubblicano i loro messaggi attraverso richieste POST a un endpoint specifico del client e a un URL specifico dell'argomento:

```
https://IoT_data_endpoint/topics/url_encoded_topic_name?qos=1
```

- *`IoT_data_endpoint`* è l'endpoint dei dati AWS IoT del dispositivo. È possibile trovare l'endpoint nella AWS IoT console nella pagina dei dettagli dell'oggetto o sul client utilizzando il AWS CLI comando:

  ```
  aws iot describe-endpoint --endpoint-type iot:Data-ATS
  ```

  L'endpoint dovrebbe assomigliare a questo: `a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`

- *url_encoded_topic_name* è il nome completo dell'argomento del messaggio inviato.

Esempi di codice di messaggio HTTPS

Questi sono alcuni esempi di come inviare un messaggio HTTPS a AWS IoT.

Python (port 8443)

```
import requests
import argparse

# define command-line parameters
parser = argparse.ArgumentParser(description="Send messages through an HTTPS
 connection.")
parser.add_argument('--endpoint', required=True, help="Your AWS IoT data custom
 endpoint, not including a port. " +
                                                       "Ex: \"abcdEXAMPLExyz-
ats.iot.us-east-1.amazonaws.com\"")
parser.add_argument('--cert', required=True, help="File path to your client
 certificate, in PEM format.")
parser.add_argument('--key', required=True, help="File path to your private key, in
 PEM format.")
parser.add_argument('--topic', required=True, default="test/topic", help="Topic to
 publish messages to.")
parser.add_argument('--message', default="Hello World!", help="Message to publish. "
 +
                                                       "Specify empty string to
 publish nothing.")

# parse and load command-line parameter values
args = parser.parse_args()

# create and format values for HTTPS request
publish_url = 'https://' + args.endpoint + ':8443/topics/' + args.topic + '?qos=1'
publish_msg = args.message.encode('utf-8')

# make request
publish = requests.request('POST',
            publish_url,
            data=publish_msg,
            cert=[args.cert, args.key])

# print results
```

```
print("Response status: ", str(publish.status_code))
if publish.status_code == 200:
        print("Response body:", publish.text)
```

Python (port 443)

```
import requests
import http.client
import json
import ssl

ssl_context = ssl.SSLContext(protocol=ssl.PROTOCOL_TLS_CLIENT)
ssl_context.minimum_version = ssl.TLSVersion.TLSv1_2

# note the use of ALPN
ssl_context.set_alpn_protocols(["x-amzn-http-ca"])
ssl_context.load_verify_locations(cafile="./<root_certificate>")

# update the certificate and the AWS endpoint
ssl_context.load_cert_chain("./<certificate_in_PEM_Format>",
 "<private_key_in_PEM_format>")
connection = http.client.HTTPSConnection('<the ats IoT endpoint>', 443,
 context=ssl_context)
message = {'data': 'Hello, I'm using TLS Client authentication!'}
json_data = json.dumps(message)
connection.request('POST', '/topics/device%2Fmessage?qos=1', json_data)

# make request
response = connection.getresponse()

# print results
print(response.read().decode())
```

CURL

Per utilizzare curl per inviare un messaggio da un client o da un dispositivo a AWS IoT.

Per usare curl per inviare un messaggio da un dispositivo AWS IoT client

1. Controlla la versione curl.

   a. Sul client, eseguire questo comando al prompt dei comandi.

```
curl --help
```

Nel testo della guida, cercare le opzioni TLS. Viene visualizzata l'opzione `--tlsv1.2`.

b. Se viene visualizzata l'opzione `--tlsv1.2`, continuare.

c. Se non visualizzi l'opzione `--tlsv1.2` o ricevi un errore `command not found`, aggiorna o installa curl sul client o installa `openssl` prima di continuare.

2. Installare i certificati sul client.

Copia i file di certificato che hai creato quando hai registrato il tuo client (cosa) nella AWS IoT console. Assicurarsi di disporre di questi tre file di certificato sul client prima di continuare.

- Il file del certificato CA (*`Amazon-root-CA-1.pem`* in questo esempio).
- File di certificato del client (*`device.pem.crt`* in questo esempio).
- Il file della chiave privata del client (*`private.pem.key`* in questo esempio).

3. Crea la riga di comando curl, sostituendo i valori sostituibili per quelli dell'account e del sistema.

```
curl --tlsv1.2 \
    --cacert Amazon-root-CA-1.pem \
    --cert device.pem.crt \
    --key private.pem.key \
    --request POST \
    --data "{ \"message\": \"Hello, world\" }" \
    "https://IoT_data_endpoint:8443/topics/topic?qos=1"
```

--tlsv1.2

Utilizza TLS 1.2 (SSL).

--cacert *`Amazon-root-CA-1.pem`*

Il nome del file e il percorso, se necessario, del certificato CA per verificare il peer.

--certificato *`device.pem.crt`*

Il nome e il percorso del file del certificato del client, se necessario.

--chiave *`private.pem.key`*

Il nome e il percorso del file della chiave privata del client, se necessario.

--request POST

Il tipo di richiesta HTTP (in questo caso POST).

--dati "» *{ \"message\": \"Hello, world\" }*

Dati HTTP POST che vuoi pubblicare. In questo caso, è una stringa JSON, con le virgolette interne precedute da un carattere di escape barra rovesciata (\).

«https: *IoT_data_endpoint* //8443/topics/? *topic* qos=1"

L'URL dell'endpoint di dati del AWS IoT dispositivo del client, seguito dalla porta HTTPS:8443, seguita dalla parola chiave `/topics/` e dal nome dell'argomento, in questo caso. `topic` Specifica la qualità del servizio come parametro di query, `?qos=1`.

4. Apri il client di test MQTT nella AWS IoT console.

   Segui le istruzioni Visualizza i messaggi MQTT con il AWS IoT client MQTT e configura la console per sottoscrivere i messaggi con il nome dell'argomento *topic* usato nel curl comando oppure usa il filtro wildcard per gli argomenti di. #

5. Testare il comando.

   Durante il monitoraggio dell'argomento nel client di test della console AWS IoT , andare al client ed emettere la riga di comando curl creata nel passaggio 3. Nella console vengono visualizzati i messaggi del client.

Argomenti MQTT

Gli argomenti MQTT identificano i AWS IoT messaggi. AWS IoT i client identificano i messaggi che pubblicano assegnando ai messaggi i nomi degli argomenti. I client identificano i messaggi a cui si desidera effettuare la sottoscrizione (ricevere) registrando un filtro argomento con AWS IoT Core. Il broker di messaggi usa nomi e filtri argomento per instradare messaggi dai client di pubblicazione ai client di sottoscrizione.

Il broker di messaggi utilizza argomenti per identificare messaggi inviati tramite MQTT e inviati tramite HTTP a URL del messaggio HTTPS.

Sebbene AWS IoT supporti alcuni argomenti di sistema riservati, la maggior parte degli argomenti MQTT viene creata e gestita dall'utente, il progettista del sistema. AWS IoT utilizza gli argomenti per identificare i messaggi ricevuti dai client di pubblicazione e selezionare i messaggi da inviare ai client abbonati, come descritto nelle sezioni seguenti. Prima di creare uno spazio dei nomi degli argomenti

per il sistema, esaminare le caratteristiche degli argomenti MQTT per creare la gerarchia dei nomi degli argomenti più adatta al sistema IoT.

Nomi argomento

I nomi degli argomenti e i filtri degli argomenti sono stringhe codificate in UTF-8. Possono rappresentare una gerarchia di informazioni utilizzando il carattere barra (/) per separare i livelli della gerarchia. Ad esempio, il nome di questo argomento potrebbe riferirsi a un sensore di temperatura nella stanza 1:

- `sensor/temperature/room1`

In questo esempio, potrebbero esserci altri tipi di sensori in altre stanze con nomi di argomenti come:

- `sensor/temperature/room2`
- `sensor/humidity/room1`
- `sensor/humidity/room2`

> **Note**
>
> Considerando i nomi degli argomenti per i messaggi nel sistema, tieni presente:
>
> - I nomi degli argomenti e i filtri degli argomenti distinguono tra maiuscole e minuscole.
> - I nomi degli argomenti non devono contenere informazioni di identificazione personale.
> - I nomi degli argomenti che iniziano con $ sono argomenti riservatiche possono essere utilizzati solo da AWS IoT Core.
> - AWS IoT Core non posso inviare o ricevere messaggi tra Account AWS s o regioni.

Per ulteriori informazioni sulla progettazione dei nomi degli argomenti e dello spazio dei nomi, consulta il nostro whitepaper Progettazione di argomenti MQTT per AWS IoT Core.

Per esempi di come le app possono pubblicare e sottoscrivere messaggi, inizia con Guida introduttiva ai AWS IoT Core tutorial e AWS IoT Client per dispositivi SDKs SDKs, dispositivi mobili e AWS IoT dispositivi mobili.

> **Important**
>
> Lo spazio dei nomi degli argomenti è limitato a una regione Account AWS and. Ad esempio, l'`sensor/temp/room1`argomento utilizzato da una Account AWS regione è distinto dall'`sensor/temp/room1`argomento utilizzato dallo stesso AWS account in un'altra regione o utilizzato da qualsiasi altro Account AWS in qualsiasi regione.

Topic ARN (ARN argomento)

Tutti gli argomenti ARNs (Amazon Resource Names) hanno il seguente formato:

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

Ad esempio `arn:aws:iot:us-west-2:123EXAMPLE456:topic/application/topic/device/sensor` è un ARN per l'argomento `application/topic/device/sensor`.

Filtri per i nomi degli argomenti

I clienti abbonati registrano i filtri dei nomi degli argomenti con il broker di messaggi per specificare gli argomenti dei messaggi che il broker dei messaggi deve inviare loro. Un filtro per i nomi di argomento può essere un singolo nome di argomento per sottoscrivere un singolo nome di argomento oppure può includere caratteri jolly per sottoscrivere più nomi di argomento contemporaneamente.

I client di pubblicazione non possono utilizzare caratteri jolly nei nomi degli argomenti che pubblicano.

Nella tabella seguente sono elencati i caratteri jolly che possono essere utilizzati in un filtro argomento.

Caratteri jolly per gli argomenti

Carattere jolly.	Corrispondenze	Note
#	Tutte le stringhe al livello e al di sotto del livello nella gerarchia degli argomenti.	Deve essere l'ultimo carattere nel filtro argomento. Deve essere l'unico carattere nel suo livello della gerarchia degli argomenti.

Carattere jolly.	Corrispondenze	Note
		Può essere utilizzato in un filtro argomento che contiene anche il carattere jolly +.
+	Qualsiasi stringa nel livello che contiene il carattere.	Deve essere l'unico carattere nel suo livello della gerarchia degli argomenti. Può essere utilizzato in più livelli di un filtro argomento.

Utilizzo di caratteri jolly con gli esempi di nomi dell'argomento del sensore precedenti:

- Una sottoscrizione di `sensor/#` riceve i messaggi pubblicati in `sensor/`, `sensor/temperature` e `sensor/temperature/room1`, ma non i messaggi pubblicati in `sensor`.
- Una sottoscrizione a `sensor/+/room1` riceve i messaggi pubblicati a `sensor/temperature/room1` e `sensor/humidity/room1`, ma non messaggi inviati a `sensor/temperature/room2` o `sensor/humidity/room2`.

ARN del filtro argomenti

Il filtro per tutti gli argomenti ARNs (Amazon Resource Names) ha la seguente forma:

```
arn:aws:iot:aws-region:AWS-account-ID:topicfilter/TopicFilter
```

Ad esempio, `arn:aws:iot:us-west-2:123EXAMPLE456:topicfilter/application/topic/+/sensor` è un ARN per il filtro argomento `application/topic/+/sensor`.

Payload del messaggio MQTT

Il payload dei messaggi che viene inviato nei tuoi messaggi MQTT non è specificato da AWS IoT, a meno che non sia per uno dei. the section called “Argomenti riservati” Per soddisfare le esigenze della tua applicazione, ti consigliamo di definire il payload dei messaggi per gli argomenti entro i vincoli di AWS IoT Core Service Quotas per i protocolli.

L'utilizzo di un formato JSON per il payload dei messaggi consente al motore di AWS IoT regole di analizzare i messaggi e applicare ad essi query SQL. Se l'applicazione non richiede che il motore delle regole applichi query SQL ai payload dei messaggi, è possibile utilizzare qualsiasi formato di dati richiesto dall'applicazione. Per informazioni sulle limitazioni e sui caratteri riservati in un documento JSON utilizzato nelle query SQL, consulta Estensioni JSON.

Per ulteriori informazioni sulla progettazione degli argomenti MQTT e dei relativi payload dei messaggi, consulta Progettazione di argomenti MQTT per AWS IoT Core.

Se un limite di dimensioni del messaggio supera le quote di servizio, restituirà un `CLIENT_ERROR` con motivo `PAYLOAD_LIMIT_EXCEEDED` e il messaggio "Message payload exceeds size limit for message type" ("Il payload del messaggio supera il limite di dimensione per il tipo di messaggio"). Per ulteriori informazioni sul limite delle dimensioni dei messaggi, consulta la sezione Limiti e quote del broker di messaggi AWS IoT Core.

Argomenti riservati

Gli argomenti che iniziano con il simbolo del dollaro ($) sono riservati a AWS IoT. È possibile effettuare la sottoscrizione e la pubblicazione di questi argomenti riservati, tuttavia non è possibile creare nuovi argomenti che inizino con il simbolo del dollaro. Le operazioni di pubblicazione o sottoscrizione non supportate di argomenti riservati possono causare la chiusura della connessione.

Argomenti del modello di asset

Topic	Operazioni client consentite	Description
$ aws/sitewise/asset -models/ /assets/ *assetModelId* /properties/ *assetId* *propertyId*	Subscribe	AWS IoT SiteWise pubblica notifiche sulle proprietà degli asset relative a questo argomento. Per ulteriori informazioni, consulta Interazione con altri AWS servizi nella Guida per l'AWS IoT SiteWise utente.

AWS IoT Device Defender argomenti

Questi messaggi supportano i buffer di risposta in formato Concise Binary Object Representation (CBOR) e JavaScript Object Notation (JSON), a seconda dell'argomento. *payload-format* AWS IoT Device Defender gli argomenti supportano solo la pubblicazione in formato MQTT.

payload-format	Tipo di dati del formato della risposta
cbor	Concise Binary Object Representation (CBOR)
json	JavaScript Notazione di oggetti (JSON)

Per ulteriori informazioni, consulta Invio di metriche dai dispositivi.

Topic	Operazioni consentite	Description
$aws/things/ *thingName* /defender/metrics/ *payload-format*	Pubblica	AWS IoT Device Defender gli agenti pubblicano le metriche relative a questo argomento. Per ulteriori informazioni, consulta Invio di metriche dai dispositivi.
$aws/things/ /defender /metrics/ /accepted *thingName payload-format*	Subscribe	AWS IoT viene pubblicato su questo argomento dopo che un agente ha pubblicato con successo un messaggio su $aws/things/ /defender/metrics/. AWS IoT Device Defender *thingName payload-format* Per ulteriori informazioni, consulta Invio di metriche dai dispositivi.
$aws/things/ /defender /metrics/ /rejected *thingName payload-format*	Subscribe	AWS IoT viene pubblicato su questo argomento dopo che un agente ha pubblicato un messaggio non riuscito su $aws/things/ /defender/metrics/. AWS IoT Device Defender *thingName payload-format* Per ulteriori informazi

Topic	Operazioni consentite	Description
		oni, consulta Invio di metriche dai dispositi vi.

AWS IoT Core Argomenti sulla posizione dei dispositivi

AWS IoT Core Device Location può risolvere i dati di misurazione dal dispositivo e fornire una posizione stimata dei dispositivi IoT. I dati di misurazione del dispositivo possono includere GNSS, Wi-Fi, rete cellulare e indirizzo IP. AWS IoT Core Device Location sceglie quindi il tipo di misurazione che offre la massima precisione e risolve le informazioni sulla posizione del dispositivo. Per ulteriori informazioni, consultare AWS IoT Core Ubicazione del dispositivo e Risoluzione della posizione del dispositivo utilizzando gli argomenti MQTT di AWS IoT Core Device Location.

Topic	Operazioni consentite	Description
$aws/device_location/ / get_position_estimate *customer_device_id*	Pubblica	Un dispositivo pubblica su questo argomento per ottenere i dati di misurazio ne grezzi scansionati da risolvere tramite Device Location. AWS IoT Core
$aws/device_location/ / get_position_estimate/ accepted *customer_ device_id*	Subscribe	AWS IoT Core Device Location viene pubblicato su questo argomento dopo aver risolto correttamente la posizione del dispositivo.
$aws/device_location/ / get_position_estimate/ rejected *customer_ device_id*	Subscribe	AWS IoT Core Device Location viene pubblicato su questo argomento quando non è in grado di risolvere correttamente la posizione del dispositivo a causa di errori 4xx.

Argomenti di eventi

I messaggi relativi agli eventi vengono pubblicati quando si verificano determinati eventi. Ad esempio, vengono generati eventi dal registro quando vengono aggiunti, aggiornati o eliminati oggetti. La tabella mostra i vari AWS IoT eventi e i relativi argomenti riservati.

Topic	Operazioni client consentite	Description
$aws/events/certificates/registered/*caCertificateId*	Subscribe	AWS IoT pubblica questo messaggio quando registra AWS IoT automaticamente un certificato e quando un client presenta un certificato con lo `PENDING_ACTIVATION` stato. Per ulteriori informazioni, consulta the section called “Configurare la prima connessione da parte di un client per la registrazione automatica”.
$/aws/events/job/annullato *jobID*	Subscribe	AWS IoT pubblica questo messaggio quando un lavoro viene annullato. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$//cancellazione_in corso aws/events/job *jobID*	Subscribe	AWS IoT pubblica questo messaggio quando è in corso l'annullamento di un lavoro. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/job//completato *jobID*	Subscribe	AWS IoT pubblica questo messaggio quando un lavoro è completato. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/job//cancellato *jobID*	Subscribe	AWS IoT pubblica questo messaggio quando viene eliminato un lavoro. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/job/*jobID*/deletion_in_progress	Subscribe	AWS IoT pubblica questo messaggio quando è in corso l'eliminazione di un lavoro. Per ulteriori informazioni, consulta Eventi del servizio Jobs.

Topic	Operazioni client consentite	Description
$aws/events/jobExecution/*jobID*/annullato	Subscribe	AWS IoT pubblica questo messaggio quando l'esecuzione di un lavoro viene annullata. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$//eliminato aws/events/jobExecution *jobID*	Subscribe	AWS IoT pubblica questo messaggio quando viene eliminata l'esecuzione di un lavoro. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/jobExecution/*jobID*/non riuscito	Subscribe	AWS IoT pubblica questo messaggio quando l'esecuzione di un lavoro non è riuscita. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/jobExecution//rifiutato *jobID*	Subscribe	AWS IoT pubblica questo messaggio quando l'esecuzione di un lavoro è stata rifiutata. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/jobExecution//rimosso *jobID*	Subscribe	AWS IoT pubblica questo messaggio quando è stata rimossa l'esecuzione di un lavoro. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/jobExecution//riuscito *jobID*	Subscribe	AWS IoT pubblica questo messaggio quando l'esecuzione di un lavoro è riuscita. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$//timed_out aws/events/jobExecution *jobID*	Subscribe	AWS IoT pubblica questo messaggio quando scade il timeout dell'esecuzione di un job. Per ulteriori informazioni, consulta Eventi del servizio Jobs.

Topic	Operazioni client consentite	Description
$aws/events/presence/connected/*clientId*	Subscribe	AWS IoT pubblica su questo argomento quando un client MQTT con l'ID client specificato si connette a. AWS IoT Per ulteriori informazioni, consulta Eventi di connessione/disconnessione.
$aws/events/presence/disconnected/*clientId*	Subscribe	AWS IoT pubblica su questo argomento quando un client MQTT con l'ID client specificato si disconnette a. AWS IoT Per ulteriori informazioni, consulta Eventi di connessione/disconnessione.
$aws/events/subscriptions/subscribed/*clientId*	Subscribe	AWS IoT pubblica su questo argomento quando un client MQTT con l'ID client specificato sottoscrive un argomento MQTT. Per ulteriori informazioni, consulta Eventi di sottoscrizione/annullamento della sottoscrizione.
$aws/events/subscriptions/unsubscribed/*clientId*	Subscribe	AWS IoT pubblica su questo argomento quando un client MQTT con l'ID client specificato annulla l'iscrizione a un argomento MQTT. Per ulteriori informazioni, consulta Eventi di sottoscrizione/annullamento della sottoscrizione.
$//creato aws/events/thing *thingName*	Subscribe	AWS IoT pubblica su questo argomento quando l'*thingName* oggetto viene creato. Per ulteriori informazioni, consulta the section called “Eventi del registro”.

Topic	Operazioni client consentite	Description
$aws/events/thing//aggiornato *thingName*	Subscribe	AWS IoT viene pubblicato su questo argomento quando l'*thingName* elemento viene aggiornato. Per ulteriori informazioni, consulta the section called “Eventi del registro”.
$aws/events/thing//eliminato *thingName*	Subscribe	AWS IoT pubblica su questo argomento quando l'*thingName* elemento viene eliminato. Per ulteriori informazioni, consulta the section called “Eventi del registro”.
$aws/events/thingGroup//created *thingGroupName*	Subscribe	AWS IoT pubblica su questo argomento quando viene creato il gruppo di oggetti*thingGroupName* . Per ulteriori informazioni, consulta the section called “Eventi del registro”.
$aws/events/thingGroup//aggiornato *thingGroupName*	Subscribe	AWS IoT pubblica su questo argomento quando il gruppo di cose *thingGroupName* viene aggiornato. Per ulteriori informazioni, consulta the section called “Eventi del registro”.
$aws/events/thingGroup//eliminato *thingGroupName*	Subscribe	AWS IoT pubblica su questo argomento quando il gruppo di oggetti *thingGroupName* viene eliminato. Per ulteriori informazioni, consulta the section called “Eventi del registro”.
$aws/events/thingType//created *thingTypeName*	Subscribe	AWS IoT pubblica su questo argomento quando viene creato il tipo di *thingTypeName* oggetto. Per ulteriori informazioni, consulta the section called “Eventi del registro”.

Topic	Operazioni client consentite	Description
$aws/events/thingType// aggiornato *thingType Name*	Subscribe	AWS IoT viene pubblicato su questo argomento quando il tipo di *thingType Name* oggetto viene aggiornato. Per ulteriori informazioni, consulta the section called “Eventi del registro”.
$aws/events/thingType// eliminato *thingType Name*	Subscribe	AWS IoT pubblica su questo argomento quando il tipo di *thingTypeName* oggetto viene eliminato. Per ulteriori informazioni, consulta the section called “Eventi del registro”.
$aws/events/thingT ypeAssociation/ thing/*thingName* /*thingTypeName*	Subscribe	AWS IoT pubblica su questo argomento quando l'oggetto *thingName* è associato o dissociato dal tipo di oggetto. *thingTypeName* Per ulteriori informazi oni, consulta the section called “Eventi del registro”.
$aws/events/thingG roupMembership/thi ngGroup//thing/ /added *thingGroupName* *thingName*	Subscribe	AWS IoT pubblica su questo argomento quando un oggetto *thingName* viene aggiunto al gruppo di oggetti. *thingGrou pName* Per ulteriori informazioni, consulta the section called “Eventi del registro”.
$aws/events/thingG roupMembership/ thingGroup//thing/ *thingGroupName* / rimosso *thingName*	Subscribe	AWS IoT pubblica su questo argomento quando un oggetto *thingName* viene rimosso dal gruppo di oggetti. *thingGroupName* Per ulteriori informazioni, consulta the section called “Eventi del registro”.

Topic	Operazioni client consentite	Description
$aws/events/thingGroupHierarchy/thingGroup/*parentThingGroupName* childThingGroup//aggiunto *childThingGroupName*	Subscribe	AWS IoT pubblica su questo argomento quando il gruppo di oggetti *childThingGroupName* viene aggiunto al gruppo di oggetti. *parentThingGroupName* Per ulteriori informazioni, consulta the section called “Eventi del registro”.
$aws/events/thingGroupHierarchy/thingGroup/*parentThingGroupName* childThingGroup//rimosso *childThingGroupName*	Subscribe	AWS IoT pubblica su questo argomento quando il gruppo di oggetti *childThingGroupName* viene rimosso dal gruppo di oggetti. *parentThingGroupName* Per ulteriori informazioni, consulta the section called “Eventi del registro”.

Argomenti relativi al provisioning del parco istanze

> **Note**
>
> Le operazioni client contrassegnate come Receive in questa tabella indicano gli argomenti che vengono AWS IoT pubblicati direttamente sul client che li ha richiesti, indipendentemente dal fatto che il client abbia sottoscritto l'argomento o meno. I clienti dovrebbero aspettarsi di ricevere questi messaggi di risposta anche se non si sono abbonati a loro. Questi messaggi di risposta non passano attraverso il broker di messaggi e non possono essere sottoscritti da altri client o regole.

Questi messaggi supportano i buffer di risposta in formato Concise Binary Object Representation (CBOR) e JavaScript Object Notation (JSON), a seconda dell'argomento. *payload-format*

payload-format	Tipo di dati del formato della risposta
cbor	Concise Binary Object Representation (CBOR)
json	JavaScript Notazione di oggetti (JSON)

Per ulteriori informazioni, consulta API MQTT di provisioning del dispositivo.

Topic	Operazioni client consentite	Description
$aws/certificates/create/*payload-format*	Pubblica	Pubblicare in questo argomento per creare un certificato da una richiesta di firma del certificato (CSR).
$//accettato aws/certificates/create *payload-format*	Sottoscriviti, Ricevi	AWS IoT pubblica su questo argomento dopo una chiamata riuscita a $aws/certificates/create/. *payload-format*
$aws/certificates/create//rifiutato *payload-format*	Sottoscriviti, Ricevi	AWS IoT pubblica su questo argomento dopo una chiamata non riuscita a $/. aws/certificates/create *payload-format*
$ -from-csr/ aws/certificates/create *payload-format*	Pubblica	Pubblica in questo argomento per creare un certificato da una CSR.
$ aws/certificates/create -from-csr/ /accettato *payload-format*	Sottoscriviti, Ricevi	AWS IoT pubblica su questo argomento una chiamata riuscita a $ -from-csr/. aws/certificates/create *payload-format*
$ -from-csr/ /rejected aws/certificates/create *payload-format*	Sottoscriviti, Ricevi	AWS IoT pubblica su questo argomento una chiamata non riuscita a $ -from-csr /. aws/certificates/create *payload-format*

Topic	Operazioni client consentite	Description
templateName $aws/provisioning-templates/ /provision/ *payload-format*	Pubblica	Pubblica in questo argomento per registrare un oggetto.
$aws/provisioning-templates/ *templateName* /provision/ /accepted *payload-format*	Sottoscriviti, Ricevi	AWS IoT pubblica su questo argomento dopo una chiamata riuscita a $aws/provisioning-templates/ /provision/. *templateName* *payload-format*
$aws/provisioning-templates/ /provision/ /rejected *templateName* *payload-format*	Sottoscriviti, Ricevi	AWS IoT pubblica su questo argomento dopo una chiamata non riuscita a $aws/provisioning-templates/ /provision/. *templateName* *payload-format*

Argomenti di processo

> **Note**
>
> Le operazioni client indicate come Receive in questa tabella indicano gli argomenti che vengono AWS IoT pubblicati direttamente sul client che li ha richiesti, indipendentemente dal fatto che il client abbia sottoscritto l'argomento o meno. I clienti dovrebbero aspettarsi di ricevere questi messaggi di risposta anche se non si sono abbonati a loro.
> Questi messaggi di risposta non passano attraverso il broker di messaggi e non possono essere sottoscritti da altri client o regole. Per effettuare la sottoscrizione ai messaggi relativi all'attività di lavoro, utilizza il `notify` e gli argomenti `notify-next`.
> Quando si sottoscrive il processo e argomenti di evento `jobExecution` per la tua soluzione di monitoraggio del parco istanze, devi prima abilitare eventi di processo e di esecuzione di processi per ricevere eventi sul lato cloud.
> Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.

Topic	Operazioni client consentite	Description
$aws/things/ /jobs/get *thingName*	Pubblica	I dispositivi pubblicano un messaggio in questo argomento per effettuare una richiesta `GetPendingJobExecutions` . Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.
$aws/cose/ *thingName* jobs/get/accepted	Sottoscriviti, Ricevi	I dispositivi si sottoscrivono a questo argomento per ricevere risposte corrette da una richiesta `GetPendingJobExecutions` . Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.
thingName $aws/cose// jobs/get/rejected	Sottoscriviti, Ricevi	I dispositivi sottoscrivono questo argomento per ricevere una risposta quando una richiesta `GetPendingJobExecutions` viene rifiutata. Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.
$aws/things/ *thingName* / jobs/start-next	Pubblica	I dispositivi pubblicano un messaggio in questo argomento per effettuare una richiesta `StartNextPendingJobExecution` . Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.
$ aws/cose/ *thingName* jobs/start-next/accepted	Sottoscriviti, Ricevi	I dispositivi si sottoscrivono a questo argomento per ricevere risposte corrette per una richiesta `StartNextPendingJobExecution` . Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.

Topic	Operazioni client consentite	Description
thingName $aws/cose//jobs/start-next/rejected	Sottoscriviti, Ricevi	I dispositivi sottoscrivono questo argomento per ricevere una risposta quando una richiesta `StartNextPendingJobExecution` viene rifiutata. Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.
$aws/things/ /jobs/ /get *thingName* *jobId*	Pubblica	I dispositivi pubblicano un messaggio in questo argomento per effettuare una richiesta `DescribeJobExecution`. Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.
$aws/things/ /jobs/ *thingName* /get/accepted *jobId*	Sottoscriviti, Ricevi	I dispositivi si sottoscrivono a questo argomento per ricevere risposte corrette per una richiesta `DescribeJobExecution`. Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.
$aws/things/ /jobs/ *thingName* /get/rejected *jobId*	Sottoscriviti, Ricevi	I dispositivi sottoscrivono questo argomento per ricevere una risposta quando una richiesta `DescribeJobExecution` viene rifiutata. Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.
$aws/things/ /jobs/ *thingName* /update *jobId*	Pubblica	I dispositivi pubblicano un messaggio in questo argomento per effettuare una richiesta `UpdateJobExecution`. Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.

Topic	Operazioni client consentite	Description
$aws/things/ /jobs/ *thingName* /update/accepted *jobId*	Sottoscriviti, Ricevi	I dispositivi si sottoscrivono a questo argomento per ricevere risposte corrette per una richiesta `UpdateJobExecution`. Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi. **Nota** Solo il dispositivo che pubblica su $aws/things/ /jobs/ /update riceve messaggi su questo argomento. *thingName* *jobId*
$aws/things/ /jobs/ /update/rejected *thingName* *jobId*	Sottoscriviti, Ricevi	I dispositivi sottoscrivono questo argomento per ricevere una risposta quando una richiesta `UpdateJobExecution` viene rifiutata. Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi. **Nota** Solo il dispositivo che pubblica su $aws/things/ /jobs/ /update riceve messaggi su questo argomento. *thingName* *jobId*

Topic	Operazioni client consentite	Description
$aws/things/ /jobs/notify *thingName*	Sottoscriviti, Ricevi	I dispositivi si sottoscrivono a questo argomento per ricevere notifiche quando viene aggiunta o rimossa l'esecuzione di un lavoro nell'elenco di esecuzioni in sospeso per una cosa. Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.
$aws/things/ /jobs/notify-next *thingName*	Sottoscriviti, Ricevi	I dispositivi si sottoscrivono a questo argomento per ricevere notifiche quando viene modificata la successiv a esecuzione del processo in sospeso per la cosa. Per ulteriori informazioni, consulta Operazioni API MQTT del dispositivo dei processi.
$//completato aws/events/ job *jobId*	Subscribe	Il servizio Jobs pubblica un evento in questo argomento al termine di un processo. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/job/*jobId*/ annullato	Subscribe	Il servizio Jobs pubblica un evento in questo argomento quando un processo viene annullato. Per ulteriori informazi oni, consulta Eventi del servizio Jobs.
$//eliminato aws/events/job *jobId*	Subscribe	Il servizio Jobs pubblica un evento su questo argomento quando un lavoro viene cancellato. Per ulteriori informazi oni, consulta Eventi del servizio Jobs.

Topic	Operazioni client consentite	Description
$aws/events/job/*jobId*/ cancellazione_in corso	Subscribe	Il servizio Jobs pubblica un evento in questo argomento quando inizia l'annullamento del processo. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/job/*jobId*/ eliminazione_in corso	Subscribe	Il servizio Jobs pubblica un evento in questo argomento quando inizia l'elimina zione di un processo. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$ aws/events/jobExec ution*jobId*//successo	Subscribe	Il servizio Jobs pubblica un evento in questo argomento quando l'esecuzi one del processo ha esito positivo. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$/aws/events/jobExecution/ non riuscito *jobId*	Subscribe	Il servizio Jobs pubblica un evento in questo argomento quando l'esecuzione di un processo non riesce. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/jobExecution//r ifiutato *jobId*	Subscribe	Il servizio Jobs pubblica un evento in questo argomento quando viene respinta l'esecuzione di un processo. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/jobExe cution/*jobId*/annullato	Subscribe	Il servizio Jobs pubblica un evento in questo argomento quando viene annullata l'esecuzione di un processo. Per ulteriori informazioni, consulta Eventi del servizio Jobs.

Topic	Operazioni client consentite	Description
$//time_out aws/events/ jobExecution *jobId*	Subscribe	Il servizio Jobs pubblica un evento in questo argomento quando si verifica il timeout dell'esecuzione di un processo. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$ aws/events/jobExec ution*jobId*//rimosso	Subscribe	Il servizio Jobs pubblica un evento in questo argomento quando viene rimossa l'esecuzione di un processo. Per ulteriori informazioni, consulta Eventi del servizio Jobs.
$aws/events/jobExecution// eliminato *jobId*	Subscribe	Il servizio Jobs pubblica un evento in questo argomento quando viene eliminata l'esecuzione di un processo. Per ulteriori informazioni, consulta Eventi del servizio Jobs.

Argomenti sui comandi

> **Note**
>
> Le operazioni client indicate come Receive in questa tabella indicano gli argomenti che vengono AWS IoT pubblicati direttamente sul client che li ha richiesti, indipendentemente dal fatto che il client abbia sottoscritto l'argomento o meno. I clienti dovrebbero aspettarsi di ricevere questi messaggi di risposta anche se non si sono abbonati a loro.
> Questi messaggi di risposta non passano attraverso il broker di messaggi e non possono essere sottoscritti da altri client o regole.

Topic	Operazioni client consentite	Description
$aws/commands/ /executions/ /request/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>* $aws/comandi/ *<devices> <DeviceID>* /esecuzioni/ /richiesta *<ExecutionId>*	Sottoscriviti, Ricevi	I dispositivi ricevono un messaggio su questo argomento quando viene effettuata una richiesta per avviare l'esecuzione di un comando dalla console o utilizzando l'API. `StartCommandExecution` In questo caso, *<devices>* possono essere oggetti IoT o client MQTT e *<DeviceID>* possono essere il nome dell'oggetto IoT o l'ID del client MQTT.
$aws/commands///esecuzioni/ /response/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>*	Pubblica	I dispositivi utilizzano l'API `UpdateCommandExecution` MQTT per pubblicare un messaggio su questo argomento sull'esecuzione del comando. Il messaggio viene pubblicato come risposta alla richiesta di avviare l'esecuzione di un comando dalla console o utilizzando l'`StartCommandExecution` API. Il messaggio pubblicato utilizzerà JSON o CBOR come. *<PayloadFormat>*
$aws/commands///executions/ /response/ accepted/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>* $aws/comandi/ *<devices> <DeviceID>* /esecuzioni/ /response/accepted *<ExecutionId>*	Sottoscriviti, Ricevi	Se il servizio cloud ha elaborato correttamente il risultato dell'esecuzione del comando, pubblica una risposta all'argomento /accepted. AWS IoT Device Management

Topic	Operazioni client consentite	Description
$aws/commands///executions/ /response/rejected/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>* $aws/comandi/ *<devices> <DeviceID>* /esecuzioni/ /response/rejected *<ExecutionId>*	Pubblica	Se il servizio cloud non è riuscito a elaborare il risultato dell'esecuzione del comando, pubblica una risposta all'argomento /rejecte. AWS IoT Device Management

Argomenti di regole

Topic	Operazioni client consentite	Description
$aws/regole/ *ruleName*	Pubblica	Un dispositivo o un'applicazione pubblica su questo argomento per arrivare le regole direttamente. Per ulteriori informazioni, consulta Riduzione dei costi di messaggistica con Basic Ingest.

Argomenti di tunneling sicuro

Topic	Operazioni client consentite	Description
$aws/things/ /tunnels/notify *thing-name*	Subscribe	AWS IoT pubblica questo messaggio per consentire a un agente IoT di avviare un proxy locale sul dispositivo remoto.

Topic	Operazioni client consentite	Description
		Per ulteriori informazioni, consulta the section called “Snippet dell'agente IoT”.

Argomenti copie shadow

Gli argomenti di questa sezione vengono utilizzati da copie shadow con nome e senza nome. Gli argomenti utilizzati da ciascuno differiscono solo nel prefisso dell'argomento. Questa tabella mostra il prefisso dell'argomento utilizzato da ogni tipo di copia shadow.

ShadowTopicPrefix value	Tipo di copia shadow
$aws/things/ /shadow *thingName*	Copia shadow senza nome (classica)
$aws/things/ *thingName* /shadow/name/ *shadowName*	Copia shadow con nome

Per creare un argomento completo, seleziona il *ShadowTopicPrefix* tipo di ombra a cui vuoi fare riferimento, sostituiscilo *thingName* e, se applicabile, con i valori corrispondenti*shadowName*, quindi aggiungilo con lo stub dell'argomento come mostrato nella tabella seguente. Ricorda che gli argomenti prevedono una distinzione tra lettere maiuscole e minuscole.

Topic	Operazioni client consentite	Description
ShadowTopicPrefix / elimina	Pubblicazione/Sottoscrizione	Un dispositivo o un'applicazione pubblica in questo argomento per eliminare una copia shadow. Per ulteriori informazioni, consulta /delete.
ShadowTopicPrefix / elimina/accettato	Subscribe	Il servizio Device Shadow invia messaggi a questo argomento quando viene eliminata una copia shadow. Per

Topic	Operazioni client consentite	Description
		ulteriori informazioni, consulta /delete/accepted.
ShadowTopicPrefix / elimina/rifiutato	Subscribe	Il servizio Device Shadow invia messaggi a questo argomento quando viene rifiutata una richiesta di eliminazione di una copia shadow. Per ulteriori informazioni, consulta /delete/rejected.
ShadowTopicPrefix / ottenere	Pubblicazione/Sottoscrizione	Un'applicazione o un oggetto pubblica un messaggio vuoto in questo argomento per ottenere una copia shadow. Per ulteriori informazioni, consulta Argomenti MQTT di Device Shadow.
ShadowTopicPrefix / get/accepted	Subscribe	Il servizio Device Shadow invia messaggi a questo argomento quando viene effettuata correttamente una richiesta di una copia shadow. Per ulteriori informazioni, consulta /get/accepted.
ShadowTopicPrefix / get/rejected	Subscribe	Il servizio Device Shadow invia messaggi a questo argomento quando viene rifiutata una richiesta di una copia shadow. Per ulteriori informazioni, consulta /get/rejected.
ShadowTopicPrefix / aggiornare	Pubblicazione/Sottoscrizione	Un oggetto o un'applicazione pubblica in questo argomento per aggiornare una copia shadow. Per ulteriori informazioni, consulta /update.

Topic	Operazioni client consentite	Description
ShadowTopicPrefix / update/accettato	Subscribe	Il servizio Device Shadow invia messaggi a questo argomento quando viene effettuato correttamente un aggiornamento di una copia shadow. Per ulteriori informazioni, consulta /update/accepted.
ShadowTopicPrefix / update/rifiutato	Subscribe	Il servizio Device Shadow invia messaggi a questo argomento quando viene rifiutato un aggiornamento di una copia shadow. Per ulteriori informazioni, consulta /update/rejected.
ShadowTopicPrefix / update/delta	Subscribe	Il servizio Device Shadow invia messaggi a questo argomento quando viene rilevata una differenza tra le sezioni sullo stato segnalato e sullo stato desiderato di una copia shadow. Per ulteriori informazioni, consulta /update/delta.
ShadowTopicPrefix / aggiornamento/documenti	Subscribe	AWS IoT pubblica un documento di stato su questo argomento ogni volta che un aggiornamento dello shadow viene eseguito correttamente. Per ulteriori informazioni, consulta /update/documents.

Argomenti di distribuzione dei file basati su MQTT

> **Note**
>
> Le operazioni client contrassegnate come Receive in questa tabella indicano gli argomenti che vengono AWS IoT pubblicati direttamente sul client che li ha richiesti, indipendentemente

dal fatto che il client abbia sottoscritto l'argomento o meno. I clienti dovrebbero aspettarsi di ricevere questi messaggi di risposta anche se non si sono abbonati a loro. Questi messaggi di risposta non passano attraverso il broker di messaggi e non possono essere sottoscritti da altri client o regole.

Questi messaggi supportano i buffer di risposta in formato Concise Binary Object Representation (CBOR) e JavaScript Object Notation (JSON), a seconda dell'argomento. *payload-format*

payload-format	Tipo di dati del formato della risposta
cbor	Concise Binary Object Representation (CBOR)
json	JavaScript Notazione di oggetti (JSON)

Topic	Operazioni client consentite	Description
$aws/things/ *ThingName* /streams/ /data/ *StreamId* *payload-format*	Sottoscriviti, Ricevi	AWS La distribuzione di file basata su MQTT viene pubblicata su questo argomento se viene accettata la richiesta "" da un dispositivo. GetStream Il payload contiene i dati del flusso. Per ulteriori informazioni, consulta Utilizzo della distribuzione di file AWS IoT basata su MQTT nei dispositivi.
$aws/things/ /streams/ /get/ *ThingName* *StreamId* *payload-format*	Pubblica	Un dispositivo pubblica su questo argomento per eseguire una richiesta "». GetStream Per ulteriori informazioni, consulta Utilizzo della distribuzione di file AWS IoT basata su MQTT nei dispositivi.
$aws/things/ /streams/ *ThingName* /descrizione/	Sottoscriviti, Ricevi	AWS La distribuzione di file basata su MQTT viene pubblicata su questo

Topic	Operazioni client consentite	Description
StreamId payload-format		argomento se viene accettata la richiesta "" da un dispositivo. DescribeStream Il payload contiene la descrizione del flusso. Per ulteriori informazioni, consulta Utilizzo della distribuzione di file AWS IoT basata su MQTT nei dispositivi.
$aws/things/ /streams/ /describe/ *ThingName StreamId payload-format*	Pubblica	Un dispositivo pubblica su questo argomento per eseguire una richiesta "». DescribeStream Per ulteriori informazioni, consulta Utilizzo della distribuzione di file AWS IoT basata su MQTT nei dispositivi.
$aws/things/ /streams/ /rejected/ *ThingName StreamId payload-format*	Sottoscriviti, Ricevi	AWS La distribuzione di file basata su MQTT viene pubblicata su questo argomento se una richiesta "" o "" proveniente da un dispositivo viene rifiutata. DescribeStream GetStream Per ulteriori informazioni, consulta Utilizzo della distribuzione di file AWS IoT basata su MQTT nei dispositivi.

ARN di argomenti riservati

Tutti gli argomenti riservati ARNs (Amazon Resource Names) hanno il seguente formato:

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

Ad esempio, `arn:aws:iot:us-west-2:123EXAMPLE456:topic/$aws/things/thingName/jobs/get/accepted` è un ARN per l'argomento riservato `$aws/things/thingName/jobs/get/accepted`.

Configurazioni del dominio

Nel AWS IoT Core, puoi utilizzare le configurazioni di dominio per configurare e gestire i comportamenti degli endpoint di dati. Con le configurazioni di dominio, puoi generare più endpoint di AWS IoT Core dati, personalizzarli con nomi di dominio completi (FQDN) e certificati server associati e anche associare un autorizzatore personalizzato. Per ulteriori informazioni, consulta Autenticazione e autorizzazione personalizzata.

> **Note**
>
> Questa funzionalità non è disponibile in. AWS GovCloud (US) Regioni AWS

In questo capitolo:

- In cosa consiste una configurazione di dominio?
- Creazione e configurazione di domini AWS gestiti
- Creazione e configurazione di domini gestiti dai clienti
- Gestione delle configurazioni di dominio
- Configurazione delle impostazioni TLS nelle configurazioni di dominio
- Configurazione del certificato server per stapling OCSP

In cosa consiste una configurazione di dominio?

In AWS IoT Core, una configurazione di dominio si riferisce alla configurazione e alla configurazione di un dominio (dominio AWS gestito o dominio gestito dal cliente) per gli endpoint di AWS IoT Core dati. AWS IoT Core fornisce inoltre un endpoint predefinito per l' AWS account (`iot:Data-ATS`) con cui i dispositivi possono comunicare. AWS IoT Core

In questo argomento:

- Casi d’uso
- Concetti chiave
- Note importanti

Casi d’uso

È possibile utilizzare le configurazioni di dominio per semplificare attività come le seguenti.

- Esegui la migrazione dei dispositivi su. AWS IoT Core
- Supporta parchi di dispositivi eterogenei mantenendo configurazioni di dominio separate per tipi di dispositivo separati.
- Mantieni l'identità del marchio (ad esempio, tramite il nome di dominio) durante la migrazione dell'infrastruttura applicativa a. AWS IoT Core

Concetti chiave

I seguenti concetti forniscono dettagli sulle configurazioni dei domini e sui concetti correlati.

- Configurazione del dominio

  La configurazione e la configurazione di un dominio per i tuoi AWS IoT Core endpoint.

- Dominio endpoint predefinito

  Il dominio che AWS IoT fornisce l'endpoint predefinito, ad esempio. `iot:Data-ATS` Per trovare l'endpoint predefinito, esegui il comando describe-endpoint o CLI. describe-domain-configuration In alternativa, vai alla AWS IoT Core console e scegli Configurazioni di dominio da Connect nella barra di navigazione a sinistra. L'endpoint predefinito è elencato con il nome. `iot:Data-ATS`

- AWS dominio gestito

  Il dominio che AWS gestirà. La scelta di un dominio AWS gestito significa che i dispositivi si connetteranno utilizzando un endpoint di dati fornito da AWS. AWS gestirà il dominio e i certificati.

- dominio gestito dal cliente

  Il dominio che gestirai. Noto anche come dominio personalizzato. La scelta di un dominio gestito dal cliente significa che i dispositivi si connetteranno utilizzando un endpoint di dati di dominio personalizzato. Gestirai il dominio e i certificati. Il dominio gestito dal cliente ti consente di personalizzare l'endpoint in base URLs alle tue esigenze. Ad esempio, puoi utilizzare un nome di dominio personalizzato (`your-domain-name.com`) o applicare politiche di accesso specifiche.

- Tipo di autenticazione

Il tipo di autenticazione a cui scegli per autenticare i tuoi dispositivi durante la AWS IoT Core connessione. Quando si crea una configurazione di dominio, è necessario specificare un tipo di autenticazione. Per ulteriori informazioni, consulta ???.

- Protocollo di applicazione

  I protocolli a livello di applicazione utilizzati dai dispositivi per la connessione AWS IoT Core. Quando si crea una configurazione di dominio, è necessario specificare un protocollo applicativo. Per ulteriori informazioni, consulta ???.

Note importanti

AWS IoT Core utilizza l'estensione TLS SNI (Server Name Indication) per applicare le configurazioni di dominio. Quando si collegano i dispositivi AWS IoT Core, i client possono inviare l'estensione Server Name Indication (SNI), necessaria per funzionalità come la registrazione di più account, endpoint configurabili, domini personalizzati ed endpoint VPC. Devono inoltre passare un nome server identico a quello specificato nella configurazione del dominio. Per testare questo servizio, utilizza la versione v2 del dispositivo in.AWS IoT SDKs GitHub

Se crei più endpoint di dati nel tuo Account AWS, questi condivideranno AWS IoT Core risorse come argomenti MQTT, ombre dei dispositivi e regole.

Quando fornisci i certificati del server per la configurazione AWS IoT Core personalizzata del dominio, i certificati hanno un massimo di quattro nomi di dominio. Per ulteriori informazioni, consulta Endpoint e quote per AWS IoT Core.

Creazione e configurazione di domini AWS gestiti

Puoi creare un endpoint configurabile su un dominio AWS gestito utilizzando l'API. CreateDomainConfiguration Una configurazione di dominio per un dominio AWS gestito è composta da quanto segue:

- `domainConfigurationName`

  Un nome definito dall'utente che identifica la configurazione del dominio e il valore devono essere univoci per il tuo. Regione AWS Non è possibile utilizzare i nomi di configurazioni del dominio che iniziano con `IoT:` perché sono riservati agli endpoint predefiniti.
- `defaultAuthorizerName` (facoltativo)

Il nome dell'autorizzazione ad hoc da utilizzare nell'endpoint.

- `allowAuthorizerOverride` (facoltativo)

  Un valore booleano che specifica se i dispositivi possono sovrascrivere l'autorizzazione predefinita specificando un'autorizzazione differente nell'intestazione HTTP della richiesta. Questo valore è obbligatorio se viene specificato un valore per `defaultAuthorizerName`.

- `serviceType` (facoltativo)

  Il tipo di servizio fornito dall'endpoint. AWS IoT Core supporta solo il tipo `DATA` di servizio. Quando specifichi `DATA`, AWS IoT Core restituisce un endpoint con un endpoint di tipo `iot:Data-ATS`. Non è possibile creare un endpoint configurabile `iot:Data` (VeriSign).

- `TlsConfig` (facoltativo)

  Un oggetto che specifica la configurazione TLS per un dominio. Per ulteriori informazioni, consulta ???.

Il AWS CLI comando di esempio seguente crea una configurazione di dominio per un `Data` endpoint.

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" --service-type "DATA"
```

L'output del comando può essere simile al seguente.

```
{
    "domainConfigurationName": "myDomainConfigurationName",
    "domainConfigurationArn": "arn:aws:iot:us-east-1:123456789012:domainconfiguration/
myDomainConfigurationName/itihw"
}
```

Creazione e configurazione di domini gestiti dai clienti

Le configurazioni di dominio consentono di specificare un nome di dominio completo personalizzato (FQDN) a cui connettersi a AWS IoT Core. L'utilizzo dei domini gestiti dai clienti (noti anche come domini personalizzati) offre molti vantaggi: è possibile esporre il proprio dominio o quello della propria azienda ai clienti per scopi di branding; è possibile modificare facilmente il proprio dominio per indirizzarlo a un nuovo broker; è possibile supportare la multi-tenancy per servire clienti con domini diversi all'interno dello stesso Account AWS; e puoi gestire i dettagli dei certificati del tuo server,

come l'autorità di certificazione principale (CA) utilizzata per firmare il certificato, l'algoritmo di firma, la profondità della catena di certificati e il ciclo di vita del certificato.

Il flusso di lavoro per impostare una configurazione di dominio con un dominio personalizzato consiste nelle tre fasi seguenti.

1. Registrazione dei certificati del server in AWS Certificate Manager
2. Creazione di una configurazione di dominio
3. Creazione di record DNS

Registrazione dei certificati del server in Certificate Manager AWS

Prima di creare una configurazione di dominio con un dominio personalizzato, è necessario registrare la catena di certificati server in AWS Certificate Manager (ACM). È possibile utilizzare i tre tipi di certificati server seguenti.

- Certificati pubblici generati da ACM
- Certificati esterni firmati da una CA pubblica
- Certificati esterni firmati da una CA privata

> **Note**
>
> AWS IoT Core considera che un certificato sia firmato da una CA pubblica se è incluso nel pacchetto ca-bundle affidabile di Mozilla.

Requisiti del certificato

Consulta Prerequisites for Importing Certificates (Prerequisiti per l'importazione di certificati) per i requisiti per l'importazione di certificati in ACM. Oltre a questi requisiti, AWS IoT Core aggiunge i seguenti requisiti.

- Il certificato leaf deve includere l'estensione Extended Key Usage x509 v3 con il valore ServerAuth (TLS Web Server Authentication). Se si richiede il certificato da ACM, questa estensione viene aggiunta automaticamente.
- La profondità massima della catena di certificati è di 5 certificati.
- La dimensione massima della catena di certificati è di 16 KB.

- Gli algoritmi crittografici e le dimensioni delle chiavi supportati includono RSA 2048 bit (RSA_2048) ed ECDSA 256 bit (EC_Prime256v1).

Utilizzo di un certificato per più domini

Se si prevede di utilizzare un certificato per coprire più sottodomini, utilizzare un dominio con caratteri jolly nel campo nome comune (CN) o nomi alternativi oggetto (SAN). Ad esempio, utilizzare **`*.iot.example.com`** per coprire dev.iot.example.com, qa.iot.example.com e prod.iot.example.com. Ogni nome di dominio completo (FQDN) richiede la propria configurazione di dominio, ma più configurazioni di dominio possono utilizzare lo stesso valore jolly. Il CN o la SAN devono coprire il nome di dominio completo che si desidera utilizzare come dominio personalizzato. Se SANs presenti, il CN viene ignorato e una SAN deve coprire l'FQDN che si desidera utilizzare come dominio personalizzato. Questa copertura può essere una corrispondenza esatta o una corrispondenza jolly. Dopo che un certificato jolly è stato convalidato e registrato su un account, agli altri account della regione viene impedito di creare domini personalizzati che si sovrappongono al certificato.

Nelle sezioni seguenti viene descritto come ottenere ogni tipo di certificato. Ogni risorsa certificato richiede un Amazon Resource Name (ARN) registrato con ACM da utilizzare quando si crea la configurazione del dominio.

Certificati pubblici generati da ACM

Puoi generare un certificato pubblico per il tuo dominio personalizzato utilizzando l'API. RequestCertificate Quando si genera un certificato in questo modo, ACM convalida la proprietà del dominio personalizzato. Per ulteriori informazioni, consulta Request a Public Certificate (Richiesta di un certificato pubblico) nella AWS Certificate Manager User Guide (Guida per l'utente Amazon Certificate Manager).

Certificati esterni firmati da una CA pubblica

Se disponi già di un certificato server firmato da una CA pubblica (una CA inclusa nel pacchetto ca-bundle affidabile di Mozilla), puoi importare la catena di certificati direttamente in ACM utilizzando l'API. ImportCertificate Per ulteriori informazioni su questa attività, sui prerequisiti e sui requisiti di formato dei certificati, consulta Importazione dei certificati.

Certificati esterni firmati da una CA privata

Se si dispone già di un certificato server firmato da una CA privata o autofirmato, è possibile utilizzare il certificato per creare la configurazione del dominio, ma è anche necessario creare un certificato

pubblico aggiuntivo in ACM per convalidare la proprietà del dominio. A tale scopo, registra la catena di certificati del server in ACM utilizzando l'API. ImportCertificate Per ulteriori informazioni su questa attività, sui prerequisiti e sui requisiti di formato dei certificati, consulta Importazione dei certificati.

Creazione di un certificato di convalida

Dopo aver importato il certificato in ACM, genera un certificato pubblico per il tuo dominio personalizzato utilizzando l'RequestCertificateAPI. Quando si genera un certificato in questo modo, ACM convalida la proprietà del dominio personalizzato. Per ulteriori informazioni, vedere Richiedi un certificato pubblico. Quando si crea la configurazione del dominio, utilizzare questo certificato pubblico come certificato di convalida.

Creazione di una configurazione di dominio

Puoi creare un endpoint configurabile su un dominio personalizzato utilizzando l'API. CreateDomainConfiguration Una configurazione di dominio per un dominio personalizzato è costituita dai seguenti elementi:

- `domainConfigurationName`

  un nome definito dall'utente che identica la configurazione del dominio. I nomi di configurazione del dominio che iniziano con `IoT:` sono riservati agli endpoint predefiniti e non possono essere utilizzati. Inoltre, questo valore deve essere unico per il tuo. Regione AWS
- `domainName`

  Il nome di dominio completo utilizzato dai dispositivi per la connessione. AWS IoT Core AWS IoT Core sfrutta l'estensione TLS SNI (Server Name Indication) per applicare configurazioni di dominio. I dispositivi devono utilizzare questa estensione durante la connessione e passare un nome server identico al nome di dominio specificato nella configurazione del dominio.
- `serverCertificateArns`

  L'ARN della catena di certificati del server che hai registrato con ACM. AWS IoT Core attualmente supporta solo un certificato server.
- `validationCertificateArn`

  L'ARN del certificato pubblico generato in ACM per convalidare la proprietà del dominio personalizzato. Questo argomento non è necessario se si utilizza un certificato server con firma pubblica o generato da ACM.
- `defaultAuthorizerName (optional)`

Il nome dell'autorizzazione ad hoc da utilizzare nell'endpoint.

- `allowAuthorizerOverride`

  Un valore booleano che specifica se i dispositivi possono sovrascrivere l'autorizzazione predefinita specificando un'autorizzazione differente nell'intestazione HTTP della richiesta. Questo valore è obbligatorio se viene specificato un valore per `defaultAuthorizerName`.

- `serviceType`

  AWS IoT Core attualmente supporta solo il tipo di `DATA` servizio. Quando si specifica`DATA`, AWS IoT restituisce un endpoint con un tipo di endpoint di. `iot:Data-ATS`

- `TlsConfig` (facoltativo)

  Un oggetto che specifica la configurazione TLS per un dominio. Per ulteriori informazioni, consulta ???.

- `serverCertificateConfig` (facoltativo)

  Un oggetto che specifica la configurazione del certificato del server per un dominio. Per ulteriori informazioni, consulta ???.

Il AWS CLI comando seguente crea una configurazione di dominio per iot.example.com.

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" --service-type "DATA"
--domain-name "iot.example.com" --server-certificate-arns serverCertARN --validation-
certificate-arn validationCertArn
```

> **Note**
>
> Dopo aver creato la configurazione del dominio, potrebbero essere necessari fino a 60 minuti prima che i certificati AWS IoT Core server personalizzati vengano forniti.

Per ulteriori informazioni, consulta ???.

Creazione di record DNS

Dopo aver registrato la catena di certificati server e creato la configurazione del dominio, creare un record DNS in modo che il dominio personalizzato punti a un dominio AWS IoT . Questo record

deve puntare a un AWS IoT endpoint di tipo`iot:Data-ATS`. Puoi ottenere il tuo endpoint utilizzando l'DescribeEndpointAPI.

Il AWS CLI comando seguente mostra come ottenere l'endpoint.

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

Dopo aver ottenuto l'`iot:Data-ATS`endpoint, crea un `CNAME` record dal tuo dominio personalizzato a questo AWS IoT endpoint. Se crei più domini personalizzati nello stesso dispositivo Account AWS, assegnali come alias allo stesso endpoint. `iot:Data-ATS`

Risoluzione dei problemi

Se hai problemi a connettere i dispositivi a un dominio personalizzato, assicurati che AWS IoT Core abbia accettato e applicato il certificato del tuo server. Puoi verificare di AWS IoT Core aver accettato il tuo certificato utilizzando la AWS IoT Core console o il AWS CLI.

Per utilizzare la AWS IoT Core console, vai alla pagina Configurazioni del dominio e seleziona il nome di configurazione del dominio. Nella sezione Server certificate details (Dettagli certificato server), controlla lo stato e i dettagli dello stato. Se il certificato non è valido, sostituiscilo in ACM con un certificato che soddisfa i requisiti del certificato elencati nella sezione precedente. Se il certificato ha lo stesso ARN, lo AWS IoT Core raccoglieremo e lo applicheremo automaticamente.

Per verificare lo stato del certificato utilizzando il AWS CLI, chiama l'DescribeDomainConfigurationAPI e specifica il nome di configurazione del dominio.

> **Note**
>
> Se il certificato non è valido, AWS IoT Core continuerà a fornire l'ultimo certificato valido.

È possibile verificare quale certificato viene fornito sull'endpoint utilizzando il seguente comando openssl.

`openssl s_client -connect` *`custom-domain-name`*`:8883 -showcerts -servername` *`custom-domain-name`*

Gestione delle configurazioni di dominio

Questo argomento tratta le operazioni chiave per la gestione delle risorse di configurazione del dominio. È inoltre possibile gestire i cicli di vita delle configurazioni esistenti utilizzando

quanto segue APIs: ListDomainConfigurations,, DescribeDomainConfiguratione. UpdateDomainConfigurationDeleteDomainConfiguration

In questo argomento:

- Visualizzazione delle configurazioni del dominio
- Aggiornamento delle configurazioni del dominio
- Eliminazione delle configurazioni di dominio
- Rotazione di certificati in domini personalizzati

Visualizzazione delle configurazioni del dominio

Per restituire un elenco impaginato di tutte le configurazioni di dominio presenti nel tuo, utilizza l'API. Account AWSListDomainConfigurations Puoi visualizzare i dettagli di una particolare configurazione di dominio utilizzando l'API. DescribeDomainConfiguration Questa API accetta un singolo parametro `domainConfigurationName` e restituisce i dettagli della configurazione specificata.

Esempio

Aggiornamento delle configurazioni del dominio

Per aggiornare lo stato o l'autorizzatore personalizzato della configurazione del tuo dominio, utilizza l'UpdateDomainConfigurationAPI. Solo l'utente può impostare lo stato su `ENABLED` o `DISABLED`. Se si disattiva la configurazione del dominio, i dispositivi connessi a tale dominio ricevono un errore di autenticazione. Attualmente non è possibile aggiornare il certificato del server nella configurazione del dominio. Per modificare il certificato di una configurazione di dominio, è necessario eliminarlo e ricrearlo.

Esempio

Eliminazione delle configurazioni di dominio

Prima di eliminare una configurazione di dominio, utilizza l'UpdateDomainConfigurationAPI per impostare lo stato su. `DISABLED` In questo modo è possibile evitare di eliminare accidentalmente l'endpoint. Dopo aver disabilitato la configurazione del dominio, eliminala utilizzando l'DeleteDomainConfigurationAPI. Devi impostare lo `DISABLED` stato AWS-managed domain per 7 giorni prima di poterli eliminare. Puoi impostare lo `DISABLED` stato dei domini personalizzati e quindi eliminarli contemporaneamente.

Esempio

Dopo aver eliminato una configurazione di dominio, AWS IoT Core non viene più utilizzato il certificato del server associato a quel dominio personalizzato.

Rotazione di certificati in domini personalizzati

Potrebbe essere necessario sostituire periodicamente il certificato del server con un certificato aggiornato. La tariffa con cui esegui questa operazione dipende dal periodo di validità del certificato. Se il certificato del server è stato generato utilizzando AWS Certificate Manager (ACM), è possibile impostare il certificato per il rinnovo automatico. Quando ACM rinnova il certificato, ritira AWS IoT Core automaticamente il nuovo certificato. Non hai bisogno di eseguire alcuna operazione aggiuntiva. Se il certificato del server è stato importato da un'altra fonte, può essere ruotato reimportandolo in ACM. Per informazioni sulla reimportazione dei certificati, consulta Reimport a certificate (Reimportazione di un certificato).

> **Note**
>
> AWS IoT Core raccoglie gli aggiornamenti dei certificati solo nelle seguenti condizioni.
>
> - Il nuovo certificato ha lo stesso ARN di quello precedente.
> - Il nuovo certificato ha lo stesso algoritmo di firma, nome comune o nome alternativo soggetto di quello precedente.

Configurazione delle impostazioni TLS nelle configurazioni di dominio

AWS IoT Core fornisce policy di sicurezza predefinite per personalizzare le impostazioni Transport Layer Security (TLS) per TLS 1.2 e TLS 1.3 nelle configurazioni di dominio. Una policy di sicurezza è una combinazione di protocolli TLS e delle relative crittografie che determinano i protocolli e crittografie supportati durante le negoziazioni TLS tra un client e un server. Con le politiche di sicurezza supportate, puoi gestire le impostazioni TLS dei tuoi dispositivi con maggiore flessibilità, applicare la maggior parte delle misure di up-to-date sicurezza quando colleghi nuovi dispositivi e mantenere configurazioni TLS coerenti per i dispositivi esistenti.

Nella tabella riportata di seguito vengono descritte le policy di sicurezza, le relative versioni TLS e le regioni supportate:

Nome policy di sicurezza	Supportato Regioni AWS
TSecurityPolitica Io_ _1_3_2022_10 TLS13	Tutto Regioni AWS
TSecurityPolitica Io_ _1_2_2022_10 TLS13	Tutto Regioni AWS
TSecurityPolitica Io_ _1_2_2022_10 TLS12	Tutto Regioni AWS
TSecurityPolitica Io_ _1_0_2016_01 TLS12	ap-east-1, ap-northeast-2, ap-south-1, ap-southeast-2, ca-central-1, cn-nord-1, cn-nordovest-1, eu-nordovest-1, eu-nord-1, eu-west-2, eu-west-3, me-sud-1, us-east-1, us-east-2, us-west-1
Politica TSecurity TLS12 IO_ _1_0_2015 _01	ap-northeast-1, ap-southeast-1, eu-central-1, eu-west-1, us-east-1, us-west-2

I nomi delle politiche di sicurezza AWS IoT Core includono informazioni sulla versione basate sull'anno e sul mese in cui sono state rilasciate. Se si crea una nuova configurazione di dominio, la policy di sicurezza verrà preimpostata su `IoTSecurityPolicy_TLS13_1_2_2022_10`. Per una tabella completa delle politiche di sicurezza con dettagli su protocolli, porte TCP e cifrari, vedi Criteri di sicurezza. AWS IoT Core non supporta politiche di sicurezza personalizzate. Per ulteriori informazioni, consulta ???.

Per configurare le impostazioni TLS nelle configurazioni di dominio, puoi utilizzare la AWS IoT console o il. AWS CLI

Indice

- Configurazione delle impostazioni TLS nelle configurazioni di dominio (console)
- Configurazione delle impostazioni TLS nelle configurazioni di dominio (interfaccia a riga di comando)

Configurazione delle impostazioni TLS nelle configurazioni di dominio (console)

Per configurare le impostazioni TLS utilizzando la console AWS IoT

1. Accedi a Console di gestione AWS e apri la AWS IoT console.
2. Per configurare le impostazioni TLS quando si crea una nuova configurazione di dominio, attenersi alla seguente procedura.
    1. Nel riquadro di navigazione a sinistra, scegli Configurazioni di dominio, quindi scegli Crea configurazione di dominio.
    2. Nella pagina Crea configurazione del dominio, nella sezione Impostazioni del dominio personalizzato - facoltativo, scegli una policy di sicurezza da Seleziona policy di sicurezza.
    3. Segui il widget e completa i passaggi restanti. Scegli Crea configurazione del dominio.
3. Per aggiornare le impostazioni TLS in una configurazione del dominio esistente, attenersi alla seguente procedura.
    1. Nel riquadro di navigazione a sinistra, scegli Configurazioni di dominio, quindi scegli una configurazione di dominio.
    2. Nella pagina Dettagli di configurazione del dominio, scegli Modifica. Quindi, nella sezione Impostazioni del dominio personalizzato - opzionale, in Seleziona policy di sicurezza, scegli una policy di sicurezza.
    3. Seleziona Aggiorna configurazione del dominio.

Per ulteriori informazioni, consultare Creazione di una configurazione di dominio e Gestione delle configurazioni di dominio.

Configurazione delle impostazioni TLS nelle configurazioni di dominio (interfaccia a riga di comando)

È possibile utilizzare i comandi dell'interfaccia a riga di comando create-domain-configuration e update-domain-configuration per configurare le impostazioni TLS nelle configurazioni di dominio.

1. Per specificare le impostazioni TLS utilizzando il comando dell'interfaccia a riga di comando create-domain-configuration:

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

L'output di questo comando può essere simile al seguente:

```
{
"domainConfigurationName": "test",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

Se crei una nuova configurazione di dominio senza specificare la policy di sicurezza, il valore verrà preimpostato su: `IoTSecurityPolicy_TLS13_1_2_2022_10`.

2. Per descrivere le impostazioni TLS utilizzando il comando dell'interfaccia a riga di comando describe-domain-configuration:

```
aws iot describe-domain-configuration \
    --domain-configuration-name domainConfigurationName
```

Questo comando può restituire i dettagli di configurazione del dominio che includono le impostazioni TLS come le seguenti:

```
{
 "tlsConfig": {
 "securityPolicy": "IoTSecurityPolicy_TLS13_1_2_2022_10"
 },
 "domainConfigurationStatus": "ENABLED",
 "serviceType": "DATA",
 "domainType": "AWS_MANAGED",
 "domainName": "d1234567890abcdefghij-ats.iot.us-west-2.amazonaws.com",
 "serverCertificates": [],
 "lastStatusChangeDate": 1678750928.997,
 "domainConfigurationName": "test",
 "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

3. Per aggiornare le impostazioni TLS utilizzando il comando dell'interfaccia a riga di comando update-domain-configuration:

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
```

```
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

L'output di questo comando può essere simile al seguente:

```
{
"domainConfigurationName": "test",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

4. Per aggiornare le impostazioni TLS per l'endpoint ATS, esegui il comando dell'interfaccia a riga di comando update-domain-configuration. Il nome della configurazione di dominio per l'endpoint ATS è `iot:Data-ATS`.

```
aws iot update-domain-configuration \
    --domain-configuration-name "iot:Data-ATS" \
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

L'output del comando può essere simile al seguente:

```
{
"domainConfigurationName": "iot:Data-ATS",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
iot:Data-ATS"
}
```

Per ulteriori informazioni, consulta le pagine CreateDomainConfiguration e UpdateDomainConfiguration nella Documentazione di riferimento dell'API AWS .

Configurazione del certificato server per stapling OCSP

AWS IoT Core supporta la pinzatura OCSP (Online Certificate Status Protocol) per i certificati server, nota anche come cucitura OCSP dei certificati server o pinzatura OCSP. È un meccanismo di sicurezza utilizzato per verificare lo stato di revoca sul certificato del server in un handshake Transport Layer Security (TLS). L'integrazione OCSP AWS IoT Core consente di aggiungere un ulteriore livello di verifica alla validità del certificato del server del dominio personalizzato.

È possibile abilitare l'aggiunta automatica del certificato del server OCSP AWS IoT Core per verificare la validità del certificato interrogando periodicamente il risponditore OCSP. L'impostazione

di graffatura OCSP fa parte del processo di creazione o aggiornamento di una configurazione di dominio con un dominio personalizzato. OCSP Stapling verifica continuamente lo stato di revoca sul certificato del server. Questo aiuta a verificare che i certificati che sono stati revocati dalla CA non siano più considerati attendibili dai client che si connettono ai domini personalizzati. Per ulteriori informazioni, consulta ???.

L'archiviazione OCSP dei certificati server consente di controllare lo stato di revoca in tempo reale, riduce la latenza associata al controllo dello stato di revoca e migliora la privacy e l'affidabilità delle connessioni sicure. Per ulteriori informazioni sui vantaggi dell'utilizzo della graffatura OCSP, vedere. ???

> **Note**
>
> Questa funzionalità non è disponibile in. AWS GovCloud (US) Regions

In questo argomento:

- Che cos'è OCSP?
- Come funziona la cucitura OCSP
- Abilitazione del certificato del server OCSP in AWS IoT Core
- Configurazione del certificato del server OCSP per endpoint privati in AWS IoT Core
- Note importanti per l'utilizzo del certificato del server OCSP in AWS IoT Core
- Risoluzione dei problemi relativi all'inserimento del certificato OCSP del server AWS IoT Core

Che cos'è OCSP?

L'Online Certificate Status Protocol (OCSP) aiuta a fornire lo stato di revoca di un certificato server per un handshake Transport Layer Security (TLS).

Concetti chiave

I seguenti concetti chiave forniscono dettagli sull'Online Certificate Status Protocol (OCSP).

OCSP

OCSP viene utilizzato per verificare lo stato di revoca del certificato durante l'handshake Transport Layer Security (TLS). OCSP consente la convalida in tempo reale dei certificati. Ciò conferma che il

certificato non è stato revocato o scaduto da quando è stato emesso. OCSP è anche più scalabile rispetto ai tradizionali elenchi di revoca dei certificati (). CRLs Le risposte OCSP sono più piccole e possono essere generate in modo efficiente, il che le rende più adatte per infrastrutture a chiave privata su larga scala (). PKIs

Risponditore OCSP

Un risponditore OCSP (noto anche come server OCSP) riceve e risponde alle richieste OCSP dei client che cercano di verificare lo stato di revoca dei certificati.

OCSP lato client

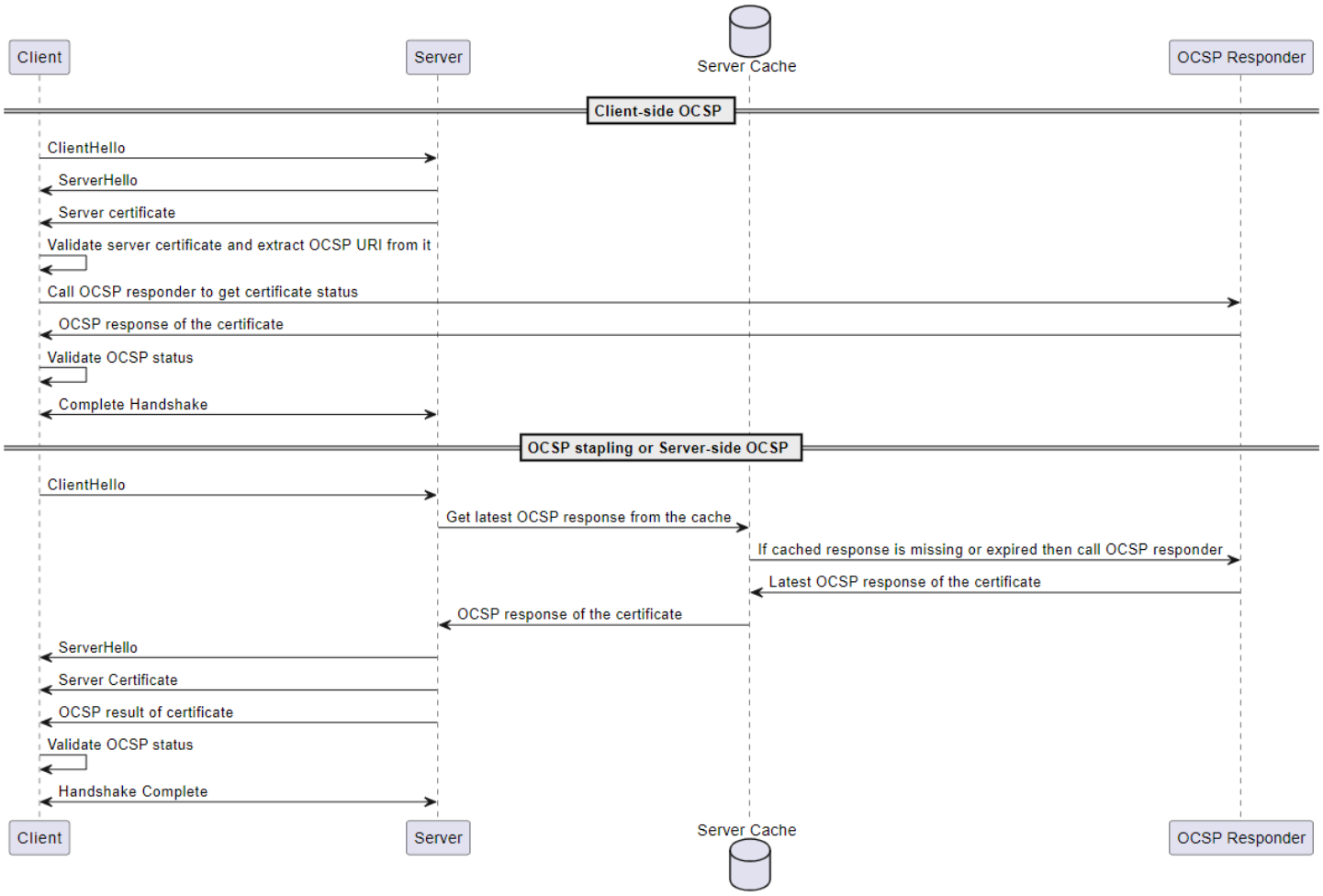
In OCSP lato client, il client utilizza OCSP per contattare un risponditore OCSP per verificare lo stato di revoca del certificato durante l'handshake TLS.

OCSP lato server

In OCSP lato server (noto anche come pinzatura OCSP), il server è abilitato (anziché il client) a effettuare la richiesta al risponditore OCSP. Il server memorizza la risposta OCSP al certificato e la restituisce al client durante l'handshake TLS.

Diagrammi OCSP

Il diagramma seguente illustra il funzionamento di OCSP lato client e OCSP lato server.



OCSP lato client

1. Il client invia un `ClientHello` messaggio per avviare l'handshake TLS con il server.
2. Il server riceve il messaggio e risponde con un messaggio. `ServerHello` Il server invia inoltre il certificato del server al client.
3. Il client convalida il certificato del server e ne estrae un URI OCSP.
4. Il client invia una richiesta di controllo della revoca del certificato al risponditore OCSP.
5. Il risponditore OCSP invia una risposta OCSP.
6. Il client convalida lo stato del certificato dalla risposta OCSP.
7. L'handshake TLS è completato.

OCSP lato server

1. Il client invia un `ClientHello` messaggio per avviare l'handshake TLS con il server.

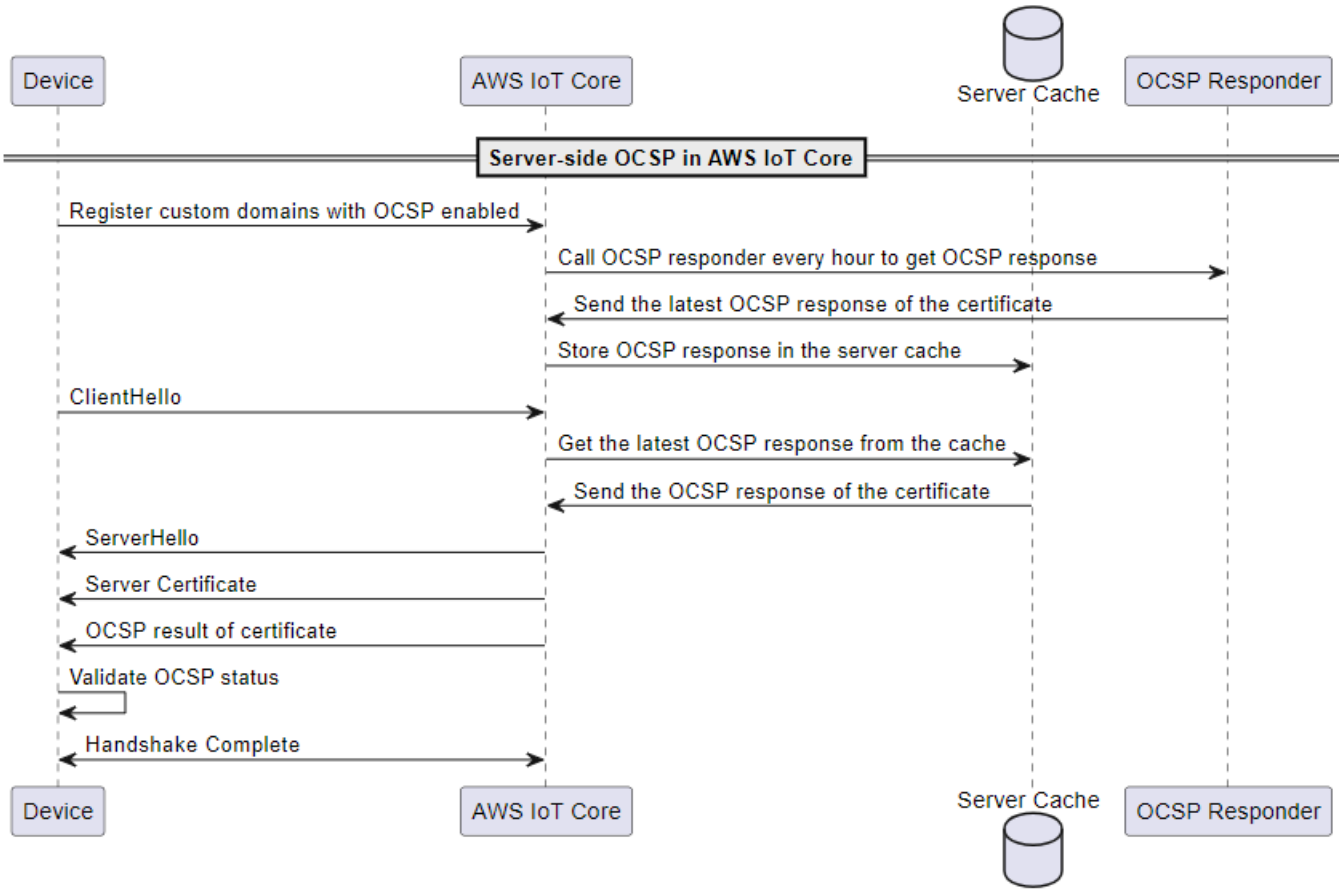
2. Il server riceve il messaggio e riceve l'ultima risposta OCSP memorizzata nella cache. Se la risposta memorizzata nella cache è mancante o è scaduta, il server chiamerà il risponditore OCSP per verificare lo stato del certificato.
3. Il risponditore OCSP invia una risposta OCSP al server.
4. Il server invia un messaggio. `ServerHello` Il server invia inoltre il certificato del server e lo stato del certificato al client.
5. Il client convalida lo stato del certificato OCSP.
6. L'handshake TLS è completato.

Come funziona la cucitura OCSP

Lo stapling OCSP viene utilizzato durante l'handshake TLS tra il client e il server per verificare lo stato di revoca del certificato del server. Il server invia la richiesta OCSP al risponditore OCSP e archivia le risposte OCSP ai certificati restituiti al client. Facendo in modo che il server effettui la richiesta al risponditore OCSP, le risposte possono essere memorizzate nella cache e quindi utilizzate più volte per molti client.

Come funziona la cucitura OCSP in AWS IoT Core

Il diagramma seguente mostra come funziona la graffatura OCSP sul lato server. AWS IoT Core



1. Il dispositivo deve essere registrato con domini personalizzati con la graffatura OCSP abilitata.
2. AWS IoT Core chiama il risponditore OCSP ogni ora per ottenere lo stato del certificato.
3. Il risponditore OCSP riceve la richiesta, invia la risposta OCSP più recente e archivia la risposta OCSP nella cache.
4. Il dispositivo invia un `ClientHello` messaggio con cui avviare l'handshake TLS. AWS IoT Core
5. AWS IoT Core ottiene la risposta OCSP più recente dalla cache del server, che risponde con una risposta OCSP del certificato.
6. Il server invia un `ServerHello` messaggio al dispositivo. Il server invia inoltre il certificato del server e lo stato del certificato al client.
7. Il dispositivo convalida lo stato del certificato OCSP.
8. L'handshake TLS è completato.

Vantaggi dell'utilizzo della pinzatura OCSP rispetto ai controlli OCSP lato client

Alcuni vantaggi dell'utilizzo della graffatura OCSP dei certificati server includono quanto segue:

Migliore privacy

Senza l'utilizzo di caratteri OCSP, il dispositivo del client può esporre le informazioni a risponditori OCSP di terze parti, compromettendo potenzialmente la privacy degli utenti. La graffatura OCSP mitiga questo problema facendo in modo che il server ottenga la risposta OCSP e la invii direttamente al client.

Affidabilità migliorata

La pinzatura OCSP può migliorare l'affidabilità delle connessioni sicure perché riduce il rischio di interruzioni del server OCSP. Quando le risposte OCSP vengono pinzate, il server include la risposta più recente con il certificato. In questo modo i client hanno accesso allo stato di revoca anche se il risponditore OCSP è temporaneamente non disponibile. Lo stapling OCSP aiuta a mitigare questi problemi perché il server recupera periodicamente le risposte OCSP e include le risposte memorizzate nella cache nell'handshake TLS. Ciò riduce la dipendenza dalla disponibilità in tempo reale dei risponditori OCSP.

Carico ridotto del server

Lo stapling OCSP alleggerisce al server l'onere di rispondere alle richieste OCSP dei risponditori OCSP. Questo può aiutare a distribuire il carico in modo più uniforme, rendendo il processo di convalida dei certificati più efficiente e scalabile.

Latenza ridotta

Lo stapling OCSP riduce la latenza associata al controllo dello stato di revoca di un certificato durante l'handshake TLS. Invece di dover interrogare separatamente un server OCSP, il server invia la richiesta e allega la risposta OCSP al certificato del server durante l'handshake.

Abilitazione del certificato del server OCSP in AWS IoT Core

Per abilitare l'inserimento del certificato del server OCSP AWS IoT Core, crea una configurazione di dominio per un dominio personalizzato o aggiorna una configurazione di dominio personalizzata esistente. Per informazioni generali sulla creazione di una configurazione di dominio con un dominio personalizzato, vedere. ???

Utilizzare le seguenti istruzioni per abilitare la pinzatura del server OCSP utilizzando Console di gestione AWS o. AWS CLI

Console

Per abilitare la graffatura OCSP dei certificati del server utilizzando la console: AWS IoT

1. Nel menu di navigazione, scegli Impostazioni, quindi scegli Crea configurazione di dominio o scegli una configurazione di dominio esistente per un dominio personalizzato.
2. Se scegli di creare una nuova configurazione di dominio nel passaggio precedente, vedrai la pagina Crea configurazione del dominio. Nella sezione Proprietà di configurazione del dominio, scegli Dominio personalizzato. Inserisci le informazioni per creare una configurazione del dominio.

   Se scegli di aggiornare una configurazione di dominio esistente per un dominio personalizzato, vedrai la pagina dei dettagli della configurazione del dominio. Scegli Modifica.
3. Per abilitare lo stampaggio OCSP del server OCSP, scegli Abilita la graffatura OCSP dei certificati del server nella sottosezione Configurazioni dei certificati del server.
4. Scegli Crea configurazione di dominio o Aggiorna configurazione di dominio.

AWS CLI

Per abilitare la memorizzazione OCSP del certificato del server utilizzando: AWS CLI

1. Se si crea una nuova configurazione di dominio per un dominio personalizzato, il comando per abilitare lo stapling del server OCSP può essere simile al seguente:

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true|false"
```

2. Se si aggiorna una configurazione di dominio esistente per un dominio personalizzato, il comando per abilitare lo stapling del server OCSP può essere simile al seguente:

```
aws iot update-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
```

```
        --server-certificate-config "enableOCSPCheck=true|false"
```

Per ulteriori informazioni, consulta CreateDomainConfiguratione consulta l' AWS IoT API UpdateDomainConfigurationReference.

Configurazione del certificato del server OCSP per endpoint privati in AWS IoT Core

OCSP for private endpoint ti consente di utilizzare le tue risorse OCSP private all'interno di Amazon Virtual Private Cloud (Amazon VPC) per le operazioni. AWS IoT Core Il processo prevede l'impostazione di una funzione Lambda che funge da risponditore OCSP. La funzione Lambda potrebbe utilizzare le risorse OCSP private per creare risposte OCSP da utilizzare. AWS IoT Core

funzione Lambda

Prima di configurare il server OCSP per un endpoint privato, create una funzione Lambda che funga da risponditore OCSP (Online Certificate Status Protocol) conforme a Request for Comments (RFC) 6960, che supporta le risposte OCSP di base. La funzione Lambda accetta una codifica base64 della richiesta OCSP nel formato Distinguished Encoding Rules (DER). La risposta della funzione Lambda è anche una risposta OCSP con codifica base64 nel formato DER. La dimensione della risposta non deve superare i 4 kilobyte (KB). La funzione Lambda deve essere nella stessa configurazione Account AWS e nella stessa Regione AWS configurazione del dominio. Di seguito sono riportati alcuni esempi di funzioni Lambda.

Esempi di funzioni Lambda

JavaScript

```
import * as pkijs from 'pkijs';
console.log('Loading function');

export const handler = async (event, context) => {
    const requestBytes = decodeBase64(event);
    const ocspRequest = pkijs.OCSPRequest.fromBER(requestBytes);

    console.log("Here is a better look at the OCSP request");
    console.log(ocspRequest.toJSON());

    const ocspResponse = getOcspResponse();

    console.log("Here is a better look at the OCSP response");
```

```
    console.log(ocspResponse.toJSON());

   const responseBytes = ocspResponse.toSchema().toBER();
   return encodeBase64(responseBytes);
};

function getOcspResponse() {
    const responseString = "MIIC/
woBAKCCAvgwggL0BgkrBgEFBQcwAQEEggLlMIIC4TCByqFkMGIxJzAlBgNVBAoMHlJpY2hhcmQncyBEaXNjb3VudCBMY
p5w7WOtPjp3otNtVgIBAYAAGA8yMDI0MDQyMzE4NTMyNVowDQYJKoZIhvcNAQELBQADggIBAJFRyjDAHfazNejo704Ra
+s82R1spDarr3k7Pzkod9jJhwsZ2YgushlS4Npfe4lHCdwFyZR75WXrW55aXFddy03KLz01ZLNYyxkleW3f5dgrUcRU3
DEBiyS7ZsyhKo6igWU/SY7YMSKgwBvFsqSDcOa/hRYQkxWKWJ19gcz8CIkWN7NvfIxCs6VrAdzEJwmE7y3v
+jdfhxW9JmI4xStE4K0tAR9vVOOfKs7NvxXj7oc9pCSG60xl96kaEE6PaY1YsfNTsKQ7pyCJ0s7/2q
+ieZ4AtNyzw1XBadPzPJNv6E0LvI24yQZqN5wACvtut5prMMRxAHbOy
+abLZR58wloFSELtGJ7UD96LFv1GgtC5s
+2QlzPc4bEEof7Lo1EISt3j2ibNch8LxhqTQ4ufrbhsMkpSOTFYEJVMJF6aKj/OGXBUUqgc0Jx6jjJXNQd
+l5KCY9pQFeb/wVUYC6mYqZOkNNMMJxPbHHbFnqb68yO+g5BE9011N44YXoPVJYoXxBLFX+OpRu9cqPkT9/
vlkKd+SYXQknwZ81agKzhf1HsBKabtJwNVMlBKaI8g5UGa7Bxi6ewH3ezdWiERRUK7F56OM53wto/";
    const responseBytes = decodeBase64(responseString);
    return pkijs.OCSPResponse.fromBER(responseBytes);
}

function decodeBase64(input) {
    const binaryString = atob(input);

    const byteArray = new Uint8Array(binaryString.length);
    for (var i = 0; i < binaryString.length; i++) {
        byteArray[i] = binaryString.charCodeAt(i);
    }

    return byteArray.buffer;
}

function encodeBase64(buffer) {
    var binary = '';
    const bytes = new Uint8Array( buffer );
    const len = bytes.byteLength;

    for (var i = 0; i < len; i++) {
        binary += String.fromCharCode( bytes[ i ] );
    }

    return btoa(binary);
```

```
}
```

Java

```
package com.example.ocsp.responder;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import org.bouncycastle.cert.ocsp.OCSPReq;
import org.bouncycastle.cert.ocsp.OCSPResp;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Base64;

public class LambdaResponderApplication implements RequestHandler<String, String> {
    @Override
    public String handleRequest(final String input, final Context context) {
        LambdaLogger logger = context.getLogger();

        byte[] decodedInput = Base64.getDecoder().decode(input);

        OCSPReq req;
        try {
            req = new OCSPReq(decodedInput);
        } catch (IOException e) {
            logger.log("Got an IOException creating the OCSP request: " +
 e.getMessage());
            throw new RuntimeException(e);
        }

        try {
            OCSPResp response = businessLogic.getMyResponse();
            String toReturn =
 Base64.getEncoder().encodeToString(response.getEncoded());
            return toReturn;
        } catch (Exception e) {
            logger.log("Got an exception creating the response: " + e.getMessage());
            return "";
        }
    }
}
```

Autorizzazione AWS IoT a richiamare la funzione Lambda

Nel processo di creazione della configurazione del dominio con un risponditore Lambda OCSP, è necessario concedere l' AWS IoT autorizzazione a richiamare la funzione Lambda dopo la creazione della funzione. Per concedere l'autorizzazione, è possibile utilizzare il comando dell'interfaccia a riga di comando add-permission.

Concedi l'autorizzazione alla tua funzione Lambda utilizzando AWS CLI

1. Una volta inseriti i valori, inserisci il comando seguente. Attenzione: il valore `statement-id` deve essere univoco. Sostituisci *Id-1234* con il valore esatto che hai, altrimenti potresti ricevere un `ResourceConflictException` errore.

   ```
   aws lambda add-permission  \
   --function-name "ocsp-function" \
   --principal "iot.amazonaws.com" \
   --action "lambda:InvokeFunction" \
   --statement-id "Id-1234" \
   --source-arn arn:aws:iot:us-east-1:123456789012:domainconfiguration/<domain-config-name>/*
   --source-account 123456789012
   ```

   La configurazione del dominio IoT ARNs seguirà lo schema seguente. Il suffisso generato dal servizio non sarà noto prima della creazione, pertanto è necessario sostituire il suffisso con un. * È possibile aggiornare l'autorizzazione una volta creata la configurazione del dominio e conoscere l'ARN esatto.

   `arn:aws:iot:use-east-1:123456789012:domainconfiguration/domain-config-name/service-generated-suffix`

2. Se il comando viene completato correttamente, restituisce un'istruzione di autorizzazione come questa. È possibile passare alla sezione successiva per configurare lo stapling OCSP per endpoint privati.

   ```
   {
       "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
   \":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
   \",\"Resource\":\"arn:aws:lambda:us-east-1:123456789012:function:ocsp-
   function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:iot:us-
   east-1:123456789012:domainconfiguration/domain-config-name/*\"}}}"
   }
   ```

Se il comando non ha esito positivo, restituisce un errore come questo. Dovrai esaminare e correggere l'errore prima di continuare.

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
mission on resource: arn:aws:lambda:us-east-1:123456789012:function:ocsp-function
```

Configurazione dello stapling OCSP del server per endpoint privati

Console

Per configurare la graffatura OCSP del certificato del server utilizzando la console: AWS IoT

1. Dal menu di navigazione, scegli Impostazioni, quindi scegli Crea configurazione di dominio o scegli una configurazione di dominio esistente per un dominio personalizzato.
2. Se scegli di creare una nuova configurazione di dominio nel passaggio precedente, vedrai la pagina Crea configurazione del dominio. Nella sezione Proprietà di configurazione del dominio, scegli Dominio personalizzato. Inserisci le informazioni per creare una configurazione del dominio.

   Se scegli di aggiornare una configurazione di dominio esistente per un dominio personalizzato, vedrai la pagina dei dettagli della configurazione del dominio. Scegli Modifica.
3. Per abilitare lo stampaggio OCSP del server OCSP, scegli Abilita la graffatura OCSP dei certificati del server nella sottosezione Configurazioni dei certificati del server.
4. Scegli Crea configurazione di dominio o Aggiorna configurazione di dominio.

AWS CLI

Per configurare la graffatura OCSP del certificato del server utilizzando: AWS CLI

1. Se crei una nuova configurazione di dominio per un dominio personalizzato, il comando per configurare il certificato del server OCSP per gli endpoint privati può essere simile al seguente:

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
```

```
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true,
 ocspAuthorizedResponderArn=arn:aws:acm:us-
east-1:123456789012:certificate/certificate_ID, ocspLambdaArn=arn:aws:lambda:us-
east-1:123456789012:function:my-function"
```

2. Se si aggiorna una configurazione di dominio esistente per un dominio personalizzato, il comando per configurare il certificato del server OCSP per gli endpoint privati può essere simile al seguente:

```
aws iot update-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true,
 ocspAuthorizedResponderArn=arn:aws:acm:us-
east-1:123456789012:certificate/certificate_ID, ocspLambdaArn=arn:aws:lambda:us-
east-1:123456789012:function:my-function"
```

abilitare OCSPCheck

Questo è un valore booleano che indica se il controllo di graffatura OCSP del server è abilitato o meno. Per abilitare la graffatura OCSP del certificato del server, questo valore deve essere vero.

ocspAuthorizedResponderArn

Si tratta di un valore stringa dell'Amazon Resource Name (ARN) per un certificato X.509 memorizzato in AWS Certificate Manager (ACM). Se fornito, AWS IoT Core utilizzerà questo certificato per convalidare la firma della risposta OCSP ricevuta. Se non fornito, AWS IoT Core utilizzerà il certificato di emissione per convalidare le risposte. Il certificato deve essere nella Regione AWS stessa Account AWS configurazione del dominio. Per ulteriori informazioni su come registrare il certificato di risponditore autorizzato, consulta Importare certificati in AWS Certificate Manager.

ocspLambdaArn

Si tratta di un valore stringa dell'Amazon Resource Name (ARN) per una funzione Lambda che funge da risponditore conforme a Request for Comments (RFC) 6960 (OCSP), supportando le risposte OCSP di base. La funzione Lambda accetta una codifica base64 della richiesta OCSP

codificata utilizzando il formato DER. La risposta della funzione Lambda è anche una risposta OCSP con codifica base64 nel formato DER. La dimensione della risposta non deve superare i 4 kilobyte (KB). La funzione Lambda deve essere nella stessa configurazione Account AWS e nella stessa Regione AWS configurazione del dominio.

Per ulteriori informazioni, consulta CreateDomainConfiguratione utilizza UpdateDomainConfigurationl' AWS IoT API Reference.

Note importanti per l'utilizzo del certificato del server OCSP in AWS IoT Core

Quando utilizzi il certificato del server OCSP in AWS IoT Core, tieni presente quanto segue:

1. AWS IoT Core supporta solo i risponditori OCSP raggiungibili tramite indirizzi pubblici. IPv4
2. La funzionalità di spillatura OCSP AWS IoT Core non supporta i risponditori autorizzati. Tutte le risposte OCSP devono essere firmate dalla CA che ha firmato il certificato e la CA deve far parte della catena di certificati del dominio personalizzato.
3. La funzionalità di base OCSP AWS IoT Core non supporta i domini personalizzati creati utilizzando certificati autofirmati.
4. AWS IoT Core chiama un risponditore OCSP ogni ora e memorizza la risposta nella cache. Se la chiamata al risponditore fallisce, AWS IoT Core memorizzerà la risposta valida più recente.
5. Se non `nextUpdateTime` è più valida, AWS IoT Core rimuoverà la risposta dalla cache e l'handshake TLS non includerà i dati di risposta OCSP fino alla successiva chiamata riuscita al risponditore OCSP. Ciò può accadere quando la risposta memorizzata nella cache è scaduta prima che il server riceva una risposta valida dal risponditore OCSP. Il valore di `nextUpdateTime` suggerisce che la risposta OCSP sarà valida fino a quel momento. Per ulteriori informazioni su `nextUpdateTime`, consultare ???.
6. A volte, AWS IoT Core non riesce a ricevere la risposta OCSP o rimuove la risposta OCSP esistente perché è scaduta. Se si verificano situazioni come queste, AWS IoT Core continuerà a utilizzare il certificato del server fornito dal dominio personalizzato senza la risposta OCSP.
7. La dimensione della risposta OCSP non può superare i 4 KiB.

Risoluzione dei problemi relativi all'inserimento del certificato OCSP del server AWS IoT Core

AWS IoT Core emette la `RetrieveOCSPStapleData.Success` metrica e le voci di registro su. `RetrieveOCSPStapleData` CloudWatch La metrica e le voci di registro possono aiutare a rilevare i problemi relativi al recupero delle risposte OCSP. Per ulteriori informazioni, consultare ??? e ???.

Connect agli AWS IoT endpoint FIPS

AWS IoT fornisce endpoint che supportano il Federal Information Processing Standard (FIPS) 140-2. Gli endpoint conformi a FIPS sono diversi dagli endpoint standard. AWS Per interagire con AWS IoT in modo conforme a FIPS, è necessario utilizzare gli endpoint descritti di seguito con il client conforme FIPS. La AWS IoT console non è conforme a FIPS.

Le sezioni seguenti descrivono come accedere agli AWS IoT endpoint conformi a FIPS utilizzando l'API REST, un SDK o il. AWS CLI

Argomenti

- Endpoint del piano di controllo AWS IoT Core
- AWS IoT Core- Endpoint del piano dati
- AWS IoT Core- endpoint del fornitore di credenziali
- AWS IoT Device Management - endpoint dati di processo
- AWS IoT Device Management - endpoint Fleet Hub
- AWS IoT Device Management - Endpoint di tunneling sicuro
- AWS IoT Device Management - Endpoint di integrazioni gestite

Endpoint del piano di controllo AWS IoT Core

Gli endpoint del piano di controllo AWS IoT Core conformi a FIPS che supportano le operazioni AWS IoT e relativi Comandi CLI sono elencati in Endpoint FIPS per servizio. In Endpoint FIPS per servizio, trova il servizio di piano di controllo AWS IoT Core e cerca l'endpoint per la tua Regione AWS.

Per utilizzare l'endpoint conforme a FIPS quando accedi alle AWS IoToperazioni, utilizza l' AWS SDK o l'API REST con l'endpoint adatto al tuo. Regione AWS

Per utilizzare l'endpoint conforme FIPS durante l'esecuzione di Comandi CLI aws iot, aggiungi il parametro--endpoint con l'endpoint appropriato per la tua Regione AWS al comando.

AWS IoT Core- Endpoint del piano dati

Gli endpoint del piano dati AWS IoT Core conformi FIPS sono elencati in Endpoint FIPS per servizio. In Endpoint FIPS per servizio, trova il servizio piano dati AWS IoT Core e cerca l'endpoint per la tua Regione AWS.

Puoi utilizzare l'endpoint conforme a FIPS Regione AWS con un client conforme a FIPS utilizzando AWS IoT Device SDK e fornendo l'endpoint alla funzione di connessione dell'SDK al posto dell'endpoint predefinito del tuo account, l'endpoint del piano dati.AWS IoT Core La funzione di AWS IoT connessione è specifica del Device SDK. Per un esempio di funzione di connessione, vedete la funzione Connection in AWS IoT Device SDK for Python.

> **Note**
>
> AWS IoT non supporta Account AWS-specific AWS IoT Core- endpoint del piano dati conformi a FIPS. Le funzionalità di servizio che richiedono un endpoint Account AWS specifico nella Server Name Indication (SNI) non possono essere utilizzate. Gli endpoint AWS IoT Core - piano dati conformi a FIPS non possono supportare certificati di registrazione multi-account, domini personalizzati, autorizzazioni ad hoc ed endpoint configurabili (incluse policy TLS supportate).

AWS IoT Core- endpoint del fornitore di credenziali

Gli endpoint del fornitore di credenziali conformi AWS IoT Core a FIPS sono elencati in FIPS Endpoints by Service. In FIPS Endpoints by Service, trova il servizio AWS IoT Core - credenziali provider e cerca l'endpoint adatto al tuo. Regione AWS

> **Note**
>
> AWS IoT non supporta endpoint di provider Account AWS di credenziali specifici AWS IoT Core conformi allo standard FIPS. Le funzionalità di servizio che richiedono un endpoint Account AWS specifico nella Server Name Indication (SNI) non possono essere utilizzate. Conforme a FIPS AWS IoT Core: gli endpoint dei provider di credenziali non possono supportare certificati di registrazione multiaccount, domini personalizzati, autorizzatori personalizzati ed endpoint configurabili (incluse le politiche TLS supportate).

AWS IoT Device Management - endpoint dati di processo

Gli endpoint di AWS IoT Device Management - dati di processi conformi a FIPS sono elencati in Endpoint FIPS per servizio. In Endpoint FIPS per servizio, trova il servizio AWS IoT Device Management - dati di processi e cerca l'endpoint per il tuo Regione AWS.

Per utilizzare l'endpoint dei dati di processi AWS IoT Device Management conforme a FIPS quando esegui i comandi di aws iot-jobs-data CLI, aggiungi il parametro --endpoint con l'endpoint appropriato per il Regione AWS al comando. Puoi anche utilizzare l'API REST con questo endpoint.

Ti `Data-ATS` consigliamo di `iot:Jobs` utilizzare invece di. `iot:Data-ATS`supporta endpoint dual-stack (IPv4 e IPv6) mentre supporta solo. `iot:Jobs` IPv4

Puoi utilizzare l'endpoint conforme a FIPS per te Regione AWS con un client conforme a FIPS utilizzando AWS IoT Device SDK e fornendo l'endpoint alla funzione di connessione dell'SDK al posto dell'endpoint predefinito del tuo account, Jobs Data Endpoint.AWS IoT Device Management La funzione di connessione è specifica per l'SDK di dispositivo AWS IoT . Per un esempio di funzione di connessione, vedete la funzione Connection in AWS IoT Device SDK for Python.

AWS IoT Device Management - endpoint Fleet Hub

Gli endpoint FIPS AWS IoT Device Management- Fleet Hub da utilizzare con i comandiCLI di Fleet Hub for Device AWS IoT Management sono elencati in FIPS Endpoints by Service. in Endpoint FIPS per servizio, trova il servizio AWS IoT Device Management - Fleet Hub e cerca l'endpoint per la tua Regione AWS.

Per utilizzare l'endpoint Fleet Hub conforme AWS IoT Device Management a FIPS quando esegui i comandi aws iotfleethub CLI, aggiungi al comando il --endpoint parametro con l'endpoint appropriato per il tuo. Regione AWS Puoi anche utilizzare l'API REST con questo endpoint.

AWS IoT Device Management - Endpoint di tunneling sicuro

Gli endpoint di tunneling sicuro AWS IoT Device Management conformi a FIPS per l'API di tunneling sicuro AWS IoT e i corrispondenti comandi della CLI sono elencati nella sezione Endpoint FIPS per servizio. In Endpoint FIPS per servizio, trova il servizio AWS IoT Device Management - tunneling sicuro e cerca l'endpoint per la tua Regione AWS.

Per utilizzare l'endpoint di tunneling sicuro AWS IoT Device Management conforme a FIPS quando esegui i comandi di aws iotsecuretunneling CLI, aggiungi il parametro --endpoint con l'endpoint appropriato per il Regione AWS al comando. Puoi anche utilizzare l'API REST con questo endpoint.

AWS IoT Device Management - Endpoint di integrazioni gestite

Gli endpoint del piano di controllo conformi a FIPS che supportano le operazioni di integrazione gestita e i relativi AWS CLI comandi sono elencati in FIPS Endpoints by Service. In FIPS Endpoints

by Service, trova il servizio AWS IoT Device Management - Integrazioni gestite e cerca l'endpoint adatto al tuo. Regione AWS

Per utilizzare l'endpoint conforme a FIPS quando accedi alle operazioni di integrazione gestite, utilizza l' AWS SDK o l'API REST con l'endpoint adatto alle tue esigenze. Regione AWS

Per utilizzare l'endpoint conforme a FIPS quando esegui comandi CLI di integrazioni gestite, aggiungi al comando il --endpoint parametro con l'endpoint appropriato per il tuo. Regione AWS

Gestione dei dispositivi con AWS IoT

AWS IoT fornisce un registro che ti aiuta a gestire le cose. Una cosa è una rappresentazione di un dispositivo o di un'entità logica specifici. Può trattarsi di un dispositivo fisico o un sensore, ad esempio una lampadina o un interruttore su un muro. Può anche essere un'entità logica come un'istanza di un'applicazione o un'entità fisica che non si connette AWS IoT ma è collegata ad altri dispositivi che lo fanno (ad esempio, un'auto dotata di sensori del motore o un pannello di controllo).

Le informazioni su un oggetto vengono archiviate nel registro come dati JSON. Di seguito è illustrato un esempio di oggetto:

```
{
     "version": 3,
    "thingName": "MyLightBulb",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
}
```

Gli oggetti sono identificati da un nome. Gli oggetti possono anche avere attributi, che sono coppie nome–valore che è possibile usare per archiviare le informazioni sull'oggetto, ad esempio il numero di serie o il produttore.

Un tipico caso d'uso di un dispositivo prevede l'uso del nome dell'oggetto come ID client MQTT predefinito. Sebbene non imponiamo una mappatura tra il nome del registro di un oggetto e l'utilizzo del client MQTT IDs, dei certificati o dello stato ombra, consigliamo di scegliere un nome di oggetto e utilizzarlo come ID client MQTT sia per il registro che per il servizio Device Shadow. In questo modo, puoi ottenere organizzazione e comodità per il parco istanze IoT senza rinunciare alla flessibilità del modello di certificati dei dispositivi sottostante o delle copie shadow.

Non è necessario creare un oggetto nel registro per connettere un dispositivo ad AWS IoT. L'aggiunta di oggetti al registro permette di semplificare le attività di gestione e ricerca di dispositivi.

I dati del registro degli oggetti (inclusi gli attributi, i tipi di oggetti e le appartenenze ai gruppi) possono anche essere recuperati dinamicamente nel motore delle regole per utilizzarli nell'elaborazione e nel routing dei messaggi. Per ulteriori informazioni, consulta `get_registry_data`.

Gestire gli elementi con il registro

Si utilizza la AWS IoT console, AWS IoT l'API o il AWS CLI per interagire con il registro. Nelle seguenti sezioni viene illustrato come usare l'interfaccia a riga di comando per lavorare con il registro.

Quando nomini gli oggetti:

- Non utilizzare informazioni di identificazione personale nel nome dell'oggetto. Il nome dell'oggetto può essere visualizzato nelle comunicazioni e nei report non crittografati.

Argomenti

- Crea un oggetto
- Visualizzazione dell'elenco di oggetti
- Descrizione di oggetti
- Aggiornamento di un oggetto
- Eliminazione di un oggetto
- Collegamento di un'entità principale a un oggetto
- Elenca le cose associate a un preside
- Elenca i principi associati a una cosa
- Elenca le cose associate a una V2 principale
- Elenca i principi associati a una cosa V2
- Scollegamento di un'entità principale da un oggetto

Crea un oggetto

Il comando seguente mostra come utilizzare il AWS IoT CreateThing comando dalla CLI per creare un oggetto. Non puoi cambiare il nome di un oggetto dopo averlo creato. Per cambiare il nome di un oggetto, creane uno nuovo, assegnagli il nuovo nome e poi elimina quello vecchio.

```
$ aws iot create-thing \
    --thing-type-name "MyLightBulb" \
    --attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Il comando CreateThing visualizza il nome e il nome della risorsa Amazon (ARN) del nuovo oggetto.

```
{
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingName": "MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678"
}
```

> **Note**
>
> Non è consigliabile utilizzare informazioni di identificazione personale nei nomi degli oggetti.

Per ulteriori informazioni, consulta create-thing nella Guida di riferimento ai comandi dell' AWS CLI .

Visualizzazione dell'elenco di oggetti

Usa il comando ListThings per elencare tutti gli oggetti nell'account:

```
$ aws iot list-things
```

```
{
    "things": [
       {
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyLightBulb"
        },
        {
            "attributes": {
                "numOfStates":"3"
             },
            "version": 11,
            "thingName": "MyWallSwitch"
        }
    ]
}
```

Usa il comando ListThings per cercare tutti gli oggetti di un tipo di oggetto specifico:

```
$  aws iot list-things --thing-type-name "LightBulb"
```

```
{
    "things": [
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyRGBLight"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MySecondLightBulb"
        }
    ]
}
```

Usa il comando ListThings per cercare tutti gli oggetti che hanno un attributo con un valore specifico. Questo comando ricerca fino a tre attributi.

```
$  aws iot list-things --attribute-name "wattage" --attribute-value "75"
```

```
{
    "things": [
        {
            "thingTypeName": "StopLight",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3,
            "thingName": "MyLightBulb"
```

```
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyRGBLight"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MySecondLightBulb"
        }
    ]
}
```

Per ulteriori informazioni, consulta list-things nella Guida di riferimento ai comandi dell' AWS CLI .

Descrizione di oggetti

Utilizza il comando DescribeThing per visualizzare maggiori informazioni su un oggetto:

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "version": 3,
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "StopLight",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
}
```

Puoi anche accedere a questa API all'interno del motore delle regole utilizzando la funzione `get_registry_data()` inline. È possibile utilizzare questa funzione per accedere e utilizzare dinamicamente le informazioni del registro degli oggetti (inclusi attributi, tipi di oggetti e appartenenze ai gruppi) chiamando e `ListThingGroupsForThing` APIs direttamente all'interno delle AWS IoT regole, abilitando l'elaborazione `DescribeThing` e il routing dei messaggi in tempo reale in base ai dati del registro del dispositivo. Per ulteriori informazioni, consulta `get_registry_data`.

Per ulteriori informazioni, vedere describe-thing dal Command Reference. AWS CLI

Aggiornamento di un oggetto

Usa il comando UpdateThing per aggiornare un oggetto. Questo comando aggiorna solo gli attributi dell'oggetto. Non puoi cambiare il nome di un oggetto. Per cambiare il nome di un oggetto, creane uno nuovo, assegnagli un nuovo nome e quindi elimina quello vecchio.

```
$ aws iot update-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\":
 {\"wattage\":\"150\", \"model\":\"456\"}}"
```

Il comando UpdateThing non produce output. Usa il comando DescribeThing per visualizzare il risultato:

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "attributes": {
        "model": "456",
        "wattage": "150"
    },
    "version": 2,
    "thingName": "MyLightBulb"
}
```

Per ulteriori informazioni, consulta update-thing nella Guida di riferimento ai comandi dell' AWS CLI .

Eliminazione di un oggetto

Usa il comando DeleteThing per eliminare un oggetto:

```
$ aws iot delete-thing --thing-name "MyThing"
```

Questo comando ha esito positivo senza alcun errore se l'eliminazione va a buon fine oppure se specifichi un oggetto che non esiste.

Per ulteriori informazioni, consulta delete-thing nella Guida di riferimento ai comandi dell' AWS CLI .

Collegamento di un'entità principale a un oggetto

Un dispositivo fisico può utilizzare un dispositivo principale con cui comunicare AWS IoT. Un principale può essere un certificato X.509 o un ID Amazon Cognito. Puoi associare un certificato o un ID Amazon Cognito all'elemento nel registro che rappresenta il tuo dispositivo, eseguendo il attach-thing-principalcomando.

Per allegare un certificato o un ID Amazon Cognito al tuo dispositivo, usa il attach-thing-principalcomando:

```
$ aws iot attach-thing-principal \
    --thing-name "MyLightBulb1" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

Per allegare un certificato a un oggetto con un tipo di allegato (allegato esclusivo o allegato non esclusivo), usa il attach-thing-principalcomando e specifica un tipo nel `--thing-principal-type` campo. Un allegato esclusivo significa che il tuo elemento IoT è l'unico elemento allegato al certificato e questo certificato non può essere associato a nessun altro elemento. Un allegato non esclusivo significa che il tuo oggetto IoT è allegato al certificato e questo certificato può essere associato ad altri elementi. Per ulteriori informazioni, consulta ???.

> **Note**
>
> Per ??? questa funzionalità, è possibile utilizzare solo il certificato X.509 come principale.

```
$ aws iot attach-thing-principal \
    --thing-name "MyLightBulb2" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847" \
    --thing-principal-type "EXCLUSIVE_THING"
```

Se l'allegato ha esito positivo, il AttachThingPrincipal comando non produce alcun output. Per descrivere l'allegato, utilizzare il list-thing-principals-v comando 2 CLI.

Per ulteriori informazioni, consulta AttachThingPrincipall'AWS IoT Core API Reference.

Elenca le cose associate a un preside

Per elencare gli elementi associati al principale specificato, esegui il `list-principal-things`comando. Nota che questo comando non elenca il tipo di allegato tra l'oggetto e il certificato. Per elencare il tipo di allegato, utilizzare il `list-principal-things-v2`comando. Per ulteriori informazioni, consulta ???.

```
$ aws iot list-principal-things \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

L'output può essere simile al seguente.

```
{
    "things": [
        "MyLightBulb1",
        "MyLightBulb2"
    ]
}
```

Per ulteriori informazioni, consulta ListPrincipalThingsl'AWS IoT Core API Reference.

Elenca i principi associati a una cosa

Per elencare i principali associati all'oggetto specificato, esegui il `list-thing-principals`comando. Nota che questo comando non elenca il tipo di allegato tra l'oggetto e il certificato. Per elencare il tipo di allegato, utilizzare il `list-thing-principals-v2`comando. Per ulteriori informazioni, consulta ???.

```
$ aws iot list-thing-principals \
    --thing-name "MyLightBulb1"
```

L'output può essere simile al seguente.

```
{
    "principals": [
        "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8",
        "arn:aws:iot:us-
east-1:123456789012:cert/
1a234b39b4b68278f2e9d84bf97eac2cbf4a1c28b23ea29a44559b9bcf8d395b"
    ]
}
```

Per ulteriori informazioni, consulta ListThingPrincipalsl'AWS IoT Core API Reference.

Elenca le cose associate a una V2 principale

Per elencare gli elementi associati al certificato specificato, insieme al tipo di allegato, esegui il `list-principal-things-v2`comando. Il tipo di allegato si riferisce al modo in cui il certificato è allegato all'oggetto.

```
$ aws iot list-principal-things-v2 \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

L'output può essere simile al seguente.

```
{
    "PrincipalThingObjects": [
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_1"
        },
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_2"
        }

    ]
}
```

Per ulteriori informazioni, consulta ListPrincipalThingsV2 dall'AWS IoT Core API Reference.

Elenca i principi associati a una cosa V2

Per elencare i certificati associati all'oggetto specificato, insieme al tipo di allegato, esegui il `list-thing-principals-V2`comando. Il tipo di allegato si riferisce al modo in cui il certificato è allegato all'oggetto.

```
$ aws iot list-thing-principals-v2 \
    --thing-name "thing_1"
```

L'output può essere simile al seguente.

```
{
    "ThingPrincipalObjects": [
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
        },
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
1a234b39b4b68278f2e9d84bf97eac2cbf4a1c28b23ea29a44559b9bcf8d395b"
        }
    ]
}
```

Per ulteriori informazioni, consulta ListThingsPrincipalV2 dall'AWS IoT Core API Reference.

Scollegamento di un'entità principale da un oggetto

Usa il comando `DetachThingPrincipal` per scollegare un certificato da un oggetto:

```
$ aws iot detach-thing-principal \
    --thing-name "MyLightBulb" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

Il comando DetachThingPrincipal non produce output.

Per ulteriori informazioni, consulta detach-thing-principall'AWS IoT Core API Reference.

Tipi di oggetti

I tipi di oggetti permettono di archiviare la descrizione e le informazioni di configurazione comuni a tutti gli oggetti associati allo stesso tipo. Ciò semplifica la gestione degli oggetti nel registro. Ad esempio, puoi definire un tipo di LightBulb cosa. Tutti gli elementi associati al tipo di LightBulb oggetto condividono una serie di attributi: numero di serie, produttore e potenza. Quando si crea un oggetto di tipo LightBulb (o si modifica il tipo di un oggetto esistente LightBulb), è possibile specificare i valori per ciascuno degli attributi definiti nel tipo di LightBulb oggetto.

Anche se i tipi di oggetti sono opzionali, il loro utilizzo è un modo semplice per individuare le cose.

- Gli oggetti con un tipo possono avere fino a 50 attributi.
- Gli oggetti senza un tipo possono avere fino a tre attributi.
- Un oggetto può essere associato a un solo tipo.
- Non vi è alcun limite al numero di tipi di oggetti che è possibile creare nell'account.

Non è possibile modificare il nome di un tipo di oggetto dopo averlo creato. È possibile dichiarare obsoleto un tipo di oggetto in qualsiasi momento per impedire che vi vengano associati nuovi oggetti. È inoltre possibile eliminare i tipi di oggetti a cui non sono associati oggetti.

Argomenti:

- Creazione di un tipo di oggetto
- Visualizzazione dell'elenco di tipi di oggetti
- Descrizione di un tipo di oggetto
- Associazione di un tipo di oggetto a un oggetto
- Aggiornare un tipo di oggetto
- Dichiarazione di un tipo di oggetto come obsoleto
- Eliminazione di un tipo di oggetto

Creazione di un tipo di oggetto

Usa il comando CreateThingType per creare un tipo di oggetto:

```
$ aws iot create-thing-type

                --thing-type-name "LightBulb" --thing-type-properties
 "thingTypeDescription=light bulb type, searchableAttributes=wattage,model"
```

Il comando CreateThingType restituisce una risposta che contiene il tipo di oggetto e il relativo ARN:

```
{
    "thingTypeName": "LightBulb",
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb"
}
```

Visualizzazione dell'elenco di tipi di oggetti

Usa il comando ListThingTypes per elencare i tipi di oggetti:

```
$ aws iot list-thing-types
```

Il ListThingTypes comando restituisce un elenco dei tipi di oggetti definiti nel tuo Account AWS:

```
{
    "thingTypes": [
        {
            "thingTypeName": "LightBulb",
            "thingTypeProperties": {
                "searchableAttributes": [
                    "wattage",
                    "model"
                ],
                "thingTypeDescription": "light bulb type"
            },
            "thingTypeMetadata": {
                "deprecated": false,
                "creationDate": 1468423800950
            }
        }
    ]
}
```

Descrizione di un tipo di oggetto

Usa il comando DescribeThingType per ottenere informazioni su un tipo di oggetto:

```
$ aws iot describe-thing-type --thing-type-name "LightBulb"
```

Il comando DescribeThingType restituisce le informazioni relative al tipo specificato:

```
{
    "thingTypeProperties": {
        "searchableAttributes": [
            "model",
            "wattage"
        ],
        "thingTypeDescription": "light bulb type"
    },
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",
    "thingTypeName": "LightBulb",
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1544466338.399
    }
}
```

Associazione di un tipo di oggetto a un oggetto

Usa il comando CreateThing per specificare un tipo di oggetto durate la creazione di un oggetto:

```
$ aws iot create-thing --thing-name "MyLightBulb" --thing-type-name "LightBulb" --
attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Usa il comando UpdateThing in qualsiasi momento per modificare il tipo di oggetto associato a un oggetto:

```
$ aws iot update-thing --thing-name "MyLightBulb"
               --thing-type-name "LightBulb" --attribute-payload  "{\"attributes\":
 {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Puoi anche usare il comando UpdateThing per eliminare l'associazione di un oggetto a un tipo.

Aggiornare un tipo di oggetto

È possibile utilizzare il UpdateThingType comando per aggiornare un tipo di oggetto quando si crea un oggetto:

```
$ aws iot create-thing --thing-name "MyLightBulb" --thing-type-name "LightBulb" --
attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Usa il comando UpdateThing in qualsiasi momento per modificare il tipo di oggetto associato a un oggetto:

```
$ aws iot update-thing --thing-name "MyLightBulb"
                --thing-type-name "LightBulb" --attribute-payload  "{\"attributes\":
 {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Puoi anche usare il comando UpdateThing per eliminare l'associazione di un oggetto a un tipo.

Dichiarazione di un tipo di oggetto come obsoleto

I tipi di oggetti non sono modificabili. Dopo essere stati definiti, non possono essere modificati. È possibile tuttavia dichiarare obsoleto un tipo di oggetto per impedire agli utenti di associarvi nuovi oggetti. Tutti gli oggetti esistenti associati al tipo di oggetto non vengono modificati.

Per dichiarare obsoleto un tipo di oggetto, usa il comando DeprecateThingType:

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType"
```

Usa il comando DescribeThingType per visualizzare il risultato:

```
$ aws iot describe-thing-type --thing-type-name "StopLight":
```

```
{
    "thingTypeName": "StopLight",
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
        ],
```

```
        "thingTypeDescription": "traffic light type",
    },
    "thingTypeMetadata": {
        "deprecated": true,
        "creationDate": 1468425854308,
        "deprecationDate": 1468446026349
    }
}
```

La dichiarazione di un tipo di oggetto come obsoleto è un'operazione reversibile. È possibile annullare la dichiarazione di un tipo come obsoleto usando il flag `--undo-deprecate` con il comando DeprecateThingType dell'interfaccia a riga di comando:

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType" --undo-deprecate
```

Usa il comando DescribeThingType dell'interfaccia a riga di comando per visualizzare il risultato:

```
$ aws iot describe-thing-type --thing-type-name "StopLight":
```

```
{
    "thingTypeName": "StopLight",
    "thingTypeArn": "arn:aws:iot:us-east-1:123456789012:thingtype/StopLight",
    "thingTypeId": "12345678abcdefgh12345678ijklmnop12345678"
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
        ],
        "thingTypeDescription": "traffic light type"
    },
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1468425854308,
    }
}
```

Eliminazione di un tipo di oggetto

È possibile eliminare i tipi di oggetti solo dopo averli dichiarati obsoleti. Per eliminare un tipo di oggetto, usa il comando DeleteThingType:

```
$ aws iot delete-thing-type --thing-type-name "StopLight"
```

> **Note**
>
> Prima di poter eliminare un tipo di oggetto, attendi cinque minuti dopo averlo reso obsoleto.

Gruppi di oggetti statici

La classificazione degli oggetti in gruppi statici consente di gestire diversi oggetti contemporaneamente. I gruppi di oggetti statici contengono un gruppo di oggetti gestiti utilizzando la console, l'interfaccia a riga di comando o l'API. I gruppi di oggetti dinamici, d'altra parte, contengono oggetti che corrispondono a una query specificata. I gruppi di oggetti statici possono anche contenere altri gruppi di oggetti statici. È possibile creare una gerarchia di gruppi. È possibile collegare una policy a un gruppo padre affinché venga ereditata dai gruppi figlio, oltre che da tutti gli oggetti nel gruppo e nei relativi gruppi figlio. In questo modo, si semplifica il controllo delle autorizzazioni per un numero elevato di oggetti.

> **Note**
>
> Le policy dei gruppi di oggetti non consentono l'accesso alle operazioni del piano AWS IoT Greengrass dati. Per consentire a un oggetto l'accesso a un'operazione del piano AWS IoT Greengrass dati, aggiungi l'autorizzazione a una AWS IoT policy che alleghi al certificato dell'oggetto. Per ulteriori informazioni, consultare Autenticazione e autorizzazione del dispositivo nella Guida per gli sviluppatori di AWS IoT Greengrass .

Di seguito sono elencate le operazioni che è possibile eseguire con i gruppi di oggetti statici:

- Creare, descrivere o eliminare un gruppo.
- Aggiungere un oggetto a un gruppo o a più gruppi.
- Rimuovere un oggetto da un gruppo.
- Elencare i gruppi creati.
- Elencare tutti i gruppi figlio di un gruppo (i relativi discendenti diretti e indiretti).
- Elencare gli oggetti in un gruppo, inclusi tutti gli oggetti nei relativi gruppi figlio.

- Elencare tutti i gruppi predecessore di un gruppo (i relativi oggetti padre diretti e indiretti).
- Aggiungere, eliminare o aggiornare gli attributi di un gruppo. Gli attributi sono coppie nome–valore che è possibile usare per archiviare le informazioni su un gruppo.
- Collegare o scollegare una policy a o da un gruppo.
- Elencare le policy collegate a un gruppo.
- Elencare le policy ereditate da un oggetto (in forza delle policy collegate al relativo gruppo o a uno dei gruppi padre).
- Configurare le opzioni di logging per gli oggetti in un gruppo. Per informazioni, consulta Configurare la registrazione AWS IoT.
- Creare processi che vengono inviati ed eseguiti su ogni oggetto in un gruppo e nei relativi gruppi figlio. Per informazioni, consulta AWS IoT Lavori.

> **Note**
>
> Quando un oggetto è collegato a un gruppo di oggetti statico a cui è associata una AWS IoT Core policy, il nome dell'oggetto deve corrispondere all'ID client.

Di seguito sono elencate alcune limitazioni dei gruppi di oggetti statici:

- Un gruppo può avere un solo elemento padre diretto.
- Se un gruppo è figlio di un altro gruppo, specificalo al momento della creazione.
- Non è possibile modificare il padre di un gruppo in un secondo momento, quindi assicurati di pianificare la gerarchia del gruppo e crea un gruppo padre prima di creare i gruppi figlio in esso contenuti.
- Il numero di gruppi a cui un oggetto può appartenere è limitato.
- Non è possibile aggiungere un oggetto a più di un gruppo nella stessa gerarchia. (In altre parole, non è possibile aggiungere un oggetto a due gruppi che condividono uno stesso padre.)
- Non è possibile rinominare un gruppo.
- I nomi dei gruppo di oggetti non possono contenere caratteri internazionali, ad esempio, û, é e ñ.
- Non utilizzare informazioni che consentano l'identificazione personale nel nome del gruppo di oggetti. Il nome del gruppo di oggetti può essere visualizzato nelle comunicazioni e nei report non crittografati.

Le operazioni di collegamento e scollegamento di policy ai e dai gruppi possono migliorare notevolmente la sicurezza delle operazioni di AWS IoT in diversi modi. Il metodo per dispositivo di collegamento di una policy a un certificato, che viene quindi collegato a un oggetto, richiede molto tempo e rende difficile aggiornare rapidamente le policy o modificarle in un parco istanze di dispositivi. Se si collega una policy al gruppo di oggetti, sono necessarie meno operazioni quando è il momento di ruotare i certificati di un oggetto. Le policy vengono inoltre applicate dinamicamente agli oggetti quando cambia l'appartenenza ai gruppi, quindi non è necessario ricreare un set di autorizzazioni complesso ogni volta che per un dispositivo cambia l'appartenenza a un gruppo.

Creazione di un gruppo di oggetti statico

Utilizza il comando CreateThingGroup per creare un gruppo di oggetti statico.

```
$ aws iot create-thing-group --thing-group-name LightBulbs
```

Il comando CreateThingGroup restituisce una risposta che contiene il nome, l'ID e l'ARN del gruppo di oggetti statico:

```
{
    "thingGroupName": "LightBulbs",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
}
```

> **Note**
>
> Non è consigliabile utilizzare informazioni di identificazione personale nei nomi dei gruppi di oggetti.

Di seguito è illustrato un esempio in cui viene specificato un elemento padre del gruppo di oggetti statico al momento della creazione:

```
$ aws iot create-thing-group --thing-group-name RedLights --parent-group-name
 LightBulbs
```

Come in precedenza, il comando CreateThingGroup restituisce una risposta che contiene il nome del gruppo di oggetti statico e i relativi ID e ARN:

```
{
    "thingGroupName": "RedLights",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
}
```

Important

Quando crei gerarchie di gruppi, tieni presenti i limiti seguenti:

- Un gruppo di oggetti può avere un solo elemento padre diretto.
- Il numero di gruppi figlio diretti che un gruppo di oggetti può avere è limitato.
- La profondità massima di una gerarchia di gruppi è limitata.
- Il numero di attributi che un gruppo di oggetti può avere è limitato. Gli attributi sono coppie nome–valore che è possibile usare per archiviare le informazioni su un gruppo. Anche le lunghezze di ciascun nome di attributo e di ogni valore sono limitate.

Descrizione di un gruppo di oggetti

Usa il comando DescribeThingGroup per ottenere informazioni su un gruppo di oggetti:

```
$ aws iot describe-thing-group --thing-group-name RedLights
```

Il comando DescribeThingGroup restituisce le informazioni relative al gruppo specificato:

```
{
    "thingGroupName": "RedLights",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
    "thingGroupId": "12345678abcdefgh12345678ijklmnop12345678",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1478299948.882
        "parentGroupName": "Lights",
        "rootToParentThingGroups": [
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ShinyObjects",
                "groupName": "ShinyObjects"
```

```
            },
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs",
                "groupName": "LightBulbs"
            }
        ]
    },
    "thingGroupProperties": {
        "attributePayload": {
            "attributes": {
                "brightness": "3400_lumens"
            },
        },
        "thingGroupDescription": "string"
    },
}
```

Aggiunta di un oggetto a un gruppo di oggetti statico

È possibile usare il comando AddThingToThingGroup per aggiungere un oggetto a un gruppo di oggetti statico:

```
$ aws iot add-thing-to-thing-group --thing-name MyLightBulb --thing-group-name
 RedLights
```

Il comando AddThingToThingGroup non produce output.

> **Important**
>
> È possibile aggiungere un oggetto a un massimo di 10 gruppi. Non è tuttavia possibile aggiungere un oggetto a più di un gruppo nella stessa gerarchia. (In altre parole, non è possibile aggiungere un oggetto a due gruppi che condividono uno stesso padre.)
> Se un oggetto appartiene al massimo numero possibile di gruppi di oggetti e almeno uno di questi gruppi è un gruppo di oggetti dinamico, è possibile utilizzare il flag overrideDynamicGroups per far sì che i gruppi statici abbiano la priorità sui gruppi dinamici.

Rimozione di un oggetto da un gruppo di oggetti statico

Usa il comando RemoveThingFromThingGroup per rimuovere un oggetto da un gruppo:

```
$ aws iot remove-thing-from-thing-group --thing-name MyLightBulb --thing-group-name
 RedLights
```

Il comando RemoveThingFromThingGroup non produce output.

Elenco di oggetti in un gruppo di oggetti

Utilizza il comando ListThingsInThingGroup per elencare gli oggetti che appartengono a un gruppo:

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs
```

Il comando ListThingsInThingGroup restituisce un elenco degli oggetti nel gruppo specificato:

```
{
    "things":[
        "TestThingA"
    ]
}
```

Il parametro `--recursive` permette di elencare gli oggetti appartenenti a un gruppo e quelli nei relativi gruppi figlio:

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs --recursive
```

```
{
    "things":[
        "TestThingA",
        "MyLightBulb"
    ]
}
```

> **Note**
>
> Questa operazione è consistente finale. In altre parole, le modifiche al gruppo di cose potrebbero non essere riflesse contemporaneamente.

Visualizzazione dell'elenco di gruppi di oggetti

Puoi usare il comando ListThingGroups per elencare i gruppi di oggetti del tuo account:

```
$ aws iot list-thing-groups
```

Il ListThingGroups comando restituisce un elenco dei gruppi di cose presenti in Account AWS:

```
{
    "thingGroups": [
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ReplaceableObjects"
        }
    ]
}
```

Usa i filtri facoltativi per elencare i gruppi con un determinato gruppo padre (`--parent-group`) o i gruppi il cui nome inizia con un determinato prefisso (`--name-prefix-filter`). Il parametro `--recursive` permette di elencare anche tutti i gruppi figlio e non solo i gruppi figlio diretti di un gruppo di oggetti:

```
$ aws iot list-thing-groups --parent-group LightBulbs
```

In questo caso, il ListThingGroups comando restituisce un elenco dei gruppi figli diretti del gruppo di cose definito in Account AWS:

```
{
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        }
    ]
}
```

Utilizza il parametro `--recursive` con il comando ListThingGroups per elencare tutti i gruppi figlio di un gruppo di oggetti, non solo i figli diretti:

```
$ aws iot list-thing-groups --parent-group LightBulbs --recursive
```

Il comando ListThingGroups restituisce un elenco di tutti i gruppi figlio del gruppo di oggetti:

```
{
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
    ]
}
```

> **Note**
>
> Questa operazione è consistente finale. In altre parole, le modifiche al gruppo di cose potrebbero non essere riflesse contemporaneamente.

Elenco dei gruppi per un oggetto

Puoi utilizzare il comando ListThingGroupsForThing per elencare i gruppi diretti a cui appartiene l'oggetto:

```
$ aws iot list-thing-groups-for-thing --thing-name MyLightBulb
```

Il comando ListThingGroupsForThing restituisce un elenco dei gruppi di oggetti diretti a cui l'oggetto appartiene:

```
{
    "thingGroups":[
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ReplaceableObjects"
        }
    ]
}
```

Puoi anche accedere a questa API all'interno del motore delle regole utilizzando la funzione `get_registry_data()` inline. È possibile utilizzare questa funzione per accedere e utilizzare dinamicamente le informazioni del registro degli oggetti (inclusi attributi, tipi di oggetti e appartenenze ai gruppi) chiamando e `ListThingGroupsForThing` APIs direttamente all'interno delle AWS IoT regole, abilitando l'elaborazione `DescribeThing` e il routing dei messaggi in tempo reale in base ai dati del registro del dispositivo. Per ulteriori informazioni, consulta `get_registry_data`.

Aggiornamento di un gruppo di oggetti statico

Usa il comando UpdateThingGroup per aggiornare gli attributi di un gruppo di oggetti statico:

```
$ aws iot update-thing-group --thing-group-name "LightBulbs" --thing-group-properties
 "thingGroupDescription=\"this is a test group\", attributePayload=\"{\"attributes
\"={\"Owner\"=\"150\",\"modelNames\"=\"456\"}}"
```

Il comando UpdateThingGroup restituisce una risposta che contiene il numero di versione del gruppo dopo l'aggiornamento:

```
{
    "version": 4
}
```

> **Note**
>
> Il numero di attributi che un oggetto può avere è limitato.

Eliminazione di un gruppo di oggetti

Per eliminare un gruppo di oggetti, usa il comando DeleteThingGroup:

```
$ aws iot delete-thing-group --thing-group-name "RedLights"
```

Il comando DeleteThingGroup non produce output.

> **Important**
>
> Se tenti di eliminare un gruppo di oggetti con gruppi di oggetti figlio, viene generato un errore:
>
> ```
> A client error (InvalidRequestException) occurred when calling the
>  DeleteThingGroup
> operation: Cannot delete thing group : RedLights when there are still child
>  groups attached to it.
> ```
>
> Prima di eliminare il gruppo, elimina prima tutti i gruppi di bambini.

È possibile eliminare un gruppo con oggetti figlio, ma le autorizzazioni concesse agli oggetti in virtù dell'appartenenza al gruppo non sono più valide. Prima di eliminare un gruppo a cui è collegata una policy, controlla attentamente che la rimozione delle autorizzazioni non comprometta il corretto funzionamento degli oggetti nel gruppo. Inoltre, i comandi che mostrano a quali gruppi appartiene un oggetto (ad esempio,ListGroupsForThing) potrebbero continuare a mostrare il gruppo mentre i record nel cloud vengono aggiornati.

Collegamento di una policy a un gruppo di oggetti statico

Usa il comando AttachPolicy per collegare una policy a un gruppo di oggetti statico e, di conseguenza, a tutti gli oggetti presenti nel gruppo e agli oggetti nei relativi gruppi figlio:

```
$ aws iot attach-policy \
  --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \
  --policy-name "myLightBulbPolicy"
```

Il comando AttachPolicy non produce output

> **Important**
>
> È possibile collegare un numero massimo di due policy per un gruppo.

> **Note**
>
> Non è consigliabile utilizzare informazioni di identificazione personale nei nomi delle policy.

Il parametro `--target` può essere l'ARN di un gruppo di oggetti (come indicato sopra), l'ARN di un certificato o un'identità Amazon Cognito. Per ulteriori informazioni su policy, certificati e autenticazione, consulta Autenticazione.

Per ulteriori informazioni, consulta policy AWS IoT Core.

Scollegamento di una policy da un gruppo di oggetti statico

Usa il comando DetachPolicy per scollegare una policy da un gruppo di oggetti e, di conseguenza, da tutti gli oggetti presenti nel gruppo e gli oggetti nei relativi gruppi figlio:

```
$ aws iot detach-policy --target "arn:aws:iot:us-west-2:123456789012:thinggroup/
LightBulbs" --policy-name "myLightBulbPolicy"
```

Il comando DetachPolicy non produce output.

Elenco di policy collegate a un gruppo di oggetti statico

Usa il comando ListAttachedPolicies per elencare le policy collegate a un gruppo di oggetti statico:

```
$ aws iot list-attached-policies --target "arn:aws:iot:us-
west-2:123456789012:thinggroup/RedLights"
```

Il parametro `--target` può essere l'ARN di un gruppo di oggetti (come indicato sopra), l'ARN di un certificato o un'identità Amazon Cognito.

Aggiungi il parametro opzionale `--recursive` per includere tutte le policy collegate ai gruppi padre del gruppo.

Il comando ListAttachedPolicies restituisce un elenco di policy:

```
{
    "policies": [
        "MyLightBulbPolicy"
        ...
    ]
}
```

Visualizzazione dell'elenco di gruppi per una policy

Usa il comando ListTargetsForPolicy per elencare i target, inclusi i gruppi, a cui una policy è collegata:

```
$ aws iot list-targets-for-policy --policy-name "MyLightBulbPolicy"
```

Aggiungi il parametro opzionale `--page-size` *`number`* per specificare il numero massimo di risultati da restituire per ogni query e il parametro `--marker` *`string`* nelle chiamate successive per recuperare il set di risultati successivo, se presente.

Il comando ListTargetsForPolicy restituisce un elenco di target e il token da usare per recuperare ulteriori risultati:

```
{
    "nextMarker": "string",
    "targets": [ "string" ... ]
}
```

Recupero delle policy valide per un oggetto

Usa il comando GetEffectivePolicies per elencare le policy valide per un oggetto, incluse le policy collegate ai gruppi a cui l'oggetto appartiene, indipendentemente dal fatto che il gruppo sia un padre diretto o un predecessore indiretto:

```
$ aws iot get-effective-policies \
  --thing-name "MyLightBulb" \
  --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

Usa il parametro `--principal` per specificare l'ARN del certificato collegato all'oggetto. Se usi l'autenticazione dell'identità di Amazon Cognito, utilizza il parametro `--cognito-identity-pool-id` e, facoltativamente, aggiungi il parametro `--principal` per specificare un'identità di Amazon Cognito. Se specifichi solo `--cognito-identity-pool-id`, vengono restituite le policy associate al ruolo del pool di identità per gli utenti non autenticati. Se usi entrambi i parametri, vengono restituite le policy associate al ruolo del pool di identità per gli utenti autenticati.

Il parametro `--thing-name` è opzionale e può essere usato al posto del parametro `--principal`. Quando viene usato, vengono restituite le policy collegate a qualsiasi gruppo a cui l'oggetto appartiene e le policy collegate a qualsiasi gruppo padre di questi gruppi (fino al gruppo root della gerarchia).

Il comando GetEffectivePolicies restituisce un elenco di policy:

```
{
    "effectivePolicies": [
        {
            "policyArn": "string",
            "policyDocument": "string",
            "policyName": "string"
        }
        ...
    ]
```

```
}
```

Test dell'autorizzazione per le operazioni MQTT

Usa il comando TestAuthorization per verificare se un'operazione MQTT (`Publish`, `Subscribe`) è permessa per un oggetto:

```
aws iot test-authorization \
    --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847" \
    --auth-infos "{\"actionType\": \"PUBLISH\", \"resources\": [ \"arn:aws:iot:us-
east-1:123456789012:topic/my/topic\"]}"
```

Usa il parametro `--principal` per specificare l'ARN del certificato collegato all'oggetto. Se utilizzi l'autenticazione dell'identità di Amazon Cognito, specifica un'identità di Cognito come `--principal` oppure utilizza il parametro `--cognito-identity-pool-id` o entrambe le cose. Se specifichi solo `--cognito-identity-pool-id`, vengono considerate le policy associate al ruolo del pool di identità per gli utenti non autenticati. Se usi entrambi i parametri, vengono considerate le policy associate al ruolo del pool di identità per gli utenti autenticati.

Specifica una o più operazioni MQTT da testare elencando i set di risorse e i tipi di operazioni dopo il parametro `--auth-infos`. Il campo `actionType` deve contenere "PUBLISH", "SUBSCRIBE", "RECEIVE" o "CONNECT". Il `resources` campo deve contenere un elenco di risorse ARNs. Per ulteriori informazioni, consulta AWS IoT Core politiche.

È possibile testare gli effetti dell'aggiunta di policy specificandole con il parametro `--policy-names-to-add`. In alternativa, è possibile testare gli effetti della rimozione di policy con il parametro `--policy-names-to-skip`.

Puoi usare il parametro opzionale `--client-id` per affinare ulteriormente i risultati.

Il comando TestAuthorization restituisce i dettagli delle operazioni permesse o non permesse per ogni set di query `--auth-infos` specificato:

```
{
    "authResults": [
        {
            "allowed": {
                "policies": [
                    {
```

```
                        "policyArn": "string",
                        "policyName": "string"
                    }
                ]
            },
            "authDecision": "string",
            "authInfo": {
                "actionType": "string",
                "resources": [ "string" ]
            },
            "denied": {
                "explicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                },
                "implicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                }
            },
            "missingContextValues": [ "string" ]
        }
    ]
}
```

Gruppo di oggetti dinamici

I gruppi di oggetti dinamici vengono creati da specifiche query di ricerca nel registro. I parametri delle query di ricerca come la connettività del dispositivo, la creazione di ombre del dispositivo e i dati AWS IoT Device Defender sulle violazioni supportano questa funzionalità. I gruppi di oggetti dinamici richiedono l'indicizzazione del parco veicoli abilitata per indicizzare, cercare e aggregare i dati dei dispositivi. Puoi visualizzare in anteprima gli elementi in un gruppo di oggetti dinamico utilizzando una query di ricerca di indicizzazione della flotta prima di crearlo. Per ulteriori informazioni, consultare Indicizzazione del parco istanze e Sintassi delle query.

> **Note**
>
> Le operazioni dinamiche dei gruppi di oggetti vengono misurate nelle operazioni di registro. Per ulteriori informazioni, vedere AWS IoT Core ulteriori dettagli sulla misurazione.

I gruppi di oggetti dinamici differiscono dai gruppi di oggetti statici per le seguenti caratteristiche:

- L'appartenenza degli oggetti non è esplicitamente definita. Per creare un gruppo di oggetti dinamico, definite una stringa di query di ricerca per determinare l'appartenenza al gruppo.
- I gruppi di oggetti dinamici non possono far parte di una gerarchia.
- I gruppi di oggetti dinamici non possono avere delle policy a loro applicate.
- È possibile utilizzare un altro set di comandi per creare, aggiornare ed eliminare i gruppi di oggetti dinamici. Per tutte le altre operazioni, si utilizzano gli stessi comandi per entrambi i tipi di gruppi di oggetti.
- Il numero di gruppi dinamici per Account AWS è limitato.

- Non utilizzare informazioni che consentano l'identificazione personale nel nome del gruppo di oggetti. Il nome del gruppo di oggetti può essere visualizzato nelle comunicazioni e nei report non crittografati.

Per ulteriori informazioni sui gruppi di oggetti statici, consulta Gruppi di oggetti statici.

Casi d'uso di gruppi di oggetti dinamici

È possibile utilizzare gruppi di oggetti dinamici per i seguenti casi d'uso:

Specificate un gruppo di oggetti dinamico come destinazione per un job

La creazione di un lavoro continuo con un gruppo di oggetti dinamico come destinazione consente di indirizzare automaticamente i dispositivi quando soddisfano i criteri desiderati. I criteri possono essere lo stato della connettività o qualsiasi criterio memorizzato nel registro o nello shadow, ad esempio la versione o il modello del software. Se un oggetto non appare nel gruppo di oggetti dinamico, non riceverà il documento di lavoro dal lavoro.

Ad esempio, se il tuo parco dispositivi richiede un aggiornamento del firmware per ridurre al minimo il rischio di interruzioni durante il processo di aggiornamento e desideri aggiornare il firmware

solo su dispositivi con una durata della batteria superiore all'80%. È possibile creare un gruppo di oggetti dinamico denominato 80 PercentBatteryLife che include solo dispositivi con una durata della batteria superiore all'80% e utilizzarlo come destinazione per il proprio lavoro. Solo i dispositivi che soddisfano i criteri di durata della batteria riceveranno l'aggiornamento del firmware. Quando i dispositivi raggiungono il criterio di durata della batteria dell'80%, vengono automaticamente aggiunti al gruppo di oggetti dinamico e riceveranno l'aggiornamento del firmware.

È inoltre possibile disporre di più modelli di dispositivi con firmware o sistema operativo diversi, il che richiede versioni diverse dei nuovi aggiornamenti software. Questo è il caso d'uso più comune per i gruppi dinamici con processi continui, in cui è possibile creare un gruppo dinamico per ogni modello di dispositivo, firmware e combinazione di sistema operativo. È quindi possibile impostare lavori continui per ciascuno di questi gruppi dinamici per inviare aggiornamenti software man mano che i dispositivi diventano automaticamente membri di questi gruppi in base ai criteri definiti.

Per ulteriori informazioni sulla specifica dei gruppi di elementi come destinazioni di processo, consulta CreateJob.

Utilizza le modifiche dinamiche all'appartenenza ai gruppi per eseguire le azioni desiderate

Ogni volta che un dispositivo viene aggiunto o rimosso da un gruppo di oggetti dinamico, viene inviata una notifica a un argomento MQTT come parte degli aggiornamenti degli eventi del registro. È possibile configurare AWS IoT Core le regole per interagire con AWS i servizi in base agli aggiornamenti dinamici delle appartenenze ai gruppi e intraprendere le azioni desiderate. Le azioni di esempio includono scrivere su Amazon DynamoDB, richiamare una funzione Lambda o inviare una notifica ad Amazon SNS.

Aggiungi dispositivi a un gruppo di oggetti dinamico per il rilevamento automatico delle violazioni

AWS IoT Device Defender I clienti di Detect possono definire un profilo di sicurezza su un gruppo di oggetti dinamico. I dispositivi del gruppo di oggetti dinamico vengono rilevati automaticamente per rilevare eventuali violazioni dal profilo di sicurezza definito nel gruppo.

Imposta i livelli di log sui gruppi di oggetti dinamici per osservare i dispositivi con una registrazione dettagliata

È possibile specificare un livello di log su un gruppo di oggetti dinamico. Ciò è utile se si desidera personalizzare solo il livello e i dettagli di registrazione per i dispositivi che soddisfano determinati

criteri. Ad esempio, se sospetti che dispositivi con una determinata versione del firmware stiano causando errori nell'argomento pubblicato di una regola specifica, potresti voler impostare una registrazione dettagliata per eseguire il debug di questi problemi. In questo caso, puoi creare un gruppo dinamico per tutti i dispositivi che dispongono di questa versione del firmware, che supponiamo sia memorizzata come attributo di registro o nell'ombra di un dispositivo. È quindi possibile impostare un livello di debug, con l'obiettivo di registrazione definito come questo gruppo di oggetti dinamico. Per ulteriori informazioni sulla registrazione dettagliata, consultate Monitoraggio tramite log. AWS IoT CloudWatch Per ulteriori informazioni su come specificare un livello di registrazione per un gruppo di oggetti specifico, consulta Configurare la registrazione specifica per una risorsa. AWS IoT

Creazione di un gruppo di oggetti dinamico

Utilizza il comando CreateDynamicThingGroup per creare un gruppo di oggetti dinamico. Per creare un gruppo di oggetti dinamico per lo PercentBatteryLife scenario 80, utilizzate il comando create-dynamic-thing-group CLI:

```
$ aws iot create-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --query-string "attributes.batterylife80"
```

> **Note**
>
> Non utilizzate informazioni di identificazione personale nei nomi dei gruppi di cose dinamici.

Il CreateDynamicThingGroup comando restituisce una risposta. La risposta contiene il nome dell'indice, la stringa di query, la versione della query, il nome del gruppo di oggetti, l'ID del gruppo di oggetti e l'Amazon Resource Name (ARN) del gruppo di cose:

```
{
    "indexName": "AWS_Things",
    "queryVersion": "2017-09-30",
    "thingGroupName": "80PercentBatteryLife",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/80PercentBatteryLife",
    "queryString": "attributes.batterylife80\n",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx"
}
```

La creazione di gruppi di oggetti dinamici non avviene contemporaneamente. Il backfill del gruppo di oggetti dinamici richiede tempo per il completamento. Quando si crea un gruppo di oggetti dinamico, lo stato del gruppo è impostato su`BUILDING`. Quando il backfill è completo, lo stato diventa `ACTIVE`. Per verificare lo stato del gruppo di oggetti dinamico, utilizzate il DescribeThingGroupcomando.

Descrizione di un gruppo di oggetti dinamico

Utilizza il comando DescribeThingGroup per ottenere informazioni su un gruppo di oggetti dinamico:

```
$ aws iot describe-thing-group --thing-group-name "80PercentBatteryLife"
```

Il comando DescribeThingGroup restituisce le informazioni relative al gruppo specificato:

```
{
    "status": "ACTIVE",
    "indexName": "AWS_Things",
    "thingGroupName": "80PercentBatteryLife",
    "thingGroupArn": "arn:aws:iot:us-
west-2:123456789012:thinggroup/80PercentBatteryLife",
    "queryString": "attributes.batterylife80\n",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1548716921.289
    },
    "thingGroupProperties": {},
    "queryVersion": "2017-09-30",
    "thingGroupId": "84dd9b5b-2b98-4c65-84e4-be0e1ecf4fd8"
}
```

L'esecuzione DescribeThingGroup su un gruppo di cose dinamico restituisce attributi specifici dei gruppi di cose dinamici. Gli attributi restituiti di esempio sono QueryString e lo status.

Lo stato di un gruppo di oggetti dinamici può utilizzare i seguenti valori:

`ACTIVE`

Il gruppo di oggetti dinamico è pronto per l'uso.

`BUILDING`

Il gruppo di oggetti dinamico viene creato e l'appartenenza agli oggetti è in fase di elaborazione.

`REBUILDING`

L'appartenenza del gruppo di oggetti dinamico viene aggiornata in seguito all'adeguamento della query di ricerca del gruppo.

> **Note**
>
> Dopo aver creato un gruppo di oggetti dinamico, utilizzatelo indipendentemente dal suo stato. Solo i gruppi di oggetti dinamici con uno stato `ACTIVE` includono tutti gli oggetti che corrispondono alle query di ricerca per quel gruppo di oggetti dinamico. I gruppi di oggetti dinamici con gli stati `REBUILDING` e `BUILDING` potrebbero non includere tutti gli oggetti che corrispondono alla query di ricerca.

Aggiornamento di un gruppo di oggetti dinamico

Utilizzare il comando UpdateDynamicThingGroup per aggiornare gli attributi di un gruppo di oggetti dinamico, tra cui la query di ricerca del gruppo. Il comando seguente aggiorna due attributi. Uno è la descrizione del gruppo di oggetti e l'altro è la stringa di query che modifica i criteri di appartenenza impostandoli sulla durata della batteria > 85:

```
$ aws iot update-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --thing-
group-properties "thingGroupDescription=\"This thing group contains devices with a
 battery life greater than 85 percent.\"" --query-string "attributes.batterylife85"
```

Il comando UpdateDynamicThingGroup restituisce una risposta che contiene il numero di versione del gruppo dopo l'aggiornamento:

```
{
    "version": 2
}
```

L'aggiornamento di un gruppo di oggetti dinamico non avviene contemporaneamente. Il backfill del gruppo di oggetti dinamici richiede tempo per il completamento. Quando si aggiorna un gruppo di oggetti dinamico, lo stato del gruppo cambia `REBUILDING` mentre il gruppo aggiorna la propria appartenenza. Quando il backfill è completo, lo stato diventa `ACTIVE`. Per verificare lo stato del gruppo di oggetti dinamico, utilizzate il DescribeThingGroupcomando.

Eliminazione di un gruppo di oggetti dinamico

Utilizzare il comando DeleteDynamicThingGroup per eliminare un gruppo di oggetti dinamico:

```
$ aws iot delete-dynamic-thing-group --thing-group-name "80PercentBatteryLife"
```

Il comando DeleteDynamicThingGroup non produce output.

I comandi che mostrano i gruppi a cui un oggetto appartiene (ad esempio, ListGroupsForThing) potrebbero continuare a segnalare il gruppo mentre i record nel cloud vengono aggiornati.

Limitazioni dei gruppi di oggetti dinamici e statici

I thing group dinamici e i thing group statici condividono le seguenti limitazioni:

- Il numero di attributi che un gruppo di oggetti può avere è limitato.
- Il numero di gruppi a cui un oggetto può appartenere è limitato.
- Non è possibile rinominare i gruppi di oggetti.
- I nomi dei gruppi di oggetti non possono contenere caratteri internazionali, ad esempio, û, é e ñ.

Limitazioni di Dynamic Thing Group

I gruppi di oggetti dinamici presentano le seguenti limitazioni:

Indicizzazione della flotta

Con il servizio di indicizzazione della flotta abilitato, puoi eseguire query di ricerca sulla tua flotta di dispositivi. È possibile creare e gestire gruppi di oggetti dinamici dopo il completamento del riempimento dell'indicizzazione della flotta. Il tempo di completamento del processo di riempimento è direttamente influenzato dalle dimensioni del parco dispositivi registrato presso il. Cloud AWS Una volta attivato il servizio di indicizzazione del parco istanze per i gruppi di oggetti dinamici, non è possibile eliminare tutti i gruppi di oggetti dinamici.

> **Note**
>
> Se si dispone di autorizzazioni per eseguire query all'indice del parco istanze, è possibile accedere ai dati degli oggetti sull'intero parco istanze.

Il numero di gruppi di oggetti dinamici è limitato

Il numero di gruppi di oggetti dinamici è limitato.

I comandi eseguiti in modo corretto possono registrare errori

Quando si crea o si aggiorna un gruppo di cose dinamico, è possibile che alcuni elementi siano idonei per l'inclusione in un gruppo di cose dinamico, ma non vi vengano aggiunti. Questo scenario causerà l'esecuzione corretta del comando di creazione o aggiornamento durante la registrazione di un errore e la generazione di una metrica. `AddThingToDynamicThingGroupsFailed` Una singola metrica può rappresentare più voci di registro.

Una voce del CloudWatch registro degli errori viene creata quando si verifica quanto segue:

- Un oggetto idoneo non può essere aggiunto a un gruppo di oggetti dinamico.
- Un oggetto viene rimosso da un gruppo di oggetti dinamico per aggiungerlo a un altro gruppo.

Quando un oggetto diventa idoneo per essere aggiunto a un gruppo di oggetti dinamico, considerate quanto segue:

- L'oggetto è già nel numero massimo di gruppi in cui può essere? (Consulta limiti)
  - NO: l'oggetto viene aggiunto al gruppo di oggetti dinamico.
  - SÌ: l'oggetto è già membro di altri gruppi di oggetti dinamici?
    - NO: l'oggetto non può essere aggiunto al gruppo di oggetti dinamico, viene registrato un errore e viene generato un parametro `AddThingToDynamicThingGroupsFailed`.
    - SÌ: il gruppo di oggetti dinamico al quale deve unirsi è più vecchio di qualsiasi gruppo di oggetti dinamico al quale l'oggetto già apparteneva?
      - NO: l'oggetto non può essere aggiunto al gruppo di oggetti dinamico, viene registrato un errore e viene generato un parametro `AddThingToDynamicThingGroupsFailed`.
      - SÌ: rimuove l'oggetto dal gruppo di oggetti dinamico più recente, registra un errore e aggiunge l'oggetto al gruppo di oggetti dinamico. Questo genera un errore e un parametro `AddThingToDynamicThingGroupsFailed` per il gruppo di oggetti dinamico dal quale l'oggetto è stato rimosso.

Quando un oggetto in un gruppo di oggetti dinamico non soddisfa più la query di ricerca, l'oggetto viene rimosso dal gruppo di oggetti dinamico. Allo stesso modo, quando un oggetto viene aggiornato

per soddisfare la query di ricerca di un gruppo di oggetti dinamico, l'oggetto viene quindi aggiunto al gruppo come descritto in precedenza. Queste aggiunte e rimozioni sono normali e non producono voci di registro di errore.

Con **`overrideDynamicGroups`** abilitato, i gruppi statici hanno la priorità sui gruppi dinamici

Il numero di gruppi a cui un oggetto può appartenere è limitato. Quando si utilizzano i UpdateThingGroupsForThingcomandi AddThingToThingGroupo per aggiornare l'appartenenza ai thing, l'aggiunta del `--overrideDynamicGroups` parametro dà la priorità ai thing group statici rispetto ai thing group dinamici.

Quando aggiungete un oggetto a un gruppo di oggetti statico, considerate quanto segue:

- L'oggetto appartiene già al numero massimo di gruppi?
  - NO: l'oggetto viene aggiunto al gruppo di oggetti statico.
  - SÌ: l'oggetto è in qualche gruppo dinamico?
    - NO: l'oggetto non può essere aggiunto al gruppo di oggetti. Il comando solleva un'eccezione.
    - SÌ: è stato abilitato --overrideDynamicGroups?
      - NO: l'oggetto non può essere aggiunto al gruppo di oggetti. Il comando solleva un'eccezione.
      - SÌ: l'oggetto viene rimosso dal gruppo di oggetti dinamico creato più di recente, viene registrato un errore e viene generato un parametro `AddThingToDynamicThingGroupsFailed` per il gruppo di oggetti dinamico da cui è stato rimosso l'oggetto. Quindi, l'oggetto viene aggiunto al gruppo di oggetti statico.

I gruppi di oggetti dinamici precedenti hanno la priorità sui gruppi più recenti

Il numero di gruppi a cui un oggetto può appartenere è limitato. Quando un'operazione di creazione o aggiornamento crea un'ulteriore idoneità di gruppo per un oggetto e l'oggetto ha raggiunto il limite di gruppo, può verificarsi la rimozione da un altro gruppo di oggetti dinamico per abilitare questa aggiunta. Per ulteriori informazioni su come ciò si verifica, vedere I comandi eseguiti in modo corretto possono registrare errori e Con `overrideDynamicGroups` abilitato, i gruppi statici hanno la priorità sui gruppi dinamici per esempi.

Quando un oggetto viene rimosso da un gruppo di oggetti dinamico, viene registrato un errore e viene generato un evento.

Non è possibile applicare policy ai gruppi di oggetti dinamici

Il tentativo di applicazione di una policy a un gruppo di oggetti dinamico genera un'eccezione.

L'appartenenza a un gruppo di oggetti dinamico è consistente finale

Solo lo stato finale di un oggetto è valutato per il registro. Gli stati intermedi possono essere ignorati se vengono aggiornati rapidamente. Evita di associare una regola o un lavoro a un gruppo di oggetti dinamico la cui appartenenza dipende da uno stato intermedio.

Associazione di qualsiasi AWS IoT cosa a una connessione client MQTT

Un'associazione di oggetti esclusiva si verifica quando si allega un certificato X.509 a una singola AWS IoT cosa. In questo caso, il certificato non può essere utilizzato con altre cose. Garantendo che un certificato venga utilizzato solo da un singolo dispositivo IoT, aiuta a prevenire le vulnerabilità di sicurezza.

In AWS IoT, l'ID client è un identificatore univoco per un oggetto o un dispositivo quando si connette al broker AWS IoT Core MQTT. Se si utilizza un'associazione non esclusiva, è possibile allegare più elementi allo stesso certificato. Quando è attiva un'associazione di oggetti non esclusiva, per mantenere un'associazione chiara ed evitare potenziali conflitti, è necessario abbinare l'ID client al nome dell'oggetto.

In questo argomento

- Casi d’uso
- Come associare un oggetto a una connessione

Casi d’uso

L'associazione di un oggetto a una connessione offre le seguenti funzionalità.

> **Note**
>
> Tieni presente che se il tuo oggetto IoT e la connessione client hanno un'associazione non esclusiva, puoi utilizzare tutte le seguenti funzionalità tranne la funzionalità degli eventi del

ciclo di vita. Per includere il nome dell'oggetto nei messaggi relativi agli eventi del ciclo di vita, l'oggetto IoT e la connessione client devono avere un'associazione esclusiva.

Variabili Thing Policy: è possibile utilizzare le variabili Thing Policy per autorizzare l'accesso del dispositivo alle AWS IoT operazioni API. Queste variabili consentono di scrivere AWS IoT Core politiche che concedono o negano le autorizzazioni in base a proprietà degli oggetti come nomi, tipi e valori degli attributi. Utilizzando le variabili Thing Policy, è possibile applicare la stessa policy per controllare più AWS IoT Core dispositivi. Ciò consente di semplificare la gestione delle policy e ridurre la duplicazione delle risorse. Per ulteriori informazioni, vedere Variabili della politica di Thing.

Eventi del ciclo di vita: è possibile ricevere il nome dell'oggetto negli eventi del ciclo di vita (ad esempio, connessione, disconnessione, sottoscrizione e annullamento dell'iscrizione). Ciò consente l'elaborazione del nome dell'oggetto incluso nei messaggi, ad esempio nelle regole. Per ulteriori informazioni, consulta Eventi del ciclo di vita.

Registrazione specifica della risorsa: è possibile configurare la registrazione specifica della risorsa per i gruppi di oggetti e applicare facilmente la configurazione di registrazione desiderata per tutti gli elementi all'interno del gruppo di oggetti definito. Per ulteriori informazioni, consulta ???.

Allocazione dei costi: è possibile creare gruppi di fatturazione con tag personalizzati per l'allocazione dei costi e aggiungere gli elementi a questi gruppi. Per ulteriori informazioni, consulta Gruppi di fatturazione.

Come associare un oggetto a una connessione

Se il tuo ID client corrisponde al nome dell'oggetto nel registro, dopo aver allegato un certificato X.509 a quell'oggetto IoT, AWS IoT Core assocerà la connessione client all'oggetto. Se il tuo ID client non corrisponde al nome dell'oggetto nel registro, puoi allegare esclusivamente un certificato X.509 all'oggetto per stabilire questa associazione. La cosa che ha questo allegato esclusivo è chiamata cosa esclusiva. Altrimenti, si chiama cosa non esclusiva. Quando un certificato è associato a un oggetto esclusivo, questo certificato può essere associato ad altri elementi solo se lo si scollega dall'oggetto esclusivo. In questa sezione, scegliete una delle due opzioni Console di gestione AWS o AWS CLI se associare un oggetto a una connessione.

Console di gestione AWS

Per allegare un certificato a un oggetto utilizzando esclusivamente il Console di gestione AWS.

1. Aprire la AWS IoT home page nella AWS IoT console. Nella barra di navigazione a sinistra, da Sicurezza, scegli Certificati.
2. Nella pagina Certificati, scegli un certificato a cui desideri allegare un elemento. Quindi scegli Allega a elementi da Azioni nell'angolo in alto a destra della pagina.

   In alternativa, scegli un certificato e vai alla pagina dei dettagli del certificato. Scegli la scheda Oggetti, quindi scegli Allega agli oggetti.
3. Nella pagina Allega certificato a una o più cose, seleziona la casella di controllo Associa l'oggetto alla connessione. Quindi scegli un elemento a cui allegare questo certificato dall'elenco a discesa Oggetti.
4. Scegli Allega oggetti. Se l'azione ha esito positivo, vedrai un banner con la scritta «Allegato correttamente un elemento al certificato» e l'elemento verrà aggiunto alla scheda Oggetti.

Per scollegare un certificato da un oggetto esclusivo utilizzando il Console di gestione AWS

1. Aprire la AWS IoT home page nella AWS IoT console. Nella barra di navigazione a sinistra, da Sicurezza, scegli Certificati.
2. Nella pagina Certificati, scegli un certificato e vai alla pagina dei dettagli del certificato.
3. Nella pagina dei dettagli del certificato, scegli la scheda Cose. Quindi scegli un elemento a cui vuoi scollegare il certificato. Scegli Scollega cose.
4. Nella finestra Scollega le cose, conferma l'azione. Seleziona Scollega. Se l'azione ha esito positivo, vedrai un banner con la scritta «Hai rimosso con successo un elemento dal tuo certificato» e l'elemento non verrà più visualizzato nella scheda Oggetti.

AWS CLI

1. Per allegare un certificato a un oggetto utilizzando AWS CLI, esegui il attach-thing-principalcomando. Per specificare l' certificate-to-thingallegato esclusivo, è necessario specificare `EXCLUSIVE_THING` nel `--thing-principal-type` campo. Un comando di esempio può essere il seguente.

```
aws iot attach-thing-principal \
    --thing-name "thing_1" \
```

```
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8" \
    --thing-principal-type "EXCLUSIVE_THING"
```

Il comando non produce output. Per ulteriori informazioni, consulta ???.

2. Per elencare gli elementi associati al certificato specificato insieme al tipo di allegato, esegui il `list-principal-things-v2` comando. Il tipo di allegato si riferisce al modo in cui il certificato è allegato all'oggetto. Un comando di esempio può essere il seguente.

```
$ aws iot list-principal-things-v2 \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

L'output può essere simile al seguente.

```
{
    "PrincipalThingObjects": [
        {
            "thingPrincipalType": "EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_1"
        }
    ]
}
```

Per ulteriori informazioni, consulta ???.

3. Per elencare i principi associati all'oggetto specificato insieme al tipo di allegato, esegui il `list-thing-principals-v2` comando. Il tipo di allegato si riferisce al modo in cui il certificato è allegato all'oggetto. Un comando di esempio può essere il seguente.

```
$ aws iot list-thing-principals-v2 \
    --thing-name "thing_1"
```

L'output può essere simile al seguente.

```
{
    "ThingPrincipalObjects": [
        {
```

```
            "thingPrincipalType": "EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
        }
    ]
}
```

Per ulteriori informazioni, consulta ???.

4. Per scollegare un certificato da un oggetto, esegui il detach-thing-principalcomando.

```
aws iot detach-thing-principal \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8" \
    --thing-name "thing_1"
```

Il comando non produce output. Per ulteriori informazioni, consulta ???.

Aggiungere attributi di propagazione per l'arricchimento dei messaggi

In AWS IoT Core, è possibile arricchire i messaggi MQTT dai dispositivi aggiungendo attributi di propagazione, che sono metadati contestuali provenienti da attributi degli oggetti o dettagli di connessione. Questo processo, noto come arricchimento dei messaggi, può essere utile in vari scenari. Ad esempio, puoi arricchire i messaggi per ogni operazione di pubblicazione in entrata senza apportare modifiche sul dispositivo o dover utilizzare regole. Sfruttando gli attributi di propagazione, puoi trarre vantaggio da un modo più efficiente ed economico per arricchire i tuoi dati IoT senza la complessità della configurazione delle regole o della gestione delle configurazioni di ripubblicazione.

La funzionalità di arricchimento dei messaggi è disponibile per i clienti che utilizzano Basic Ingest e Message Broker. AWS IoT Core È importante notare che, sebbene i dispositivi di pubblicazione possano utilizzare qualsiasi versione MQTT, gli abbonati (applicazioni o servizi che utilizzano messaggi) devono supportare MQTT 5 per ricevere i messaggi arricchiti con attributi di propagazione. I messaggi arricchiti verranno aggiunti come proprietà utente MQTT 5 a ogni messaggio pubblicato dai dispositivi. Se si utilizzano le regole, è possibile sfruttare la funzione get_user_properties per recuperare i dati arricchiti per il routing o l'elaborazione dei messaggi in base ai dati.

In AWS IoT Core, puoi aggiungere attributi di propagazione quando crei o aggiorni un tipo di oggetto, utilizzando o il. Console di gestione AWS AWS CLI

> **Important**
>
> Quando si aggiungono attributi di propagazione, è necessario assicurarsi che il client che pubblica il messaggio sia stato autenticato con un certificato. Per ulteriori informazioni, consulta Autenticazione client.

> **Note**
>
> Se si tenta di testare questa funzionalità utilizzando il client di test MQTT all'interno della console, potrebbe non funzionare poiché richiede l'autenticazione dei client MQTT con un certificato associato.

Console di gestione AWS

Per aggiungere attributi di propagazione per l'arricchimento dei messaggi utilizzando il Console di gestione AWS

1. Aprire la AWS IoT home page nella AWS IoT console. Nella barra di navigazione a sinistra, da Gestisci, scegli Tutti i dispositivi. Quindi scegli Tipi di oggetti.
2. Nella pagina Tipi di oggetto, scegli Crea tipo di oggetto.

   Per configurare l'arricchimento dei messaggi aggiornando un tipo di oggetto, scegli un tipo di oggetto. Quindi, nella pagina dei dettagli del tipo di oggetto, scegli Aggiorna.
3. Nella pagina Crea tipo di oggetto, scegli o inserisci le informazioni sul tipo di oggetto nelle proprietà del tipo di oggetto.

   Se si sceglie di aggiornare un tipo di oggetto, verranno visualizzate le proprietà del tipo di oggetto dopo aver scelto Aggiorna nel passaggio precedente.
4. In Configurazione aggiuntiva, espandi Attributi di propagazione. Quindi scegliete l'attributo dell'oggetto e immettete l'attributo dell'oggetto che desiderate inserire nei messaggi pubblicati MQTT5 . Utilizzando la console, è possibile aggiungere fino a tre attributi dell'oggetto.

Nella sezione Attributi di propagazione, scegliete Attributo di connessione e immettete il tipo di attributo e, facoltativamente, il nome dell'attributo.

5. Facoltativamente, aggiungi tag. Quindi scegli Crea tipo di oggetto.

   Se scegli di aggiornare un tipo di oggetto, scegli Aggiorna tipo di oggetto.

AWS CLI

1. Per aggiungere attributi di propagazione per l'arricchimento dei messaggi creando un nuovo tipo di oggetto utilizzando il AWS CLI, esegui il comando. create-thing-type Un comando di esempio può essere il seguente.

```
aws iot create-thing-type \
    --thing-type-name "LightBulb" \
    --thing-type-properties "{\"mqtt5Configuration\":{\"propagatingAttributes\":
[{\"userPropertyKey\":\"iot:ClientId\", \"connectionAttribute\":\"iot:ClientId\"},
 {\"userPropertyKey\":\"test\", \"thingAttribute\":\"A\"}]}}" \
```

   L'output del comando può essere simile al seguente.

```
{
 "thingTypeName": "LightBulb",
 "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",
 "thingTypeId": "ce3573b0-0a3c-45a7-ac93-4e0ce14cd190"
}
```

2. Per configurare l'arricchimento dei messaggi aggiornando un oggetto type using AWS CLI, esegui il update-thing-typecomando. Tieni presente che puoi eseguire l'aggiornamento solo `mqtt5Configuration` quando esegui questo comando. Un comando di esempio può essere il seguente.

```
aws iot update-thing-type \
    --thing-type-name "MyThingType" \
    --thing-type-properties "{\"mqtt5Configuration\":{\"propagatingAttributes\":
[{\"userPropertyKey\":\"iot:ClientId\", \"connectionAttribute\":\"iot:ClientId\"},
 {\"userPropertyKey\":\"test\", \"thingAttribute\":\"A\"}]}}" \
```

   Il comando non produce output.

3. Per descrivere un tipo di oggetto, esegui il `describe-thing-type` comando. Questo comando produrrà un output con le informazioni di configurazione per l'arricchimento dei messaggi sul `thing-type-properties` campo. Un comando di esempio può essere il seguente.

```
aws iot describe-thing-type \
    --thing-type-name "LightBulb"
```

L'output può essere simile al seguente.

```
{
 "thingTypeName": "LightBulb",
 "thingTypeId": "bdf72512-0116-4392-8d79-bf39b17ef73d",
 "thingTypeArn": "arn:aws:iot:us-east-1:123456789012:thingtype/LightBulb",
 "thingTypeProperties": {
  "mqtt5Configuration": {
   "propagatingAttributes": [
    {
     "userPropertyKey": "iot:ClientId",
     "connectionAttribute": "iot:ClientId"
    },
    {
     "userPropertyKey": "test",
     "thingAttribute": "attribute"
    }
   ]
  }
 },
 "thingTypeMetadata": {
  "deprecated": false,
  "creationDate": "2024-10-18T17:37:46.656000+00:00"
 }
}
```

Per ulteriori informazioni, consulta ???.

Taggare le tue risorse AWS IoT

Per facilitare la gestione e l'organizzazione di gruppi di oggetti, tipi di oggetti, regole di argomenti, processi, audit pianificati e profili di sicurezza, puoi scegliere di assegnare i metadati a ciascuna di queste risorse sotto forma di tag. Questa sezione descrive i tag e mostra come crearli.

Per aiutarti a gestire i costi correlati agli oggetti, è possibile creare gruppi di fatturazione che contengono gli oggetti. Puoi quindi assegnare i tag contenenti i metadati a ognuno di questi gruppi di fatturazione. Questa sezione illustra inoltre i gruppi di fatturazione e i comandi disponibili per crearli e gestirli.

Nozioni di base sui tag

È possibile utilizzare i tag per classificare le AWS IoT risorse in diversi modi (ad esempio, per scopo, proprietario o ambiente). Questa funzionalità è molto utile quando hai tante risorse dello stesso tipo: puoi rapidamente individuare una risorsa in base ai tag assegnati. Ogni tag è formato da una chiave e da un valore opzionale, entrambi personalizzabili. Ad esempio, puoi definire un set di tag per i tuoi tipi di oggetti che consente di monitorare i dispositivi per tipo. Ti consigliamo di creare un set di chiavi di tag in grado di soddisfare i requisiti di ciascun tipo di risorsa. Con un set di chiavi di tag coerente, la gestione delle risorse risulta semplificata.

Puoi cercare e filtrare le risorse in base ai tag che aggiungi o applichi. Puoi anche utilizzare i tag dei gruppi di fatturazione per classificare e monitorare i costi. È possibile anche utilizzare i tag per controllare l'accesso alle risorse, come descritto in Utilizzo dei tag con policy IAM.

Per facilitare l'uso, il Tag Editor nella console di AWS gestione offre un modo centralizzato e unificato per creare e gestire i tag. Per ulteriori informazioni, consulta Lavorare con l'editor di tag in Lavorare con la console di AWS gestione.

Puoi anche lavorare con i tag utilizzando l' AWS IoT API AWS CLI and. Puoi associare i tag a gruppi di oggetti, tipi di oggetti, regole di argomenti, processi, profili di sicurezza, policy, gruppi di fatturazione, nonché i pacchetti e le versioni associati agli oggetti quando li crei tramite il campo `Tags` nei seguenti comandi:

- CreateBillingGroup
- CreateDestination
- CreateDeviceProfile

- CreateDynamicThingGroup
- CreateJob
- CreaOTAUpdate
- CreatePolicy
- CreateScheduledAudit
- CreateSecurityProfile
- CreateServiceProfile
- CreateStream
- CreateThingGroup
- CreateThingType
- CreateTopicRule
- CreateWirelessGateway
- CreateWirelessDevice

Puoi aggiungere, modificare o eliminare i tag per le risorse esistenti che supportano il tagging utilizzando i comandi seguenti:

- TagResource
- ListTagsForResource
- UntagResource

Puoi modificare chiavi e valori di tag e rimuovere tag da una risorsa in qualsiasi momento. Puoi impostare il valore di un tag su una stringa vuota, ma non su null. Se aggiungi un tag con la stessa chiave di un tag esistente su quella risorsa, il nuovo valore sovrascrive il vecchio valore. Se elimini una risorsa, verranno eliminati anche tutti i tag associati alla risorsa.

Restrizioni e limitazioni di tag

Ai tag si applicano le seguenti limitazioni di base:

- Numero massimo di tag per risorsa: 50
- Lunghezza massima della chiave: 127 caratteri Unicode in formato UTF-8
- Lunghezza massima del valore: 255 caratteri Unicode in formato UTF-8

- I valori e le chiavi dei tag rispettano la distinzione tra maiuscole e minuscole.
- Non utilizzare il prefisso `aws:` nei nomi dei tag o nei valori. È riservato all' AWS uso. Non è possibile modificare né eliminare i nomi o i valori di tag con tale prefisso. I tag con questo prefisso non vengono conteggiati per il limite del numero di tag per risorsa.
- Se lo schema di tagging viene utilizzato in più servizi e risorse, è necessario tenere presente che in altri servizi possono essere presenti limiti sui caratteri consentiti. I caratteri consentiti sono lettere, spazi e numeri rappresentabili in formato UTF-8, più i caratteri speciali + - = . _ : / @.

Utilizzo dei tag con policy IAM

È possibile applicare autorizzazioni a livello di risorsa basate su tag nelle policy IAM utilizzate per le operazioni API AWS IoT . In questo modo è possibile controllare meglio le risorse che un utente può creare, modificare o utilizzare. Puoi utilizzare l'elemento `Condition` (denominato anche blocco `Condition`) con i seguenti valori e chiavi di contesto di condizione in una policy IAM per controllare l'accesso dell'utente (autorizzazione) in base ai tag della risorsa:

- Utilizza `aws:ResourceTag/`*`tag-key`*`: `*`tag-value`* per concedere o negare agli utenti operazioni su risorse con specifici tag.
- Utilizza `aws:RequestTag/`*`tag-key`*`: `*`tag-value`* per richiedere che un tag specifico venga utilizzato (o non utilizzato) durante la creazione di una richiesta API per creare o modificare una risorsa che abilita i tag.
- Utilizza `aws:TagKeys: [`*`tag-key`*`, ...]` per richiedere che un set di tag specifico venga utilizzato (o non utilizzato) durante la creazione di una richiesta API per creare o modificare una risorsa che abilita i tag.

> **Note**
>
> Le chiavi e i valori del contesto delle condizioni in una policy IAM si applicano solo a quelle AWS IoT azioni in cui un identificatore di una risorsa che può essere taggata è un parametro obbligatorio. Ad esempio, l'uso di non DescribeEndpointè consentito o negato sulla base delle chiavi e dei valori del contesto della condizione perché in questa richiesta non viene fatto riferimento a nessuna risorsa etichettabile (gruppi di oggetti, tipi di oggetti, regole degli argomenti, job o profilo di sicurezza). Per ulteriori informazioni sulle AWS IoT risorse etichettabili e sulle chiavi di condizione che supportano, leggi Azioni, risorse e chiavi di condizione per. AWS IoT

Per ulteriori informazioni sull'utilizzo dei tag, consulta Controllo degli accessi tramite tag nella Guida per l'utente di AWS Identity and Access Management . La sezione relativa al riferimento alle policy JSON IAM della guida ha una sintassi dettagliata, descrizioni ed esempi di elementi, variabili e logica di valutazione delle policy JSON in IAM.

La policy di esempio seguente applica due restrizioni basate su tag per operazioni in `ThingGroup`. Un utente IAM limitato da questa policy:

- Non può creare un gruppo di oggetti con il tag "env=prod" (nell'esempio, consulta la riga `"aws:RequestTag/env" : "prod"`).
- Non può modificare o accedere a un gruppo di oggetti con un tag "env=prod" esistente (nell'esempio, consulta la riga `"aws:ResourceTag/env" : "prod"`).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "iot:CreateThingGroup",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:UpdateThingGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "prod"
        }
      }
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:UpdateThingGroup"
      ],
      "Resource": "*"
    }
  ]
}
```

È anche possibile specificare più valori di tag per una determinata chiave tag racchiudendoli in un elenco, come segue:

```
          "StringEquals" : {
            "aws:ResourceTag/env" : ["dev", "test"]
          }
```

> **Note**
>
> Se consenti o neghi a un utente l'accesso a risorse in base ai tag, devi considerare esplicitamente di negare agli utenti la possibilità di aggiungere o rimuovere tali tag dalle stesse risorse. In caso contrario, un utente può eludere le restrizioni e ottenere l'accesso a una risorsa modificandone i tag.

Gruppi di fatturazione

AWS IoT non ti consente di applicare direttamente i tag a singoli elementi, ma ti consente di inserire elementi in gruppi di fatturazione e di applicare tag a questi. Infatti AWS IoT, l'allocazione dei dati sui costi e sull'utilizzo in base ai tag è limitata ai gruppi di fatturazione.

AWS IoT Core per le risorse LoRa WAN, come dispositivi wireless e gateway, non possono essere aggiunte ai gruppi di fatturazione. Tuttavia, possono essere associati a AWS IoT elementi, che possono essere aggiunti ai gruppi di fatturazione.

Sono disponibili i seguenti comandi:

- AddThingToBillingGroup aggiunge un oggetto a un gruppo di fatturazione.
- CreateBillingGroup crea un gruppo di fatturazione.
- DeleteBillingGroup elimina il gruppo di fatturazione.
- DescribeBillingGroup restituisce informazioni su un gruppo di fatturazione.
- ListBillingGroups elenca i gruppi di fatturazione creati.
- ListThingsInBillingGroup elenca gli oggetti che hai aggiunto al gruppo di fatturazione specificato.
- RemoveThingFromBillingGroup rimuove l'oggetto specificato dal gruppo di fatturazione.
- UpdateBillingGroup aggiorna le informazioni sul gruppo di fatturazione.
- CreateThing consente di specificare un gruppo di fatturazione per l'oggetto al momento della creazione.
- DescribeThing restituisce la descrizione di un oggetto, incluso il gruppo di fatturazione a cui appartiene l'oggetto, se presente.

L'API AWS IoT Wireless fornisce queste azioni per associare dispositivi e gateway wireless agli AWS IoT oggetti.

- AssociateWirelessDeviceWithThing
- AssociateWirelessGatewayWithThing

Visualizzazione dell'allocazione dei costi e dei dati di utilizzo

Puoi utilizzare i tag dei gruppi di fatturazione per classificare e monitorare i costi. Quando applichi i tag ai gruppi di fatturazione (e quindi agli elementi che includono), AWS genera un rapporto sull'allocazione dei costi come file con valori separati da virgole (CSV) con l'utilizzo e i costi aggregati dai tag. Puoi applicare i tag che rappresentano categorie di business (come centri di costo, nomi di applicazioni o proprietari) per organizzare i costi tra più servizi. Per ulteriori informazioni sull'utilizzo dei tag per l'allocazione dei costi, consulta la sezione Utilizzo di tag per l'allocazione dei costi della Guida per l'utente alla gestione fatturazione e costi di AWS.

> **Note**
>
> Per associare in modo preciso i dati sull'utilizzo e sui costi agli oggetti inseriti nei gruppi di fatturazione, ogni dispositivo o applicazione deve:

- Fatti registrare come oggetto in. AWS IoT Per ulteriori informazioni, consulta Gestione dei dispositivi con AWS IoT.
- Connect al broker di AWS IoT messaggi tramite MQTT utilizzando solo il nome dell'oggetto come ID client. Per ulteriori informazioni, consulta the section called “Protocolli di dispositivo di comunicazione”. Se l'ID client non corrisponde al nome dell'oggetto, puoi abilitare l'allegato esclusivo dell'oggetto per stabilire l'associazione. Per ulteriori informazioni, consulta ???.
- Eseguire l'autenticazione utilizzando un certificato client associato all'oggetto.

Per i gruppi di fatturazione sono disponibili le seguenti dimensioni di prezzo (in base alle attività degli oggetti associati al gruppo di fatturazione):

- Connettività (in base al nome dell'oggetto utilizzato come ID client per la connessione).
- Messaggistica (in base ai messaggi in entrata e in uscita da e verso un oggetto; solo MQTT).
- Operazioni shadow (in base all'oggetto il cui messaggio ha attivato un aggiornamento shadow).
- Regole attivate (in base all'oggetto il cui messaggio in entrata ha attivato la regola; non si applica alle regole attivate da eventi del ciclo di vita MQTT).
- Aggiornamenti indice dell'oggetto (in base all'oggetto aggiunto all'indice).
- Azioni remote (in base all'oggetto aggiornato).
- AWS IoT Device Defender rileva i report (in base all'oggetto la cui attività viene segnalata).

I dati sui costi e sull'utilizzo basati sui tag (e riportati per un gruppo di fatturazione) non riflettono le seguenti attività:

- Operazioni di registro del dispositivo (inclusi gli aggiornamenti di oggetti, gruppi di oggetti e tipi di oggetti). Per ulteriori informazioni, vedere Gestione dei dispositivi con AWS IoT).
- Aggiornamenti indice gruppo oggetti (quando si aggiunge un gruppo di oggetti).
- Query di ricerca indice.
- Provisioning dei dispositivi.
- AWS IoT Device Defender rapporti di audit.

Sicurezza in AWS IoT

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei programmi di conformitàAWS. Per ulteriori informazioni sui programmi di conformità applicabili AWS IoT, consulta AWS Services in Scope by Compliance Program.
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. Inoltre, sei responsabile anche di altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda e le leggi e le normative applicabili.

Questa documentazione ti aiuta a capire come applicare il modello di responsabilità condivisa durante l'utilizzo AWS IoT. I seguenti argomenti mostrano come configurare per AWS IoT soddisfare gli obiettivi di sicurezza e conformità. Imparerai anche a utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere AWS IoT le tue risorse.

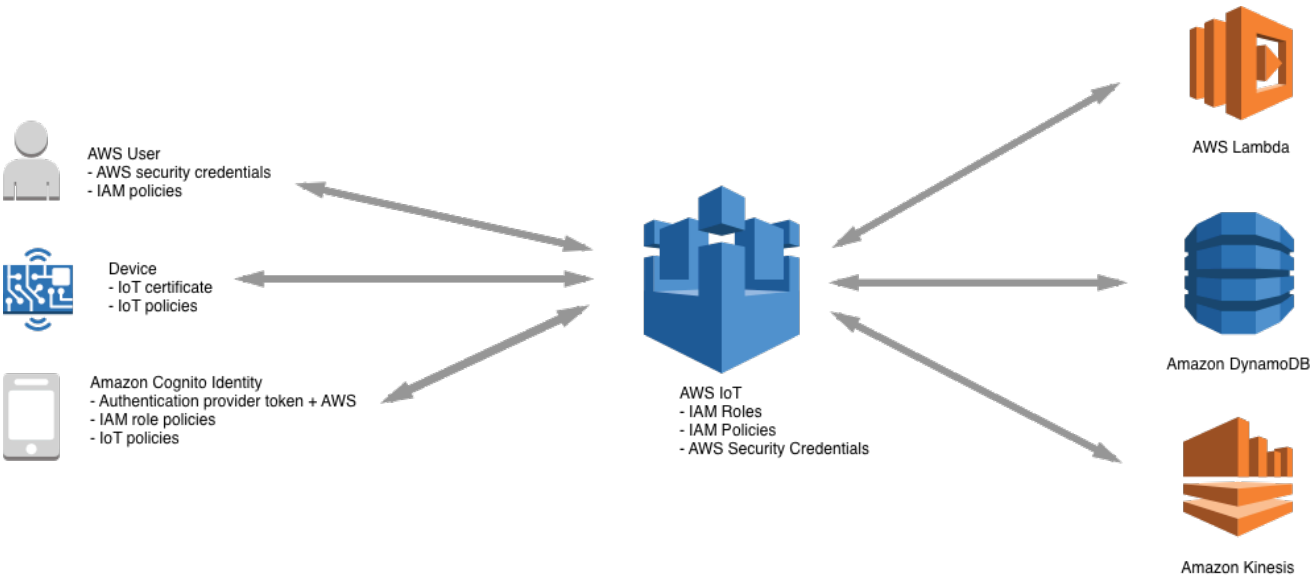
Argomenti

- AWS IoT sicurezza
- Autenticazione
- Autorizzazione
- Protezione dei dati in AWS IoT Core
- Gestione delle identità e degli accessi per AWS IoT
- Registrazione e monitoraggio
- Convalida della conformità per Core AWS IoT
- Resilienza in AWS IoT Core
- Utilizzo AWS IoT Core con gli endpoint VPC dell'interfaccia
- Sicurezza dell'infrastruttura in AWS IoT

- Monitoraggio della sicurezza delle flotte o dei dispositivi di produzione con Core AWS IoT
- Le migliori pratiche di sicurezza in AWS IoT Core
- AWS formazione e certificazione

AWS IoT sicurezza

Ogni dispositivo o client connesso deve disporre di una credenziale per interagire con AWS IoT. Tutto il traffico da e verso AWS IoT viene inviato in modo sicuro tramite Transport Layer Security (TLS). AWS i meccanismi di sicurezza cloud proteggono i dati mentre si spostano tra servizi AWS IoT e altri AWS servizi.



- L'utente è responsabile della gestione delle credenziali del dispositivo (certificati X.509, credenziali dei servizi AWS , Amazon Cognito, identità, identità federate o token di autenticazione personalizzati) e delle policy in AWS IoT. L'utente è inoltre responsabile dell'assegnazione di identità univoche a ciascun dispositivo e della gestione delle autorizzazioni per ogni dispositivo o gruppo di dispositivi.
- I tuoi dispositivi si connettono AWS IoT utilizzando certificati X.509 o identità Amazon Cognito tramite una connessione TLS sicura. Durante la ricerca e lo sviluppo e per alcune applicazioni che effettuano chiamate o utilizzano API WebSockets, puoi anche autenticarti utilizzando utenti e gruppi IAM o token di autenticazione personalizzati. Per ulteriori informazioni, consulta Utenti, gruppi e ruoli IAM.
- Quando si utilizza AWS IoT l'autenticazione, il broker di messaggi è responsabile dell'autenticazione dei dispositivi, dell'acquisizione sicura dei dati dei dispositivi e della concessione

o negazione delle autorizzazioni di accesso specificate per i dispositivi utilizzando le policy. AWS IoT

- Quando si utilizza l'autenticazione personalizzata, un autorizzatore personalizzato è responsabile dell'autenticazione dei dispositivi e della concessione o negazione delle autorizzazioni di accesso specificate per i dispositivi che utilizzano le nostre politiche IAM. AWS IoT
- Il motore AWS IoT delle regole inoltra i dati del dispositivo ad altri dispositivi o altri AWS servizi in base alle regole definite dall'utente. Viene utilizzato AWS Identity and Access Management per trasferire in modo sicuro i dati alla destinazione finale. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per AWS IoT.

Autenticazione

L'autenticazione è un meccanismo in cui si verifica l'identità di un client o di un server. L'autenticazione del server è il processo in cui i dispositivi o altri client assicurano che stiano comunicando con un endpoint effettivo AWS IoT . L'autenticazione del client è il processo con cui i dispositivi o altri client si autenticano. AWS IoT

Panoramica sui certificati X.509

I certificati X.509 sono certificati digitali che usano lo standard di infrastruttura a chiave pubblica X.509 per associare una chiave pubblica a un'identità contenuta in un certificato. I certificati X.509 vengono rilasciati da un'entità attendibile denominata autorità di certificazione (CA). L'autorità di certificazione mantiene uno o più certificati speciali chiamati certificati CA, che usa per rilasciare certificati X.509. Solo l'autorità di certificazione ha accesso a certificati CA. Le catene di certificati X.509 vengono utilizzate sia per l'autenticazione server da parte dei client sia per l'autenticazione client da parte del server.

Autenticazione del server

Quando il dispositivo o un altro client tenta di connettersi AWS IoT Core, il AWS IoT Core server invierà un certificato X.509 utilizzato dal dispositivo per autenticare il server. L'autenticazione avviene a livello TLS tramite la convalida della Catena di certificati X.509. Si tratta dello stesso metodo utilizzato dal browser quando si visita un URL HTTPS. Per utilizzare certificati della tua certification authority, consulta Gestire i certificati CA personali.

Quando i tuoi dispositivi o altri client stabiliscono una connessione TLS a un AWS IoT Core endpoint, AWS IoT Core presenta una catena di certificati che i dispositivi utilizzano per verificare che stiano

comunicando e non che un altro server si spacci per impersonare. AWS IoT Core AWS IoT Core La catena presentata dipende da una combinazione del tipo di endpoint a cui il dispositivo si connette e dalla suite di crittografia negoziata dal client durante l'handshake TLS. AWS IoT Core

Tipi di endpoint

AWS IoT Core supporti. `iot:Data-ATS` `iot:Data-ATS`gli endpoint presentano un certificato server firmato da una CA di Amazon Trust Services.

I certificati presentati dagli endpoint ATS hanno la firma incrociata di Starfield. Alcune implementazioni client TLS richiedono la convalida root trust e richiedono che i certificati CA Starfield siano installati negli archivi attendibili del client.

> **Warning**
>
> Non è consigliabile utilizzare un metodo di aggiunta dei certificati che esegue l'hash dell'intero certificato (incluso il nome dell'emittente e così via) perché ciò causerà un errore di verifica del certificato e i certificati ATS forniti hanno la firma incrociata di Starfield e un nome di emittente diverso.

> **Important**
>
> Usa gli `iot:Data-ATS` endpoint. I certificati Symantec e Verisign sono obsoleti e non sono più supportati da. AWS IoT Core

Puoi utilizzare il comando `describe-endpoint` per creare l'endpoint ATS.

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

Il comando `describe-endpoint` restituisce un endpoint nel formato seguente.

```
account-specific-prefix.iot.your-region.amazonaws.com
```

> **Note**
>
> La prima volta che viene chiamato `describe-endpoint`, viene creato un endpoint. Tutte le chiamate successive a `describe-endpoint` restituiscono lo stesso endpoint.

> **Note**
>
> Per visualizzare il tuo **`iot:Data-ATS`** endpoint nella console, scegli Impostazioni. AWS IoT Core La console visualizza solo l'endpoint `iot:Data-ATS`.

Creazione di un file **`IotDataPlaneClient`** con l' AWS SDK for Java

Per creare un dispositivo `IotDataPlaneClient` che utilizza un `iot:Data-ATS` endpoint, devi fare quanto segue.

- Crea un `iot:Data-ATS` endpoint utilizzando l'DescribeEndpointAPI.
- Specifica l'endpoint quando crei il `IotDataPlaneClient`.

Nell'esempio seguente vengono eseguite entrambe queste operazioni.

```
public void setup() throws Exception {
        IotClient client =
 IotClient.builder().credentialsProvider(CREDENTIALS_PROVIDER_CHAIN).region(Region.US_EAST_1).b
        String endpoint = client.describeEndpoint(r -> r.endpointType("iot:Data-
ATS")).endpointAddress();
        iot = IotDataPlaneClient.builder()
                                .credentialsProvider(CREDENTIALS_PROVIDER_CHAIN)
                                .endpointOverride(URI.create("https://" + endpoint))
                                .region(Region.US_EAST_1)
                                .build();
}
```

Certificati CA per l'autenticazione del server

A seconda del tipo di endpoint di dati in uso e della suite di crittografia negoziata, i certificati di autenticazione AWS IoT Core del server sono firmati da uno dei seguenti certificati CA root:

Amazon Trust Services Endpoint (preferito)

> **Note**
>
> Potrebbe essere necessario fare clic con il pulsante destro del mouse su questi collegamenti e selezionare Salva collegamento come... per salvare questi certificati come file.

- Chiave RSA a 2048 bit: Amazon Root CA 1.
- Chiave RSA a 4096 bit: Amazon Root CA 2. Riservato per uso futuro.
- Chiave ECC a 256 bit: Amazon Root CA 3.
- Chiave ECC a 384 bit: Amazon Root CA 4. Riservato per uso futuro.

Questi certificati hanno tutti la firma incrociata di Starfield Root CA Certificate. Tutte le nuove AWS IoT Core regioni, a partire dal lancio del 9 maggio 2018 AWS IoT Core nella regione Asia Pacifico (Mumbai), offrono solo certificati ATS.

VeriSign Endpoint (legacy)

- Chiave RSA a 2048 bit: certificato CA principale G5 pubblico primario di VeriSign classe 3

Linee guida per l'autenticazione del server

Esistono molte variabili che possono influire sulla capacità di un dispositivo di convalidare il certificato di autenticazione del server AWS IoT Core . Ad esempio, i dispositivi potrebbero avere una quantità eccessiva di memoria per contenere tutti i possibili certificati CA radice oppure i dispositivi possono implementare un metodo non standard di convalida del certificato. Per questi motivi suggeriamo di seguire queste linee guida:

- Ti consigliamo di utilizzare il tuo endpoint ATS e di installare tutti i certificati Amazon Root CA supportati.
- Se non riesci ad archiviare tutti questi certificati sul tuo dispositivo e se i tuoi dispositivi non utilizzano la convalida basata su ECC, puoi omettere i certificati ECC Amazon Root CA 3 e Amazon Root CA 4. Se i dispositivi non implementano la convalida dei certificati basata su RSA, puoi omettere i certificati RSA di Amazon Root CA 1 e di Amazon Root CA 2. Potrebbe essere necessario fare clic con il pulsante destro del mouse su questi collegamenti e selezionare Salva collegamento come... per salvare questi certificati come file.

- Se riscontri problemi di convalida dei certificati server durante la connessione all'endpoint ATS, prova ad aggiungere il certificato Amazon Root CA pertinente con firma incrociata al tuo archivio attendibile. Potrebbe essere necessario fare clic con il pulsante destro del mouse su questi collegamenti e selezionare Salva collegamento come... per salvare questi certificati come file.
  - Firma incrociata Amazon Root CA 1
  - Autorità di certificazione Amazon Root CA 2 a firma incrociata – Riservata per uso futuro.
  - Firma incrociata Amazon Root CA 3
  - Amazon Root CA 4 a firma incrociata - Riservata per uso futuro.
- Se si verificano problemi di convalida dei certificati server, il dispositivo potrebbe considerare esplicitamente attendibile la CA principale. Prova ad aggiungere Starfield Root CA Certificate al tuo negozio di fiducia.
- Se si verificano ancora problemi dopo aver eseguito i passaggi precedenti, contattare il Supporto AWS Developer.

> **Note**
>
> I certificati CA hanno una data di scadenza dopo la quale non possono più essere usati per convalidare un certificato del server. Potrebbe essere necessario sostituire i certificati CA prima della data di scadenza. Assicurati di poter aggiornare i certificati CA root in tutti i dispositivi o client per garantire la connessione continua e per mantenere il sistema aggiornato con le best practice di sicurezza.

> **Note**
>
> Quando ti connetti AWS IoT Core al codice del tuo dispositivo, trasferisci il certificato all'API che stai utilizzando per la connessione. L'API utilizzata varierà in base all'SDK. Per ulteriori informazioni, consulta il AWS IoT Core dispositivo SDKs.

Autenticazione client

AWS IoT supporta tre tipi di principi di identità per l'autenticazione del dispositivo o del client:

- Certificati client X.509

- Utenti, gruppi e ruoli IAM
- Identità Amazon Cognito

Queste identità possono essere utilizzate con dispositivi, applicazioni mobili, Web o desktop. Possono anche essere utilizzati da un utente che digita i comandi dell'interfaccia a riga di AWS IoT comando (CLI). In genere, AWS IoT i dispositivi utilizzano certificati X.509, mentre le applicazioni mobili utilizzano identità Amazon Cognito. Le applicazioni Web e desktop usano le identità IAM o federate. I comandi AWS CLI usano IAM. Per ulteriori informazioni sulle identità IAM, consulta Gestione delle identità e degli accessi per AWS IoT.

Certificati client X.509

I certificati X.509 offrono la possibilità di AWS IoT autenticare le connessioni di client e dispositivi. I certificati client devono essere registrati AWS IoT prima che un client possa comunicare con. AWS IoT Un certificato client può essere registrato in più Account AWS file contemporaneamente Regione AWS per facilitare lo spostamento di dispositivi tra Account AWS i server della stessa regione. Per ulteriori informazioni, consulta Utilizzo dei certificati client X.509 in più Account AWS secondi con registrazione multiaccount.

Consigliamo che a ogni dispositivo o client sia assegnato un certificato univoco per permettere operazioni di gestione granulare del client, inclusa la revoca di certificati. I dispositivi e i client devono anche supportare la rotazione e la sostituzione dei certificati per garantire un funzionamento corretto allo scadere dei certificati.

Per informazioni sull'utilizzo dei certificati X.509 per supportare più dispositivi, consulta Provisioning dei dispositivi per esaminare le diverse opzioni di gestione dei certificati e provisioning supportate da AWS IoT .

AWS IoT supporta questi tipi di certificati client X.509:

- certificati X.509 generati da AWS IoT
- Certificati X.509 firmati da una CA registrata con. AWS IoT
- Certificati X.509 firmati da una CA non registrata con AWS IoT.

In questa sezione viene descritto come gestire i certificati X.509 in AWS IoT. È possibile utilizzare la AWS IoT console o AWS CLI eseguire le seguenti operazioni sui certificati:

- Crea certificati AWS IoT client

- Creare certificati client personali
- Registrare un certificato client
- Attivare o disattivare un certificato client
- Revocare un certificato client

Per ulteriori informazioni sui AWS CLI comandi che eseguono queste operazioni, consulta AWS IoT CLI Reference.

Utilizzo dei certificati client X.509

I certificati X.509 autenticano le connessioni di client e dispositivi a. AWS IoT I certificati X.509 offrono diversi vantaggi rispetto ad altri meccanismi di identificazione e autenticazione. I certificati X.509 permettono l'uso di chiavi asimmetriche con i dispositivi. Ad esempio, è possibile masterizzare le chiavi private in un dispositivo di storage sicuro in modo che il materiale crittografico sensibile resta sempre nel dispositivo. I certificati X.509 forniscono un'autenticazione client più affidabile rispetto ad altri schemi, ad esempio un nome utente e una password o un token di connessione, perché la chiave privata resta sempre nel dispositivo.

AWS IoT autentica i certificati client utilizzando la modalità di autenticazione client del protocollo TLS. Il supporto TLS è disponibile in molti linguaggi di programmazione e sistemi operativi e viene usato in genere per crittografare i dati. Nell'autenticazione client TLS, AWS IoT richiede un certificato client X.509 e convalida lo stato del certificato e confrontandolo con un registro di certificati. Account AWS Quindi richiede al client la prova della proprietà della chiave privata che corrisponde alla chiave pubblica contenuta nel certificato. AWS IoT richiede ai client di inviare l'estensione SNI (Server Name Indication) al protocollo Transport Layer Security (TLS). Per ulteriori informazioni sulla configurazione dell'estensione SNI, consulta Sicurezza dei trasporti in AWS IoT Core.

Per facilitare una connessione client sicura e coerente al AWS IoT core, un certificato client X.509 deve possedere quanto segue:

- Registrato in Core. AWS IoT Per ulteriori informazioni, consulta Registrare un certificato client.
- Avere uno stato di `ACTIVE`. Per ulteriori informazioni, consulta Attivare o disattivare un certificato client.
- La data di scadenza del certificato non è ancora stata raggiunta.

Puoi creare certificati client che utilizzano l'autorità di certificazione root Amazon e puoi utilizzare i tuoi certificati client firmati da un'altra CA. Per ulteriori informazioni sull'utilizzo della AWS IoT console

per creare certificati che utilizzano Amazon Root CA, consultaCrea certificati AWS IoT client. Per ulteriori informazioni sull'utilizzo dei certificati X.509, consulta Creare certificati client personali.

La data e l'ora di scadenza per i certificati firmati da un certificato CA vengono impostate alla creazione del certificato. I certificati X.509 generati da AWS IoT scadono alla mezzanotte UTC del 31 dicembre 2049 (2049-12-31T 23:59:59 Z).

AWS IoT Device Defender può eseguire audit su te Account AWS e sui dispositivi che supportano le migliori pratiche di sicurezza IoT comuni. Ciò include la gestione delle date di scadenza dei certificati X.509 firmati dalla tua CA o da Amazon Root CA. Per ulteriori informazioni sulla gestione della data di scadenza di un certificato, consulta Scadenza del certificato del dispositivo e Scadenza del certificato CA.

Nel AWS IoT blog ufficiale, un approfondimento sulla gestione della rotazione dei certificati dei dispositivi e sulle migliori pratiche di sicurezza è esplorato in Come gestire la rotazione dei certificati dei dispositivi IoT utilizzando AWS IoT.

Utilizzo dei certificati client X.509 in più Account AWS secondi con registrazione multiaccount

La registrazione a più account consente di spostare dispositivi tra i tuoi Account AWS nella stessa regione o in regioni diverse. In questo modo puoi registrare, testare e configurare un dispositivo in un account di pre-produzione, quindi registrare e utilizzare lo stesso dispositivo e certificato del dispositivo in un account di produzione. È inoltre possibile registrare il certificato client sul dispositivo o i certificati del dispositivo senza una CA registrata con. AWS IoT Per ulteriori informazioni, consulta Registrazione di un certificato client firmato da una CA non registrata (CLI).

> **Note**
>
> I certificati utilizzati per la registrazione con più account sono supportati in `iot:Data-ATS`, `iot:Data` (legacy), `iot:Jobs` e nei tipi di endpoint `iot:CredentialProvider`. Per ulteriori informazioni sugli endpoint dei AWS IoT dispositivi, consultaAWS IoT dati del dispositivo e endpoint di servizio.

I dispositivi che utilizzano la registrazione multiaccount devono inviare l'estensione SNI (Server Name Indication) al protocollo Transport Layer Security (TLS) e fornire l'indirizzo completo dell'endpoint sul `host_name` campo, quando si connettono. AWS IoT AWS IoT utilizza l'indirizzo dell'endpoint in `host_name` per indirizzare la connessione all'account corretto. AWS IoT I dispositivi esistenti che non inviano un indirizzo endpoint valido in `host_name` continueranno a funzionare, ma non

saranno in grado di utilizzare le funzionalità che richiedono queste informazioni. Per ulteriori informazioni sull'estensione SNI e per informazioni su come identificare l'indirizzo endpoint per il campo `host_name`, vedere Sicurezza dei trasporti in AWS IoT Core.

Per utilizzare la registrazione con più account

1. Puoi registrare i certificati del dispositivo con una CA. Puoi registrare la CA di firma in più account in modalità `SNI_ONLY` e utilizzare tale CA per registrare lo stesso certificato client su più account. Per ulteriori informazioni, consulta Registrazione di un certificato CA in modalità SNI_ONLY (CLI) - Consigliato.
2. Puoi registrare i certificati del dispositivo senza una CA. Per informazioni, consulta Registra un certificato client firmato da una CA (CLI) non registrata. La registrazione di una CA è facoltativa. Non è necessario registrare la CA con AWS IoT cui ha firmato i certificati del dispositivo.

Algoritmi di firma dei certificati supportati da AWS IoT

AWS IoT supporta i seguenti algoritmi di firma dei certificati:

- SHA256CON RSA
- SHA384CON RSA
- SHA512CON RSA
- SHA256WITHRSAANDMGF1 (RASSA-PSS)
- SHA384WITHRSAANDMGF1 (RASSA-PSS)
- SHA512WITHRSAANDMGF1 (RASSA-PSS)
- DSA_CON_ SHA256
- ECDSA-CON- SHA256
- ECDSA-CON- SHA384
- ECDSA-CON- SHA512

Per ulteriori informazioni sull'autenticazione e la sicurezza dei certificati, vedere Device certificate key quality.

> **Note**
>
> La richiesta di firma del certificato (CSR) deve includere una chiave pubblica. La chiave può essere una chiave RSA con lunghezza di almeno 2048 bit o una chiave ECC da

curva NIST P–256, NIST P–384 o NIST P-521. Per ulteriori informazioni, consulta CreateCertificateFromCsr nella guida di riferimento delle API AWS IoT .

Algoritmi chiave supportati da AWS IoT

La tabella seguente mostra come sono supportati gli algoritmi chiave:

Algoritmo chiave	Algoritmo di firma dei certificati	Versione TLS	Supportato? Sì o No
RSA con una dimensione della chiave di almeno 2048 bit	Tutti	TLS 1.2 TLS 1.3	Sì
ECC NIST P-256/P-384/ P-521	Tutti	TLS 1.2 TLS 1.3	Sì
RSA-PSS con una dimensione della chiave di almeno 2048 bit	Tutti	TLS 1.2	No
RSA-PSS con una dimensione della chiave di almeno 2048 bit	Tutti	TLS 1.3	Sì

Per creare un certificato utilizzando CreateCertificateFromCSR, puoi utilizzare un algoritmo a chiave supportato per generare una chiave pubblica per la tua CSR. Per registrare il proprio certificato utilizzando la RegisterCertificatenostra RegisterCertificateWithoutCA, è possibile utilizzare un algoritmo a chiave supportato per generare una chiave pubblica per il certificato.

Per ulteriori informazioni, consulta Politiche di sicurezza.

Crea certificati AWS IoT client

AWS IoT fornisce certificati client firmati dall'autorità di certificazione Amazon Root (CA).

In questo argomento viene descritto come creare un certificato client firmato dall'autorità di certificazione root Amazon e scaricare i file del certificato. Dopo aver creato i file di certificato client, è necessario installarli nel client.

> **Note**
>
> Ogni certificato client X.509 fornito da AWS IoT contiene gli attributi di emittente e soggetto impostati al momento della creazione del certificato. Gli attributi del certificato sono immutabili solo dopo la creazione del certificato.

Puoi utilizzare la AWS IoT console o il AWS CLI per creare un AWS IoT certificato firmato dall'autorità di certificazione Amazon Root.

Crea un AWS IoT certificato (console)

Per creare un AWS IoT certificato utilizzando la AWS IoT console

1. Accedi a Console di gestione AWS e apri la AWS IoT console.
2. Nel riquadro di navigazione, selezionare Sicurezza, scegliere Certificati, quindi Crea.
3. Scegliere One-click certificate creation (Creazione certificato con un clic) – Create certificate (Crea certificato).
4. Dalla pagina Certificato creato scarica i file del certificato client per l'oggetto, la chiave pubblica e la chiave privata in una posizione sicura. Questi certificati generati da AWS IoT sono disponibili solo per l'uso con AWS IoT i servizi.

   Se è necessario anche il file del certificato dell'autorità di certificazione root Amazon, questa pagina contiene anche il collegamento alla pagina da cui è possibile scaricarlo.
5. A questo punto, un certificato client è stato creato e registrato con AWS IoT. È necessario attivare il certificato prima di utilizzarlo in un client.

   Scegli Attiva per attivare subito il certificato client. Se non desideri attivare il certificato, in Attivare un certificato client (console) viene descritto come attivare il certificato in un secondo momento.
6. Se desideri collegare una policy al certificato, scegli Attach a policy (Collega una policy).

   Se non desideri collegare una policy, scegli Done (Fatto) per terminare. È possibile collegare una policy in un secondo momento.

Dopo aver completato la procedura, installare i file di certificato sul client.

Creare un AWS IoT certificato (CLI)

AWS CLI fornisce il create-keys-and-certificate comando per creare certificati client firmati dall'autorità di certificazione Amazon Root. Questo comando però non scarica il file del certificato dell'autorità di certificazione root Amazon. Puoi scaricare il file del certificato dell'autorità di certificazione root Amazon da Certificati CA per l'autenticazione del server.

Questo comando crea file e registri con chiave privata, chiave pubblica e certificati X.509 e attiva il certificato con. AWS IoT

```
aws iot create-keys-and-certificate \
    --set-as-active \
    --certificate-pem-outfile certificate_filename.pem \
    --public-key-outfile public_filename.key \
    --private-key-outfile private_filename.key
```

Se non desideri attivare il certificato durante la creazione e la registrazione, questo comando crea file di certificato di chiave privata, chiave pubblica e X.509 e registra il certificato, ma non lo attiva. In Attivare un certificato client (CLI) viene descritto come attivare il certificato in un secondo momento.

```
aws iot create-keys-and-certificate \
    --no-set-as-active \
    --certificate-pem-outfile certificate_filename.pem \
    --public-key-outfile public_filename.key \
    --private-key-outfile private_filename.key
```

Installa i file di certificato sul client.

Creare certificati client personali

AWS IoT supporta i certificati client firmati da qualsiasi autorità di certificazione (CA) principale o intermedia. AWS IoT utilizza i certificati CA per verificare la proprietà dei certificati. Per utilizzare i certificati dei dispositivi firmati da una CA che non è la CA di Amazon, è necessario registrare il certificato della CA AWS IoT in modo da poter verificare la proprietà del certificato del dispositivo.

AWS IoT supporta diversi modi per portare i propri certificati (BYOC):

- Innanzitutto, registrare la CA utilizzata per firmare i certificati client e quindi registrare i singoli certificati client. Se si desidera registrare il dispositivo o il client nel relativo certificato client alla prima connessione AWS IoT (noto anche come Just-in-Time Provisioning), è necessario registrare la CA firmataria con AWS IoT e attivare la registrazione automatica.
- Se non è possibile registrare la CA di firma, è possibile scegliere di registrare i certificati client senza CA. Per i dispositivi registrati senza CA, è necessario presentare la Server Indication (SNI) quando si connettono a AWS IoT.

> **Note**
>
> Per registrare i certificati client utilizzando CA, è necessario registrare la CA firmataria con cui si effettua la firma AWS IoT, non con nessun'altra CAs nella gerarchia.

> **Note**
>
> Un certificato CA può essere registrato in modalità `DEFAULT` da un solo account in una regione. Un certificato CA può essere registrato in modalità `SNI_ONLY` da più account in una regione.

Per ulteriori informazioni sull'utilizzo dei certificati X.509 per supportare più dispositivi, consulta Provisioning dei dispositivi per esaminare le diverse opzioni di gestione dei certificati e provisioning supportate da AWS IoT .

Argomenti

- Gestire i certificati CA personali
- Creare un certificato client utilizzando il certificato CA

Gestire i certificati CA personali

In questa sezione vengono descritte le attività comuni per la gestione dei certificati dell'autorità di certificazione (CA).

Puoi registrare la tua autorità di certificazione (CA) con AWS IoT se utilizzi certificati client firmati da una CA che AWS IoT non riconosce.

Se desideri che i client registrino automaticamente i propri certificati client al AWS IoT momento della prima connessione, è necessario registrare la CA che ha firmato i certificati client AWS IoT. In caso contrario, non è necessario registrare il certificato CA che ha firmato i certificati client.

> **Note**
>
> Un certificato CA può essere registrato in modalità `DEFAULT` da un solo account in una regione. Un certificato CA può essere registrato in modalità `SNI_ONLY` da più account in una regione.

Argomenti:

- Creare un certificato CA
- Registrare il certificato CA
- Disattivare un certificato CA

Creare un certificato CA

Se non disponi di un certificato CA, puoi usare gli strumenti OpenSSL v1.1.1i per crearne uno.

> **Note**
>
> Non è possibile eseguire questa procedura nella AWS IoT console.

Per creare un certificato CA utilizzando gli strumenti OpenSSL v1.1.1i

1. Genera una coppia di chiavi.

```
openssl genrsa -out root_CA_key_filename.key 2048
```

2. Usa la chiave privata della coppia di chiavi per generare un certificato CA.

```
openssl req -x509 -new -nodes \
    -key root_CA_key_filename.key \
    -sha256 -days 1024 \
    -out root_CA_cert_filename.pem
```

Registrare il certificato CA

Queste procedure descrivono come registrare un certificato rilasciato da un'autorità di certificazione (CA) diversa dalla CA di Amazon. AWS IoT Core utilizza i certificati CA per verificare la proprietà dei certificati. Per utilizzare i certificati dei dispositivi firmati da una CA che non è la CA di Amazon, devi registrare il certificato CA AWS IoT Core in modo che possa verificare la proprietà del certificato del dispositivo.

Registrare un certificato CA (console)

> **Note**
>
> Per registrare un certificato emesso da una CA nella console, avvia la console all'indirizzo Register CA certificate (Registra certificato emesso da una CA). Puoi registrare la tua CA in modalità multi-account e senza la necessità di fornire un certificato di verifica o accedere alla chiave privata. Una CA può essere registrata in modalità multi-account da più Account AWS nella stessa Regione AWS. Puoi registrare la tua CA in modalità account singolo fornendo un certificato di verifica e una verifica della proprietà della chiave privata della CA.

Registrare un certificato CA (CLI)

Puoi registrare un certificato CA in modalità `DEFAULT` o in modalità `SNI_ONLY`. Una CA può essere registrata in `DEFAULT` modalità una Account AWS alla volta Regione AWS. Una CA può essere registrata in `SNI_ONLY` modalità da più Account AWS persone contemporaneamente Regione AWS. Per ulteriori informazioni sulla modalità dei certificati CA, consulta certificateMode.

> **Note**
>
> Ti consigliamo di registrare una CA in modalità `SNI_ONLY`. Non è necessario fornire un certificato di verifica o l'accesso alla chiave privata, ed è possibile registrare la CA più volte Account AWS contemporaneamente Regione AWS.

Registrazione di un certificato CA in modalità SNI_ONLY (CLI) - Consigliato

Prerequisiti

Verifica che i seguenti elementi siano presenti sul tuo computer prima di continuare:

- Il file del certificato CA root (a cui si fa riferimento nel seguente esempio come *root_CA_cert_filename.pem*)
- OpenSSL v1.1.1i o versione successiva

Per registrare un certificato CA in **SNI_ONLY** modalità, utilizzare il AWS CLI

1. Registrare il certificato CA con AWS IoT. Utilizzando il comando register-ca-certificate, immetti il nome del file del certificato CA. Per ulteriori informazioni, consulta register-ca-certificate nella documentazione di riferimento dei comandi della AWS CLI .

   ```
   aws iot register-ca-certificate \
       --ca-certificate file://root_CA_cert_filename.pem \
       --certificate-mode SNI_ONLY
   ```

   In caso di successo, questo comando restituisce*certificateId*.
2. A questo punto, il certificato CA è stato registrato con AWS IoT ma non è attivo. Prima di poter registrare gli eventuali certificati client che ha firmato, il certificato CA deve essere attivato.

   Questo passaggio attiva il certificato CA.

   Per attivare il certificato CA, usa il comando update-certificate come riportato di seguito. Per ulteriori informazioni, consulta update-certificate in Riferimento ai comandi AWS CLI .

   ```
   aws iot update-ca-certificate \
       --certificate-id certificateId \
       --new-status ACTIVE
   ```

Per visualizzare lo stato del certificato CA, utilizza il comando describe-ca-certificate. Per ulteriori informazioni, consulta describe-ca-certificate nella documentazione di riferimento dei comandi della AWS CLI .

Registrazione di un certificato CA in modalità **DEFAULT** (CLI)

Prerequisiti

Verifica che i seguenti elementi siano presenti sul tuo computer prima di continuare:

- Il file del certificato CA root (a cui si fa riferimento nel seguente esempio come *root_CA_cert_filename.pem*)
- Il file della chiave privata del certificato CA root (a cui si fa riferimento nel seguente esempio come *root_CA_key_filename.key*)
- OpenSSL v1.1.1i o versione successiva

Per registrare un certificato CA in **DEFAULT** modalità utilizzando il AWS CLI

1. Per ottenere un codice di registrazione da AWS IoT, usaget-registration-code. Salvare il `registrationCode` restituito per utilizzarlo come `Common Name` del certificato di verifica della chiave privata. Per ulteriori informazioni, consulta get-registration-code nella documentazione di riferimento dei comandi della AWS CLI .

   ```
   aws iot get-registration-code
   ```

2. Genera una coppia di chiavi per il certificato di verifica della chiave privata.

   ```
   openssl genrsa -out verification_cert_key_filename.key 2048
   ```

3. Creare una richiesta di firma del certificato per il certificato di verifica della chiave privata. Impostare il campo `Common Name` del certificato sul `registrationCode` restituito da get-registration-code.

   ```
   openssl req -new \
       -key verification_cert_key_filename.key \
       -out verification_cert_csr_filename.csr
   ```

   Ti verrà richiesto di immettere alcune informazioni, tra cui il `Common Name` per il certificato.

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [AU]:
       State or Province Name (full name) []:
       Locality Name (for example, city) []:
   ```

```
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:your_registration_code
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

4. Usa la richiesta di firma del certificato per creare un certificato di verifica della chiave privata.

```
openssl x509 -req \
    -in verification_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out verification_cert_filename.pem \
    -days 500 -sha256
```

5. Registra il certificato CA con AWS IoT. Passa il nome file del certificato CA e il nome file del certificato di verifica della chiave privata al comando register-ca-certificate come riportato di seguito: Per ulteriori informazioni, consulta register-ca-certificate nella documentazione di riferimento dei comandi della AWS CLI .

```
aws iot register-ca-certificate \
    --ca-certificate file://root_CA_cert_filename.pem \
    --verification-cert file://verification_cert_filename.pem
```

   Questo comando restituisce*certificateId*, in caso di successo.

6. A questo punto, il certificato CA è stato registrato AWS IoT ma non è attivo. Prima di poter registrare gli eventuali certificati client che ha firmato, il certificato CA deve essere attivato.

   Questo passaggio attiva il certificato CA.

   Per attivare il certificato CA, usa il comando update-certificate come riportato di seguito. Per ulteriori informazioni, consulta update-certificate in Riferimento ai comandi AWS CLI .

```
aws iot update-ca-certificate \
    --certificate-id certificateId \
    --new-status ACTIVE
```

Per visualizzare lo stato del certificato CA, utilizza il comando describe-ca-certificate. Per ulteriori informazioni, consulta describe-ca-certificate nella documentazione di riferimento dei comandi della AWS CLI .

Creazione di un certificato di verifica CA per registrare il certificato emesso da una CA nella console

> **Note**
>
> Questa procedura è utile solo se si sta registrando un certificato CA dalla AWS IoT console. Se non hai eseguito questa procedura dalla AWS IoT console, avvia la procedura di registrazione del certificato CA nella console all'indirizzo Register CA certificate.

Verifica che i seguenti elementi siano presenti sullo stesso computer prima di continuare:

- Il file del certificato CA root (a cui si fa riferimento nel seguente esempio come *root_CA_cert_filename.pem*)
- Il file della chiave privata del certificato CA root (a cui si fa riferimento nel seguente esempio come *root_CA_key_filename.key*)
- OpenSSL v1.1.1i o versione successiva

Per utilizzare l'interfaccia a riga di comando per creare un certificato di verifica CA per registrare il certificato CA nella console

1. Sostituisci *verification_cert_key_filename.key* con il nome del file della chiave del certificato di verifica che desideri creare (ad esempio, **verification_cert.key**). Quindi esegui questo comando per generare una coppia di chiavi per il certificato di verifica della chiave privata:

   ```
   openssl genrsa -out verification_cert_key_filename.key 2048
   ```

2. Sostituisci *verification_cert_key_filename.key* con il nome del file della chiave creato nel passaggio 1.

   Sostituisci *verification_cert_csr_filename.csr* con il nome del file CSR (Certificate Signing Request, richiesta di firma del certificato) che desideri creare. Ad esempio, **verification_cert.csr**.

   Esegui questo comando per creare il file CSR.

```
openssl req -new \
    -key verification_cert_key_filename.key \
    -out verification_cert_csr_filename.csr
```

Il comando richiede informazioni aggiuntive che vengono spiegate in seguito.

3. Nella AWS IoT console, nel contenitore del certificato di verifica, copia il codice di registrazione.
4. Le istruzioni del comando openssl sono mostrate nell'esempio seguente. Ad eccezione del campo `Common Name`, puoi inserire valori specifici o lasciare i campi vuoti.

   Nel campo `Common Name`, incolla il codice di registrazione copiato nella fase precedente.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
    State or Province Name (full name) []:
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:your_registration_code
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

   Al termine, il comando crea il file CSR.
5. Sostituisci *verification_cert_csr_filename.csr* con il *verification_cert_csr_filename.csr* utilizzato nella fase precedente.

   Sostituisci *root_CA_cert_filename.pem* con il nome file del certificato CA che desideri registrare.

Sostituisci *root_CA_key_filename.key* con il nome file del file della chiave privata del certificato CA.

Sostituisci *verification_cert_filename.pem* con il nome del file del certificato di verifica che desideri creare. Ad esempio, **verification_cert.pem**.

```
openssl x509 -req \
    -in verification_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out verification_cert_filename.pem \
    -days 500 -sha256
```

6. Al termine del comando OpenSSL, questi file saranno pronti per l'utilizzo quando si torna nella console.

   - Il file del certificato emesso da una CA (*root_CA_cert_filename.pem* usato nel comando precedente)
   - Il certificato di verifica creato nel passaggio precedente (*verification_cert_filename.pem*utilizzato nel comando precedente)

Disattivare un certificato CA

Quando un certificato di autorità di certificazione (CA) è abilitato per la registrazione automatica dei certificati client, AWS IoT controlla il certificato CA per assicurarsi che lo sia`ACTIVE`. Se il certificato CA lo è`INACTIVE`, AWS IoT non consente la registrazione del certificato client.

L'impostazione del certificato CA su `INACTIVE`, impedisce la registrazione automatica di eventuali nuovi certificati client emessi dalla CA.

> **Note**
>
> Qualsiasi certificato client registrato che sia stato firmato dal certificato CA compromesso continuerà a funzionare finché non viene esplicitamente revocato.

Disattivare un certificato CA (console)

Per disattivare un certificato CA utilizzando la console AWS IoT

1. Accedi a Console di gestione AWS e apri la AWS IoT console.
2. Nel riquadro di navigazione a sinistra, scegli Sicuro, scegli CAs.
3. Nell'elenco delle autorità di certificazione, individua quella che desideri disattivare e scegli l'icona con i puntini di sospensione per aprire il menu delle opzioni.
4. Scegliere Deactivate (Disattiva)dal menu delle opzioni.

L'autorità di certificazione viene visualizzata come Inactive (Inattiva) nell'elenco.

> **Note**
>
> La AWS IoT console non fornisce un modo per elencare i certificati firmati dalla CA che hai disattivato. Per un'opzione AWS CLI che elenca questi certificati, consulta Disattivare un certificato CA (CLI).

Disattivare un certificato CA (CLI)

AWS CLI Fornisce il update-ca-certificatecomando per disattivare un certificato CA.

```
aws iot update-ca-certificate \
    --certificate-id certificateId \
    --new-status INACTIVE
```

Utilizza il comando list-certificates-by-ca per ottenere l'elenco di tutti i certificati client registrati firmati dalla CA specificata. Per ogni certificato client firmato dal certificato CA specificato, usa il comando update-certificate per revocare il certificato del dispositivo in modo da impedirne l'uso.

Utilizza il comando describe-ca-certificate per visualizzare lo stato del certificato CA.

Creare un certificato client utilizzando il certificato CA

È possibile utilizzare la propria autorità di certificazione (CA) per creare certificati client. Il certificato client deve essere registrato AWS IoT prima dell'uso. Per informazioni sulle opzioni di registrazione per i certificati client, consulta Registrare un certificato client.

Creare un certificato client (CLI)

> **Note**
>
> Non è possibile eseguire questa procedura nella AWS IoT console.

Per creare un certificato client utilizzando il AWS CLI

1. Genera una coppia di chiavi.

```
openssl genrsa -out device_cert_key_filename.key 2048
```

2. Creare una richiesta di firma del certificato per il certificato client.

```
openssl req -new \
    -key device_cert_key_filename.key \
    -out device_cert_csr_filename.csr
```

Ti verrà richiesto di immettere alcune informazioni, come mostrato qui.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
    State or Province Name (full name) []:
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

3. Creare un certificato client dalla richiesta di firma del certificato.

```
openssl x509 -req \
    -in device_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out device_cert_filename.pem \
    -days 500 -sha256
```

A questo punto, il certificato client è stato creato, ma non è ancora stato registrato con AWS IoT. Per informazioni su come e quando registrare il certificato client, consulta Registrare un certificato client.

Registrare un certificato client

I certificati client devono essere registrati presso AWS IoT per consentire le comunicazioni tra il client e AWS IoT. È possibile registrare ogni certificato client manualmente oppure configurare i certificati client in modo che vengano registrati automaticamente quando il client si connette AWS IoT per la prima volta.

Se desideri che i client e i dispositivi registrino i certificati client quando si connettono per la prima volta, è necessario utilizzare Registrare il certificato CA per firmare il certificato client con AWS IoT nelle regioni in cui vuoi utilizzarlo. Amazon Root CA viene registrata automaticamente con AWS IoT.

I certificati client possono essere condivisi da Account AWS e regioni. Le procedure descritte in questi argomenti devono essere eseguite in ogni account e regione in cui vuoi utilizzare il certificato client. La registrazione di un certificato client in un account o in una regione non viene riconosciuta automaticamente.

> **Note**
>
> I client che utilizzano il protocollo Transport Layer Security (TLS) per connettersi ad AWS IoT devono supportare l'estensione SNI (Server Name Indication) a TLS. Per ulteriori informazioni, consulta Sicurezza dei trasporti in AWS IoT Core.

Argomenti

- Registrare manualmente un certificato client
- Registrare un certificato client quando il client si connette alla AWS IoT just-in-time registrazione (JITR)

Registrare manualmente un certificato client

È possibile registrare manualmente un certificato client utilizzando la AWS IoT console e AWS CLI.

La procedura di registrazione da utilizzare dipende dal fatto che il certificato venga condiviso da Account AWS s e Regions. La registrazione di un certificato client in un account o in una regione non viene riconosciuta automaticamente.

Le procedure descritte in questo argomento devono essere eseguite in ogni account e regione in cui vuoi utilizzare il certificato client. I certificati client possono essere condivisi da Account AWS regioni e regioni.

Registrare un certificato client firmato da una CA registrata (console)

> **Note**
>
> Prima di eseguire questa procedura, assicurati di disporre del file.pem del certificato client e che il certificato client sia stato firmato da una CA presso cui ti sei registrato. AWS IoT

Per registrare un certificato esistente AWS IoT utilizzando la console

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione, all'interno della sezione Manage (Gestione), scegliere Security (Sicurezza) e quindi Certificates (Certificati).
3. Sulla pagina Certificates (Certificati), nella finestra di dialogo Certificates (Certificati), selezionare Add certificate (Aggiungi certificato) e quindi Register certificates (Registra certificati).
4. Sulla pagina Register certificate (Registra certificato), nella finestra di dialogo Certificates to upload (Certificati da caricare) eseguire le seguenti operazioni:
   - Scegliere CA is registered with AWS IoT (La CA è registrata presso).
   - In Choose a CA certificate (Scegliere un certificato CA), selezionare il la propria Certification authority (Autorità di certificazione).
     - Scegliere Register a new CA (Registrare una nuova CA) per registrare una nuova Certification authority (Autorità di certificazione) che non è registrata su AWS IoT.
     - Lasciare il campo Choose a CA certificate (Scegliere un certificato CA) vuoto se la propria autorità di certificazione è Amazon Root certificate authority (Autorità di certificazione Amazon Root).

- Seleziona fino a 10 certificati da caricare e con cui registrarti AWS IoT.
  - Usare i file dei certificati creati in Crea certificati AWS IoT client e Creare un certificato client utilizzando il certificato CA.
- Scegliere Activate (Attiva) o Deactivate (Disattiva). Se si sceglie Deactivate (Disattiva), Attivare o disattivare un certificato client spiega come attivare il certificato dopo la sua registrazione.
- Scegli Registrati.

Sulla pagina Certificates (Certificati), nella finestra di dialogo Certificates (Certificati) verranno ora visualizzati i certificati registrati.

Registra un certificato client firmato da una CA (console) non registrata

> **Note**
>
> Prima di eseguire questa procedura, assicurati di disporre del file .pem del certificato client.

Per registrare un certificato esistente AWS IoT utilizzando la console

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione, selezionare Security (Sicurezza), scegliere Certificates (Certificati), quindi Create (Crea).
3. In Create a certificate (Crea un certificato), individua la voce Use my certificate (Usa certificato personale) e scegli Get started (Inizia).
4. In Select a CA (Seleziona una CA), scegli Next (Avanti).
5. In Register existing device certificates (Registra certificati dispositivo esistenti), seleziona Select certificates (Scegli certificati) e scegli fino a 10 file di certificato da registrare.
6. Dopo aver chiuso la finestra di dialogo del file, selezionare se si desidera attivare o revocare i certificati client al momento della registrazione.

   Se non attivi un certificato al momento della registrazione, in Attivare un certificato client (console) viene descritto come attivarlo in un secondo momento.

   Se un certificato viene revocato al momento della registrazione, non può essere attivato in un secondo momento.

Dopo aver scelto i file di certificato da registrare e aver selezionato le operazioni da eseguire dopo la registrazione, seleziona Register certificates (Registra certificati).

I certificati client registrati correttamente vengono visualizzati nell'elenco dei certificati.

Registrare un certificato client firmato da una CA registrata (CLI)

> **Note**
>
> Prima di eseguire questa procedura, assicurati di disporre del file .pem dell'autorità di certificazione (CA) e del file .pem del certificato client. Il certificato client deve essere firmato da un'autorità di certificazione (CA) presso cui ti sei registrato AWS IoT.

Utilizza il comando register-certificate per registrare ma non attivare un certificato client.

```
aws iot register-certificate \
    --certificate-pem file://device_cert_filename.pem \
    --ca-certificate-pem file://ca_cert_filename.pem
```

Il certificato client è registrato con AWS IoT, ma non è ancora attivo. Consulta Attivare un certificato client (CLI) per informazioni su come attivarlo in un secondo momento.

È inoltre possibile attivare il certificato client quando lo si registra utilizzando questo comando.

```
aws iot register-certificate \
    --set-as-active \
    --certificate-pem file://device_cert_filename.pem \
    --ca-certificate-pem file://ca_cert_filename.pem
```

Per ulteriori informazioni sull'attivazione del certificato in modo che possa essere utilizzato per la connessione AWS IoT, vedere Attivare o disattivare un certificato client

Registra un certificato client firmato da una CA (CLI) non registrata

> **Note**
>
> Prima di eseguire questa procedura, assicurati di disporre del file .pem del certificato.

Utilizzare il comando register-certificate-without-ca per registrare, ma non attivare, un certificato client.

```
aws iot register-certificate-without-ca \
    --certificate-pem file://device_cert_filename.pem
```

Il certificato client è registrato con AWS IoT, ma non è ancora attivo. Consulta Attivare un certificato client (CLI) per informazioni su come attivarlo in un secondo momento.

È inoltre possibile attivare il certificato client quando lo si registra utilizzando questo comando.

```
aws iot register-certificate-without-ca \
    --status ACTIVE \
    --certificate-pem file://device_cert_filename.pem
```

Per ulteriori informazioni sull'attivazione del certificato in modo che possa essere utilizzato per la connessione AWS IoT, consultaAttivare o disattivare un certificato client.

Registrare un certificato client quando il client si connette alla AWS IoT just-in-time registrazione (JITR)

È possibile configurare un certificato CA per abilitare la registrazione AWS IoT automatica dei certificati client con cui ha firmato la registrazione la prima volta che il client si connette. AWS IoT

Per registrare i certificati client quando un client si connette AWS IoT per la prima volta, è necessario abilitare il certificato CA per la registrazione automatica e configurare la prima connessione da parte del client per fornire i certificati richiesti.

Configurare un certificato CA per supportare la registrazione automatica (console)

Per configurare un certificato CA per supportare la registrazione automatica dei certificati client tramite la AWS IoT console

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione a sinistra, scegli Sicuro, scegli CAs.
3. Nell'elenco delle autorità di certificazione individuare quella per cui si desidera abilitare la registrazione automatica e aprire il menu delle opzioni utilizzando l'icona con i puntini di sospensione.
4. Scegliere Enable auto-registration (Abilita registrazione automatica)dal menu delle opzioni.

> **Note**
>
> Lo stato della registrazione automatica non viene visualizzato nell'elenco delle autorità di certificazione. Per visualizzare lo stato di registrazione automatica di un'autorità di certificazione, è necessario aprire la pagina Details (Dettagli) dell'autorità di certificazione.

Configurare un certificato CA per supportare la registrazione automatica (CLI)

Se hai già registrato il certificato CA con AWS IoT, usa il update-ca-certificatecomando per impostare `autoRegistrationStatus` il certificato CA su`ENABLE`.

```
aws iot update-ca-certificate \
--certificate-id caCertificateId \
--new-auto-registration-status ENABLE
```

Se si desidera abilitare `autoRegistrationStatus` quando si registra il certificato CA, utilizzare il comando register-ca-certificate.

```
aws iot register-ca-certificate \
--allow-auto-registration  \
--ca-certificate file://root_CA_cert_filename.pem \
--verification-cert file://verification_cert_filename.pem
```

Usare il comando describe-ca-certificate per visualizzare lo stato del certificato CA.

Configurare la prima connessione da parte di un client per la registrazione automatica

Quando un client tenta di connettersi AWS IoT per la prima volta, il certificato client firmato dal certificato CA deve essere presente sul client durante l'handshake Transport Layer Security (TLS).

Quando il client si connette AWS IoT, utilizza il certificato client creato in Crea certificati AWS IoT client o Crea certificati client personalizzati. AWS IoT riconosce il certificato CA come certificato CA registrato, registra il certificato client e ne imposta lo stato su. `PENDING_ACTIVATION` Questo significa che il certificato client è stato automaticamente registrato ed è in attesa dell'attivazione. Lo stato del certificato client deve essere `ACTIVE` prima che possa essere utilizzato per connettersi ad AWS IoT. Vedi Attivare o disattivare un certificato client per informazioni sull'attivazione di un certificato client.

> **Note**
>
> È possibile effettuare il provisioning dei dispositivi utilizzando la funzionalità di AWS IoT Core just-in-time registrazione (JITR) senza dover inviare l'intera catena di fiducia alla prima connessione dei dispositivi a. AWS IoT Core La presentazione del certificato emesso da una CA è facoltativa, ma il dispositivo è necessario per inviare l'estensione Server Name Indication (SNI) quando si collegano.

Quando registra AWS IoT automaticamente un certificato o quando un client presenta un certificato nello `PENDING_ACTIVATION` stato, AWS IoT pubblica un messaggio sul seguente argomento MQTT:

`$aws/events/certificates/registered/`*`caCertificateId`*

Dove *`caCertificateId`* è l'ID del certificato emesso da una CA che ha rilasciato il certificato client.

Il messaggio pubblicato in questo argomento ha la struttura seguente:

```
{
        "certificateId": "certificateId",
        "caCertificateId": "caCertificateId",
        "timestamp": timestamp,
        "certificateStatus": "PENDING_ACTIVATION",
        "awsAccountId": "awsAccountId",
        "certificateRegistrationTimestamp": "certificateRegistrationTimestamp"
}
```

Puoi creare una regola che resti in ascolto in questo argomento ed esegua alcune operazioni. Ti consigliamo di creare una regola Lambda che verifichi che il certificato client non sia incluso in un elenco di revoche di certificati (CRL), che attivi il certificato e crei e colleghi una policy a quest'ultimo. La policy determina le risorse cui il client può accedere. Se la politica che state creando richiede l'ID client dai dispositivi di connessione, potete utilizzare la funzione clientid () della regola per recuperare l'ID client. Un esempio di definizione di regola può essere simile al seguente:

```
SELECT *,
   clientid() as clientid
from $aws/events/certificates/registered/caCertificateId
```

In questo esempio, la regola sottoscrive l'argomento JITR `$aws/events/certificates/registered/`*`caCertificateID`* e utilizza la funzione clientid () per recuperare l'ID del client. La regola aggiunge quindi l'ID client al payload JITR. Per ulteriori informazioni sulla funzione clientid () della regola, vedete clientid ().

Per ulteriori informazioni su come creare una regola Lambda che ascolti l'`$aws/events/certificates/registered/`*`caCertificateID`*argomento ed esegua queste azioni, vedi just-in-time Registrazione dei certificati client su. AWS IoT

Se si verifica un errore o un'eccezione durante la registrazione automatica dei certificati client, AWS IoT invia eventi o messaggi ai registri in CloudWatch Logs. Per ulteriori informazioni sulla configurazione dei log per il tuo account, consulta la CloudWatch documentazione di Amazon.

Gestisci i certificati dei clienti

AWS IoT fornisce funzionalità per la gestione dei certificati client.

In questo argomento:

- Attivare o disattivare un certificato client
- Collegare un oggetto o una policy a un certificato client
- Revocare un certificato client
- Trasferire un certificato a un altro account

Attivare o disattivare un certificato client

AWS IoT verifica che un certificato client sia attivo quando autentica una connessione.

Puoi creare e registrare certificati client senza attivarli in modo che non possano essere utilizzati finché non lo desideri. È inoltre possibile disattivare i certificati client attivi per disabilitarli temporaneamente. Infine, puoi revocare i certificati client per impedirne qualsiasi utilizzo futuro.

Attivare un certificato client (console)

Per attivare un certificato client utilizzando la console AWS IoT

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione di sinistra, scegliere Secure (Sicurezza), quindi Certificates (Certificati).

3. Nell'elenco dei certificati individuare il certificato che si desidera attivare e aprire il menu delle opzioni utilizzando l'icona con i puntini di sospensione.
4. Scegliere Activate (Attiva)dal menu delle opzioni.

Il certificato viene visualizzato come Active (Attivo) nell'elenco dei certificati.

Disattivare un certificato client (console)

Per disattivare un certificato client utilizzando la console AWS IoT

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione di sinistra, scegliere Secure (Sicurezza), quindi Certificates (Certificati).
3. Nell'elenco dei certificati individuare il certificato che si desidera disattivare e aprire il menu delle opzioni utilizzando l'icona con i puntini di sospensione.
4. Scegliere Deactivate (Disattiva)dal menu delle opzioni.

Il certificato viene visualizzato come Inactive (Inattivo) nell'elenco dei certificati.

Attivare un certificato client (CLI)

AWS CLI Fornisce il update-certificatecomando per attivare un certificato.

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status ACTIVE
```

Se il comando ha esito positivo, lo stato del certificato è `ACTIVE`. Eseguire describe-certificate per visualizzare lo stato del certificato.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Disattivare un certificato client (CLI)

AWS CLI Fornisce il update-certificatecomando per disattivare un certificato.

```
aws iot update-certificate \
```

```
    --certificate-id certificateId \
    --new-status INACTIVE
```

Se il comando ha esito positivo, lo stato del certificato è `INACTIVE`. Eseguire describe-certificate per visualizzare lo stato del certificato.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Collegare un oggetto o una policy a un certificato client

Quando si crea e si registra un certificato separato da un AWS IoT oggetto, questo non avrà alcuna politica che autorizzi alcuna AWS IoT operazione, né sarà associato a nessun AWS IoT oggetto. In questa sezione viene descritto come aggiungere queste relazioni a un certificato registrato.

> **Important**
>
> Per completare queste procedure, è necessario aver già creato l'oggetto o la policy che si desidera collegare al certificato.

Il certificato autentica un dispositivo in AWS IoT modo che possa connettersi. Collegare il certificato a una risorsa dell'oggetto stabilisce la relazione tra il dispositivo (tramite il certificato) e la risorsa dell'oggetto. Per autorizzare il dispositivo a eseguire AWS IoT azioni, ad esempio consentirgli di connettersi e pubblicare messaggi, è necessario allegare una politica appropriata al certificato del dispositivo.

Collegare un oggetto a un certificato client (console)

Per completare questa procedura, è necessario il nome dell'oggetto.

Per collegare un oggetto a un certificato registrato

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione di sinistra, scegliere Secure (Sicurezza), quindi Certificates (Certificati).
3. Nell'elenco dei certificati individuare il certificato a cui si desidera collegare una policy, aprire il menu delle opzioni del certificato scegliendo l'icona con i puntini di sospensione e scegliere Collega oggetto.

4. Nella finestra popup individua il nome dell'oggetto che desideri collegare al certificato, scegli la relativa casella di controllo e scegli Attach (Collega).

L'oggetto dovrebbe ora essere visualizzato nell'elenco degli oggetti nella pagina dei dettagli del certificato.

Collegare una policy a un certificato client (console)

Per completare questa procedura, sarà necessario il nome dell'oggetto policy.

Per collegare un oggetto policy a un certificato registrato

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione di sinistra, scegliere Secure (Sicurezza), quindi Certificates (Certificati).
3. Nell'elenco dei certificati individuare il certificato a cui si desidera collegare una policy, aprire il menu delle opzioni del certificato scegliendo l'icona con i puntini di sospensione e scegliere Collega policy.
4. Nella finestra popup individua il nome della policy che desideri collegare al certificato, scegli la relativa casella di controllo e scegli Attach (Collega).

L'oggetto policy dovrebbe ora essere visualizzato nell'elenco delle policy nella pagina dei dettagli del certificato.

Collegare un oggetto a un certificato client (CLI)

AWS CLI Fornisce il attach-thing-principalcomando per allegare un oggetto a un certificato.

```
aws iot attach-thing-principal \
    --principal certificateArn \
    --thing-name thingName
```

Collegare una policy a un certificato client (CLI)

AWS CLI Fornisce il attach-policycomando per allegare un oggetto di policy a un certificato.

```
aws iot attach-policy \
    --target certificateArn \
    --policy-name policyName
```

Revocare un certificato client

Se rilevi un'attività sospetta in un certificato client registrato, puoi revocarlo in modo che non possa essere utilizzato nuovamente.

> **Note**
>
> Dopo che un certificato è stato revocato, il suo stato non può essere modificato. Ovvero, lo stato del certificato non può essere modificato in `Active` o in qualsiasi altro stato.

Revocare un certificato client (console)

Per revocare un certificato client utilizzando la console AWS IoT

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione di sinistra, scegliere Secure (Sicurezza), quindi Certificates (Certificati).
3. Nell'elenco dei certificati individuare il certificato che si desidera revocare e aprire il menu delle opzioni utilizzando l'icona con i puntini di sospensione.
4. Nel menu delle opzioni, scegliere Revoke (Revoca).

Se il certificato viene revocato correttamente, è visualizzato come Revoked (Revocato) nell'elenco dei certificati.

Revocare un certificato client (CLI)

AWS CLI Fornisce il update-certificatecomando per revocare un certificato.

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status REVOKED
```

Se il comando ha esito positivo, lo stato del certificato è `REVOKED`. Eseguire describe-certificate per visualizzare lo stato del certificato.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Trasferire un certificato a un altro account

I certificati X.509 che appartengono a uno Account AWS possono essere trasferiti a un altro. Account AWS

Per trasferire un certificato X.509 da uno all'altro Account AWS

1. the section called “Avviare un trasferimento di certificato”

   Il certificato deve essere disattivato e scollegato da tutti le policy e gli oggetti prima di avviare il trasferimento.
2. the section called “Accettare o rifiutare un trasferimento di certificato”

   L'account ricevente deve accettare o rifiutare esplicitamente il certificato trasferito. Dopo che l'account ricevente ha accettato il certificato, il certificato deve essere attivato prima dell'uso.
3. the section called “Annullamento di un trasferimento di certificato”

   L'account di origine può annullare un trasferimento, se il certificato non è stato accettato.

Avviare un trasferimento di certificato

È possibile iniziare a trasferire un certificato a un altro Account AWS utilizzando la AWS IoT console o il. AWS CLI

Avviare un trasferimento di certificato (console)

Per completare questa procedura, è necessario l'ID del certificato che vuoi trasferire.

Esegui questa procedura dall'account con il certificato da trasferire.

Per iniziare a trasferire un certificato a un altro Account AWS

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione di sinistra, scegliere Secure (Sicurezza), quindi Certificates (Certificati).

   Scegli il certificato con uno stato Active (Attivo) o Inactive (Inattivo) da trasferire e aprine la pagina dettagli.
3. Nella pagina Details (Dettagli) del certificato, nel menu Actions (Operazioni), se l'opzione Deactivate (Disattiva) è disponibile, selezionala per disattivare il certificato.

4. Sulla pagina Details (Dettagli) del certificato, seleziona Policies (Policy) nel menu di sinistra.
5. Se sulla pagina Policies (Policy) del certificato sono presenti delle policy allegate al certificato, scollegali aprendo il menu delle opzioni delle policy e scegliendo Detach (Distacca).

   Prima di continuare, verifica che al certificato non siano collegate policy.
6. Sul menu a sinistra della pagina Policies (Policy) del certificato seleziona Things (Oggetti).
7. Se sulla pagina Things (Oggetti) del certificato ci sono oggetti allegati al certificato, scollegali aprendo il menu delle opzioni dell'oggetto e scegliendo Detach (Distacca).

   Prima di continuare, il certificato non deve essere collegato a nessun oggetto.
8. Sulla pagina Things (Oggetti) del certificato, nel menu a sinistra seleziona Details (Dettagli).
9. Nella pagina Details (Dettagli) del certificato, nel menu Actions (Operazioni), scegli Start transfer (Avvia trasferimento) per aprire la relativa finestra di dialogo.
10. Nella finestra di dialogo Avvia trasferimento, inserisci il Account AWS numero dell'account per ricevere il certificato e un breve messaggio opzionale.
11. Scegli Start transfer (Avvio del trasferimento) per trasferire il certificato.

La console dovrebbe visualizzare un messaggio che indica l'esito o il fallimento del trasferimento. Se il trasferimento è stato avviato, lo stato del certificato viene aggiornato a Transferred (Trasferimento).

Avviare un trasferimento di certificati (CLI)

Per completare questa procedura, avrai bisogno *certificateArn* dell'*certificateId*e del certificato che desideri trasferire.

Esegui questa procedura dall'account con il certificato da trasferire.

Per iniziare a trasferire un certificato su un altro AWS account

1. Utilizza il comando update-certificate per disattivare il certificato.

   ```
   aws iot update-certificate --certificate-id certificateId --new-status INACTIVE
   ```
2. Scollega tutte le policy.

   1. Utilizza il comando list-attached-policies per elencare le policy collegate al certificato.

      ```
      aws iot list-attached-policies --target certificateArn
      ```

2. Per ogni policy allegata, utilizza il comando detach-policy per scollegare la policy.

```
aws iot detach-policy --target certificateArn --policy-name policy-name
```

3. Scollega tutti gli oggetti.

   1. Utilizza il comando list-principal-things per fare una lista delle cose collegate al certificato.

```
aws iot list-principal-things --principal certificateArn
```

   2. Utilizza il comando detach-thing-principal per scollegare ogni oggetto allegato.

```
aws iot detach-thing-principal --principal certificateArn --thing-name thing-name
```

4. Utilizza il comando transfer-certificate per avviare il trasferimento del certificato.

```
aws iot transfer-certificate --certificate-id certificateId --target-aws-
account account-id
```

Accettare o rifiutare un trasferimento di certificato

È possibile accettare o rifiutare un certificato trasferito Account AWS da un altro utente Account AWS utilizzando la AWS IoT console o il AWS CLI.

Accettare o rifiutare un trasferimento di certificato (console)

Per completare questa procedura, è necessario l'ID del certificato trasferito nel tuo account.

Eseguire questa procedura dall'account che riceve il certificato trasferito.

Per accettare o rifiutare un certificato che è stato trasferito su Account AWS

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione di sinistra, scegliere Secure (Sicurezza), quindi Certificates (Certificati).

   Scegli il certificato con lo stato Pending transfer (Trasferimento in sospeso) per accettare o rifiutare e aprirne la pagina dettagli.
3. Sulla pagina Details (Dettagli) del certificato, nel menu Actions (Operazioni),

- Per accettare il certificato, scegli Accept transfer (Accetta il trasferimento).
- Per non accettare il certificato, scegli Reject transfer (Rifiuta del trasferimento).

Accettare o rifiutare un trasferimento di certificato (CLI)

Per completare questa procedura, è necessario il trasferimento *certificateId* del certificato che si desidera accettare o rifiutare.

Eseguire questa procedura dall'account che riceve il certificato trasferito.

Per accettare o rifiutare un certificato che è stato trasferito al tuo Account AWS

1. Per accettare il certificato, utilizza il comando accept-certificate-transfer.

   ```
   aws iot accept-certificate-transfer --certificate-id certificateId
   ```

2. Per rifiutare il certificato, utilizza il comando reject-certificate-transfer.

   ```
   aws iot reject-certificate-transfer --certificate-id certificateId
   ```

Annullamento di un trasferimento di certificato

È possibile annullare un trasferimento di certificato prima che sia stato accettato utilizzando la console AWS IoT o l' AWS CLI.

Annullamento di un trasferimento di certificato (console)

Per completare questa procedura, è necessario l'ID del certificato che vuoi cancellare.

Esegui questa procedura dall'account che ha avviato il trasferimento del certificato.

Annullamento di un trasferimento di certificato

1. Accedi alla console di AWS gestione e apri la AWS IoT console.
2. Nel riquadro di navigazione di sinistra, scegliere Secure (Sicurezza), quindi Certificates (Certificati).

   Scegli il certificato con lo stato Transferred (Trasferimento) di cui si desidera annullare il trasferimento e apri il menu delle opzioni.

3. Nel menu delle opzioni del certificato, scegli l'opzione Revoke transfer (Revoca del trasferimento) per annullare il trasferimento del certificato.

> **Important**
>
> Fai attenzione a non confondere Revoke transfer (Revoca del trasferimento) con l'opzione Revoke (Revoca).
> La Revoke transfer (Revoca del trasferimento) annulla il trasferimento del certificato, mentre l'opzione Revoke (Revoca) rende il certificato irreversibilmente inutilizzabile da AWS IoT.

Annullamento di un trasferimento di certificato (CLI)

Per completare questa procedura, avrai bisogno *`certificateId`* del trasferimento del certificato che desideri annullare.

Esegui questa procedura dall'account che ha avviato il trasferimento del certificato.

Utilizza il comando cancel-certificate-transfer per annullare il trasferimento del certificato.

```
aws iot cancel-certificate-transfer --certificate-id certificateId
```

Convalida personalizzata del certificato del cliente

AWS IoT Core supporta la convalida personalizzata dei certificati client per i certificati client X.509, che migliora la gestione dell'autenticazione dei client. Questo metodo di convalida dei certificati è noto anche come controllo dei certificati di preautenticazione, in cui si valutano i certificati client in base a criteri personalizzati (definiti in una funzione Lambda) e si revocano i certificati client o il certificato dell'autorità di certificazione (CA) di firma dei certificati per impedire ai client di connettersi. AWS IoT Core Ad esempio, è possibile creare controlli di revoca dei certificati personalizzati che convalidano lo stato dei certificati rispetto alle autorità di convalida che supportano gli endpoint OCSP (Online Certificate Status Protocol) o Certificate Revocation Lists (CRL) e impediscono le connessioni per i client con certificati revocati. I criteri utilizzati per valutare i certificati client sono definiti in una funzione Lambda (nota anche come Lambda di preautenticazione). È necessario utilizzare gli endpoint impostati nelle configurazioni di dominio e il tipo di autenticazione deve essere un certificato X.509. Inoltre, i client devono fornire l'estensione SNI (Server Name Indication) durante la connessione a. AWS IoT Core

> **Note**
>
> Questa funzionalità non è supportata nelle AWS GovCloud (US) regioni.

Il processo di esecuzione della convalida personalizzata dei certificati client prevede i seguenti passaggi.

- Fase 1: Registrare i certificati client X.509 con AWS IoT Core
- Passaggio 2: creazione di una funzione Lambda
- Fase 3: AWS IoT Autorizza a richiamare la funzione Lambda
- Fase 4: Impostare la configurazione di autenticazione per un dominio

Fase 1: Registrare i certificati client X.509 con AWS IoT Core

Se non l'hai già fatto, registra e attiva i certificati client X.509 con. AWS IoT Core Altrimenti, passare alla fase successiva.

Per registrare e attivare i certificati client con AWS IoT Core, procedi nel seguente modo:

1. Se crei certificati client direttamente con AWS IoT. Questi certificati client verranno registrati automaticamente con AWS IoT Core.
2. Se crei i tuoi certificati client, segui queste istruzioni per registrarli AWS IoT Core.
3. Per attivare i certificati client, segui queste istruzioni.

Passaggio 2: creazione di una funzione Lambda

È necessario creare una funzione Lambda che esegua la verifica del certificato e venga chiamata per ogni tentativo di connessione del client per l'endpoint configurato. Quando crei questa funzione Lambda, segui le indicazioni generali di Crea la tua prima funzione Lambda. Inoltre, assicurati che la funzione Lambda aderisca ai formati di richiesta e risposta previsti come segue:

Esempio di evento della funzione Lambda

```
{
 "connectionMetadata": {
  "id": "string"
 },
 "principalId": "string",
```

```
 "serverName": "string",
 "clientCertificateChain": [
  "string",
  "string"
 ]
}
```

`connectionMetadata`

Metadati o informazioni aggiuntive relative alla connessione del client a. AWS IoT Core

`principalId`

L'identificatore principale associato al client nella connessione TLS.

`serverName`

La stringa del nome host SNI (Server Name Indication). AWS IoT Core richiede che i dispositivi inviino l'estensione SNI al protocollo Transport Layer Security (TLS) e forniscano l'indirizzo completo dell'endpoint sul campo. `host_name`

`clientCertificateChain`

L'array di stringhe che rappresenta la catena di certificati X.509 del client.

Esempio di risposta alla funzione Lambda

```
{
 "isAuthenticated": "boolean"
}
```

`isAuthenticated`

Un valore booleano che indica se la richiesta è autenticata.

> **Note**
>
> Nella risposta Lambda, `isAuthenticated` deve essere necessario procedere `true` all'ulteriore autenticazione e autorizzazione. In caso contrario, il certificato client IoT può essere disabilitato e l'autenticazione personalizzata con certificati client X.509 può essere bloccata per ulteriori autenticazioni e autorizzazioni.

Fase 3: AWS IoT Autorizza a richiamare la funzione Lambda

Dopo aver creato la funzione Lambda, è necessario concedere l'autorizzazione AWS IoT per richiamarla utilizzando il comando CLI add-permission. Nota che questa funzione Lambda verrà richiamata per ogni tentativo di connessione all'endpoint configurato. Per ulteriori informazioni, consulta Autorizzazione AWS IoT a richiamare la funzione Lambda.

Fase 4: Impostare la configurazione di autenticazione per un dominio

La sezione seguente descrive come impostare la configurazione di autenticazione per un dominio personalizzato utilizzando AWS CLI.

Imposta la configurazione del certificato client per un dominio (CLI)

Se non disponi di una configurazione di dominio, usa il comando create-domain-configurationCLI per crearne una. Se disponi già di una configurazione di dominio, utilizza il comando update-domain-configurationCLI per aggiornare la configurazione del certificato client per un dominio. È necessario aggiungere l'ARN della funzione Lambda creata nel passaggio precedente.

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type AWS_X509|CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT|HTTPS \
    --client-certificate-config 'clientCertificateCallbackArn":"arn:aws:lambda:us-
east-2:123456789012:function:my-function:1"}'
```

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type AWS_X509|CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT|HTTPS \
    --client-certificate-config '{"clientCertificateCallbackArn":"arn:aws:lambda:us-
east-2:123456789012:function:my-function:1"}'
```

`domain-configuration-name`

Il nome della configurazione di dominio.

`authentication-type`

Il tipo di autenticazione della configurazione del dominio. Per ulteriori informazioni, vedi Scelta del tipo di autenticazione.

`application-protocol`

Il protocollo applicativo con cui i dispositivi comunicano AWS IoT Core. Per ulteriori informazioni, vedere Scelta di un protocollo applicativo.

`client-certificate-config`

Un oggetto che specifica la configurazione di autenticazione del client per un dominio.

`clientCertificateCallbackArn`

L'Amazon Resource Name (ARN) della funzione Lambda che AWS IoT richiama il livello TLS quando viene stabilita una nuova connessione. Per personalizzare l'autenticazione client per eseguire la convalida personalizzata del certificato client, devi aggiungere l'ARN della funzione Lambda che hai creato nel passaggio precedente.

Per ulteriori informazioni, consulta CreateDomainConfiguratione utilizza l'AWS IoT API UpdateDomainConfigurationReference. Per ulteriori informazioni sulle configurazioni dei domini, consulta Configurazioni dei domini.

Utenti, gruppi e ruoli IAM

Gli utenti, i gruppi e i ruoli IAM; sono i meccanismi standard per la gestione di identità e autenticazione nei servizi AWS. Puoi usarli per connetterti alle interfacce AWS IoT HTTP utilizzando l' AWS SDK e. AWS CLI

I ruoli IAM consentono inoltre AWS IoT di accedere ad altre AWS risorse del tuo account per tuo conto. Ad esempio, se desideri che un dispositivo pubblichi il suo stato su una tabella DynamoDB, i ruoli IAM AWS IoT consentono di interagire con Amazon DynamoDB. Per ulteriori informazioni, consulta IAM Roles (Ruoli IAM).

Per le connessioni ai broker di messaggi tramite HTTP, AWS IoT autentica utenti, gruppi e ruoli utilizzando il processo di firma Signature Version 4. Per informazioni, consulta Firmare le richieste AWS API.

Quando si utilizza AWS Signature Version 4 con AWS IoT, i client devono supportare quanto segue nella loro implementazione TLS:

- TLS 1.2
- Convalida della firma dei certificati SHA-256 RSA.

- Uno dei pacchetti di crittografia indicati nella sezione relativa ai pacchetti di crittografia TLS supportati.

Per informazioni, consulta Gestione delle identità e degli accessi per AWS IoT.

Identità Amazon Cognito

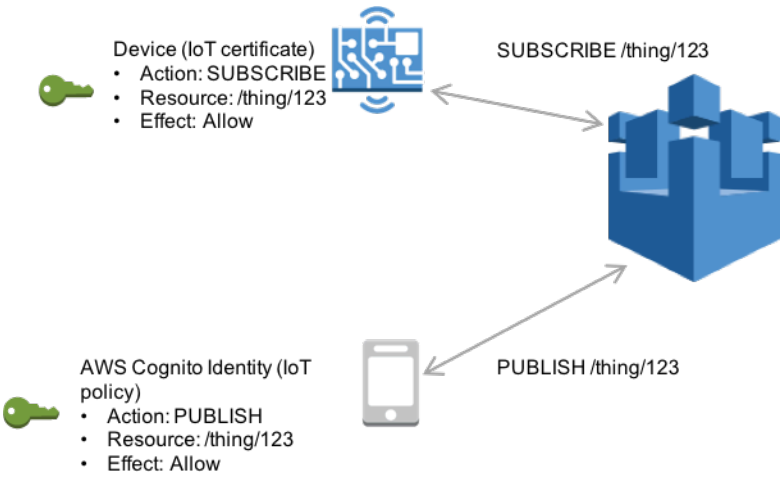
Amazon Cognito Identity ti consente di creare AWS credenziali temporanee con privilegi limitati da utilizzare in applicazioni mobili e Web. Quando usi Amazon Cognito Identity, crea pool di identità che creano identità uniche per i tuoi utenti e autenticali con provider di identità come Login with Amazon, Facebook e Google. È inoltre possibile utilizzare le identità Amazon Cognito con identità autenticate dallo sviluppatore. Per ulteriori informazioni, consulta Amazon Cognito Identity.

Per utilizzare Amazon Cognito Identity, definisci un pool di identità Amazon Cognito associato a un ruolo IAM. Il ruolo IAM è associato a una policy IAM che concede alle identità del tuo pool di identità l'autorizzazione ad accedere a AWS risorse come i servizi di chiamata. AWS

Amazon Cognito Identity crea identità non autenticate e autenticate. Le identità non autenticate vengono utilizzate per gli utenti guest di un'applicazione mobile o Web che desiderano utilizzare l'app senza accedere. Agli utenti non autenticati vengono concesse solo le autorizzazioni specificate nelle policy IAM associate al pool di identità.

Quando utilizzi identità autenticate, oltre alla policy IAM associata al pool di identità, devi allegare una AWS IoT policy a un'identità Amazon Cognito. Per allegare una AWS IoT policy, usa l'AttachPolicyAPI e concedi le autorizzazioni a un singolo utente della tua applicazione. AWS IoT Puoi utilizzare la AWS IoT policy per assegnare autorizzazioni dettagliate a clienti specifici e ai relativi dispositivi.

Gli utenti autenticati e non autenticati sono diversi tipi di identità. Se non alleghi una AWS IoT policy all'identità di Amazon Cognito, un utente autenticato non ottiene l'autorizzazione AWS IoT e non ha accesso a AWS IoT risorse e azioni. Per ulteriori informazioni sulla creazione di policy per le identità Amazon Cognito, consulta Esempi di policy di pubblicazione/sottoscrizione e Autorizzazione con identità Amazon Cognito.



Autenticazione e autorizzazione personalizzata

AWS IoT Core consente di definire autorizzatori personalizzati in modo da poter gestire l'autenticazione e l'autorizzazione dei client. Ciò è utile quando è necessario utilizzare meccanismi di autenticazione diversi da quelli supportati AWS IoT Core nativamente. (Per ulteriori informazioni sui meccanismi supportati in modo nativo, consulta the section called “Autenticazione client”).

Ad esempio, se state migrando dispositivi esistenti sul campo verso AWS IoT Core e questi dispositivi utilizzano un token bearer personalizzato o un nome utente e una password MQTT per l'autenticazione, potete migrarli AWS IoT Core senza dover fornire loro nuove identità. È possibile utilizzare l'autenticazione personalizzata con tutti i protocolli di comunicazione supportati. AWS IoT Core Per ulteriori informazioni sui protocolli supportati da AWS IoT Core , consulta the section called “Protocolli di dispositivo di comunicazione”.

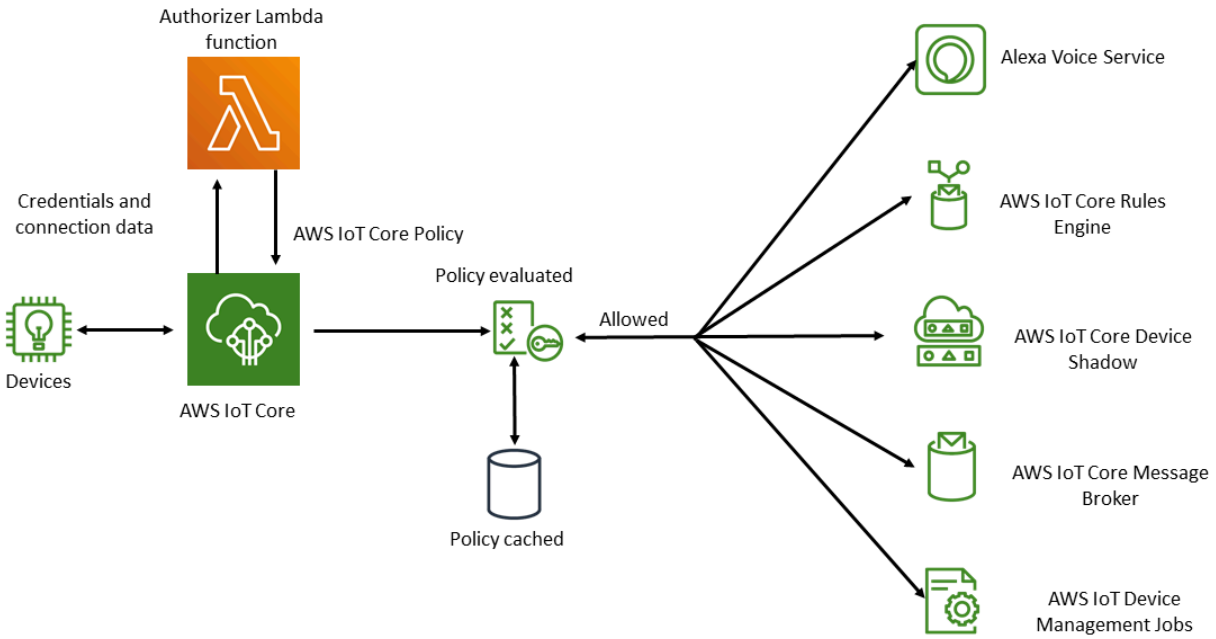
Argomenti

- Informazioni sul flusso di lavoro di autenticazione personalizzato
- Creazione e gestione di autorizzazioni personalizzate (CLI)
- Autenticazione personalizzata con certificati client X.509
- Connessione a AWS IoT Core tramite autenticazione personalizzata
- Risoluzione dei problemi relativi agli autorizzatori

Informazioni sul flusso di lavoro di autenticazione personalizzato

L'autenticazione personalizzata consente di definire come autenticare e autorizzare i client utilizzando le risorse dell'autorizzatore. Ogni autorizzatore contiene un riferimento a una funzione Lambda

gestita dal cliente, una chiave pubblica opzionale per la convalida delle credenziali del dispositivo e informazioni di configurazione aggiuntive. Il diagramma seguente illustra il flusso di lavoro di autorizzazione per l'autenticazione personalizzata in. AWS IoT Core



AWS IoT Core flusso di lavoro di autenticazione e autorizzazione personalizzato

Nell'elenco seguente viene illustrato ogni passaggio del flusso di lavoro di autenticazione e autorizzazione personalizzata.

1. Un dispositivo si connette all'endpoint di AWS IoT Core dati di un cliente utilizzando uno dei dispositivi supportatithe section called “Protocolli di dispositivo di comunicazione”. Il dispositivo passa le credenziali nei campi di intestazione o nei parametri di query della richiesta (per i WebSockets protocolli HTTP Publish o MQTT over) o nel campo nome utente e password del messaggio MQTT CONNECT (per i protocolli MQTT e MQTT over). WebSockets
2. AWS IoT Core verifica una delle due condizioni:
   - La richiesta in arrivo specifica un autorizzatore.
   - Per l'endpoint di AWS IoT Core dati che riceve la richiesta è configurato un autorizzatore predefinito.

Se AWS IoT Core trova un autorizzatore in uno di questi modi, AWS IoT Core attiva la funzione Lambda associata all'autorizzatore.

3. (Facoltativo) Se hai abilitato la firma con token, AWS IoT Core convalida la firma della richiesta utilizzando la chiave pubblica memorizzata nell'autorizzatore prima di attivare la funzione Lambda. Se la convalida non riesce, AWS IoT Core interrompe la richiesta senza invocare la funzione Lambda.
4. La funzione Lambda riceve le credenziali e i metadati di connessione nella richiesta e prende una decisione di autenticazione.
5. La funzione Lambda restituisce i risultati della decisione di autenticazione e un documento di AWS IoT Core policy che specifica quali azioni sono consentite nella connessione. La funzione Lambda restituisce anche informazioni che specificano la frequenza di AWS IoT Core riconvalida delle credenziali nella richiesta richiamando la funzione Lambda.
6. AWS IoT Core valuta l'attività sulla connessione rispetto alla policy ricevuta dalla funzione Lambda.
7. Dopo aver stabilito la connessione e richiamato inizialmente l'autorizzatore personalizzato Lambda, la chiamata successiva può essere ritardata fino a 5 minuti su connessioni inattive senza alcuna operazione MQTT. Dopodiché, le chiamate successive seguiranno l'intervallo di aggiornamento nell'autorizzatore personalizzato Lambda. Questo approccio può prevenire invocazioni eccessive che potrebbero superare il limite di concorrenza Lambda del tuo. Account AWS

Considerazioni sul dimensionamento

Poiché una funzione Lambda gestisce l'autenticazione e l'autorizzazione per l'autorizzatore, la funzione è soggetta ai prezzi e ai limiti del servizio Lambda, ad esempio la frequenza di esecuzione simultanea. Per ulteriori informazioni sui prezzi Lambda, consulta Prezzi Lambda. È possibile gestire il carico sulla funzione Lambda regolando `refreshAfterInSeconds` e `disconnectAfterInSeconds` nella risposta della funzione Lambda. Per ulteriori informazioni sui contenuti della risposta della funzione Lambda, consulta the section called “Scrittura della funzione Lambda”.

> **Note**
>
> Se si lascia abilitata la firma, è possibile impedire l'attivazione eccessiva di Lambda da parte di client non riconosciuti. Consideralo prima di disabilitare l'accesso all'autorizzatore.

> **Note**
>
> Il limite di timeout della funzione Lambda per l'autorizzatore personalizzato è di 5 secondi.

Creazione e gestione di autorizzazioni personalizzate (CLI)

AWS IoT Core implementa schemi di autenticazione e autorizzazione personalizzati utilizzando autorizzatori personalizzati. Un autorizzatore personalizzato è una AWS IoT Core risorsa che offre la flessibilità necessaria per definire e implementare le regole e le politiche in base ai requisiti specifici. Per creare un autorizzatore personalizzato con step-by-step istruzioni, vedi Tutorial: Creazione di un autorizzatore personalizzato per. AWS IoT Core

Ogni autorizzatore è costituito dai seguenti componenti:

- Nome: stringa univoca definita dall'utente che identifica l'autorizzatore.
- ARN della funzione Lambda: L'Amazon Resource Name (ARN) della funzione Lambda che implementa la logica di autorizzazione e di autenticazione.
- Nome della chiave dei token: Il nome della chiave utilizzato per estrarre il token dalle intestazioni HTTP, dai parametri di query o dal nome utente MQTT CONNECT per eseguire la convalida della firma. Questo valore è obbligatorio se la firma è abilitata nell'autorizzazione.
- Contrassegno di firma disabilitato (facoltativo): valore booleano che specifica se disattivare il requisito di firma sulle credenziali. Ciò è utile per gli scenari in cui la firma delle credenziali non ha senso, ad esempio gli schemi di autenticazione che utilizzano il nome utente e la password MQTT. Il valore predefinito è `false`, pertanto la firma è abilitata per impostazione predefinita.
- Chiave pubblica per la firma di token: chiave pubblica utilizzata da AWS IoT Core per convalidare la firma del token. La sua lunghezza minima è di 2.048 bit. Questo valore è obbligatorio se la firma è abilitata nell'autorizzatore.

Lambda addebita per il numero di volte in cui la tua funzione Lambda viene eseguita e per il tempo necessario per l'esecuzione del codice nella funzione. Per maggiori informazioni sui prezzi di Lambda consulta Prezzi Lambda. Per ulteriori informazioni sulla creazione di funzioni Lambda, consulta Guida per gli sviluppatori di Lambda.

> **Note**
>
> Se si lascia abilitata la firma, è possibile impedire l'attivazione eccessiva di Lambda da parte di client non riconosciuti. Consideralo prima di disabilitare l'accesso nel tuo autorizzatore.

> **Note**
>
> Il limite di timeout della funzione Lambda per l'autorizzatore personalizzato è di 5 secondi.

In questo capitolo:

- Scrittura della funzione Lambda
- Creazione di un autorizzatore
- Autorizzazione AWS IoT a richiamare la funzione Lambda
- Verificare le autorizzazioni
- Gestione degli autorizzatori personalizzati

Scrittura della funzione Lambda

Quando AWS IoT Core richiama l'autorizzatore, attiva la Lambda associata all'autorizzatore con un evento che contiene il seguente oggetto JSON. L'oggetto JSON di esempio contiene tutti i campi possibili. I campi non rilevanti per la richiesta di connessione non sono inclusi.

```
{
    "token" :"aToken",
    "signatureVerified": Boolean, // Indicates whether the device gateway has validated
 the signature.
    "protocols": ["tls", "http", "mqtt"], // Indicates which protocols to expect for
 the request.
    "protocolData": {
        "tls" : {
            "serverName": "serverName" // The server name indication (SNI) host_name
 string.
        },
        "http": {
            "headers": {
                "#{name}": "#{value}"
```

```
            },
            "queryString": "?#{name}=#{value}"
        },
        "mqtt": {
            "username": "myUserName",
            "password": "myPassword", // A base64-encoded string.
            "clientId": "myClientId" // Included in the event only when the device
 sends the value.
        }
    },
    "connectionMetadata": {
        "id": UUID // The connection ID. You can use this for logging.
    },
}
```

La funzione Lambda deve utilizzare queste informazioni per autenticare la connessione in ingresso e decidere quali azioni sono consentite nella connessione. La funzione deve inviare una risposta che contiene i seguenti valori.

- `isAuthenticated`: valore booleano che indica se la richiesta è stata autenticata.
- `principalId`: Una stringa alfanumerica che funge da identificatore per il token inviato dalla richiesta di autorizzazione personalizzata. Il valore deve essere una stringa alfanumerica con almeno uno e non più di 128 caratteri e corrisponde a questo pattern di espressione regolare (regex):`([a-zA-Z0-9]){1,128}`. I caratteri speciali che non sono alfanumerici non possono essere utilizzati con l'in. `principalId` AWS IoT Core Consultate la documentazione relativa agli altri AWS servizi se sono consentiti caratteri speciali non alfanumerici per. `principalId`
- `policyDocuments`: Un elenco di documenti relativi alle policy in formato JSON Per ulteriori informazioni sulla creazione di AWS IoT Core policy, vedere. AWS IoT Core the section called “AWS IoT Core politiche” Il numero massimo di documenti di policy è di 10. Ogni documento di policy può contenere un massimo di 2.048 caratteri.
- `disconnectAfterInSeconds`: numero intero che specifica la durata massima (in secondi) della connessione al gateway di AWS IoT Core . Il valore minimo è 300 secondi e il valore massimo è 86400 secondi. Il valore predefinito è 86.400.

> **Note**
>
> Il valore di `disconnectAfterInSeconds` (restituito dalla funzione Lambda) viene impostato quando viene stabilita la connessione. Questo valore non può essere modificato durante le successive chiamate Lambda di aggiornamento delle policy.

- `refreshAfterInSeconds`: numero intero che specifica l'intervallo tra gli aggiornamenti delle policy. Passato questo intervallo, AWS IoT Core richiama la funzione Lambda per consentire gli aggiornamenti delle policy. Il valore minimo è 300 secondi e il valore massimo è 86400 secondi.

Il seguente oggetto JSON contiene un esempio di risposta che la funzione Lambda può inviare.

```
{ "isAuthenticated":true, //A Boolean that determines whether
client can connect. "principalId": "xxxxxxxx",  //A string that
identifies the connection in logs. "disconnectAfterInSeconds": 86400,
  "refreshAfterInSeconds": 300,   "policyDocuments": [       {
 "Version": "2012-10-17",         "Statement": [           {
    "Action": "iot:Publish",              "Effect": "Allow",
      "Resource": "arn:aws:iot:us-east-1:<your_aws_account_id>:topic/
customauthtesting"            }         ]       }     ] }
```

Il `policyDocument` valore deve contenere un documento di policy valido. AWS IoT Core Per ulteriori informazioni sulle AWS IoT Core politiche, vederethe section called “AWS IoT Core politiche”. In MQTT su TLS e MQTT su WebSockets connessioni, AWS IoT Core memorizza nella cache questa politica per l'intervallo specificato nel valore del campo. `refreshAfterInSeconds` Nel caso di connessioni HTTP, la funzione Lambda viene richiamata per ogni richiesta di autorizzazione a meno che il dispositivo non utilizzi connessioni persistenti HTTP (chiamate anche HTTP keep-alive o riutilizzo della connessione HTTP), è possibile scegliere di abilitare la memorizzazione nella cache durante la configurazione dell'autorizzatore. Durante questo intervallo, AWS IoT Core autorizza le azioni in una connessione stabilita contro questa policy memorizzata nella cache senza attivare nuovamente la funzione Lambda. Se si verificano errori durante l'autenticazione personalizzata, interrompe la connessione. AWS IoT Core AWS IoT Core interrompe inoltre la connessione se è rimasta aperta per un periodo superiore al valore specificato nel parametro. `disconnectAfterInSeconds`

Di seguito JavaScript è riportato un esempio di funzione Lambda di Node.js che cerca una password nel messaggio MQTT Connect con un valore `test` di e restituisce un criterio che concede il

permesso di connettersi AWS IoT Core a un client `myClientName` denominato e pubblicare su un argomento che contiene lo stesso nome client. Se non trova la password prevista, restituisce una policy che nega queste due operazioni.

```
// A simple Lambda function for an authorizer. It demonstrates
// how to parse an MQTT password and generate a response.

exports.handler = function(event, context, callback) {
    var uname = event.protocolData.mqtt.username;
    var pwd = event.protocolData.mqtt.password;
    var buff = new Buffer(pwd, 'base64');
    var passwd = buff.toString('ascii');
    switch (passwd) {
        case 'test':
            callback(null, generateAuthResponse(passwd, 'Allow'));
            break;
        default:
            callback(null, generateAuthResponse(passwd, 'Deny'));
    }
};

// Helper function to generate the authorization response.
var generateAuthResponse = function(token, effect) {
    var authResponse = {};
    authResponse.isAuthenticated = true;
    authResponse.principalId = 'TEST123';

    var policyDocument = {};
    policyDocument.Version = '2012-10-17';
    policyDocument.Statement = [];
    var publishStatement = {};
    var connectStatement = {};
    connectStatement.Action = ["iot:Connect"];
    connectStatement.Effect = effect;
    connectStatement.Resource = ["arn:aws:iot:us-east-1:123456789012:client/
myClientName"];
    publishStatement.Action = ["iot:Publish"];
    publishStatement.Effect = effect;
    publishStatement.Resource = ["arn:aws:iot:us-east-1:123456789012:topic/telemetry/
myClientName"];
    policyDocument.Statement[0] = connectStatement;
    policyDocument.Statement[1] = publishStatement;
    authResponse.policyDocuments = [policyDocument];
```

```
    authResponse.disconnectAfterInSeconds = 3600;
    authResponse.refreshAfterInSeconds = 300;

    return authResponse;
}
```

La suddetta funzione Lambda restituisce il seguente JSON quando riceve la password prevista di `test` nel messaggio MQTT Connect. I valori delle proprietà `password` e `principalId` saranno i valori del messaggio MQTT Connect.

```
{
  "password": "password",
  "isAuthenticated": true,
  "principalId": "principalId",
  "policyDocuments": [
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "iot:Connect",
          "Effect": "Allow",
          "Resource": "*"
        },
        {
          "Action": "iot:Publish",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topic/telemetry/${iot:ClientId}"
        },
        {
          "Action": "iot:Subscribe",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topicfilter/telemetry/
${iot:ClientId}"
        },
        {
          "Action": "iot:Receive",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topic/telemetry/${iot:ClientId}"
        }
      ]
    }
  ],
  "disconnectAfterInSeconds": 3600,
```

```
  "refreshAfterInSeconds": 300
}
```

Creazione di un autorizzatore

È possibile creare un autorizzatore utilizzando l'API. CreateAuthorizer L'esempio seguente descrive il comando.

```
aws iot create-authorizer
--authorizer-name MyAuthorizer
--authorizer-function-arn arn:aws:lambda:us-
west-2:<account_id>:function:MyAuthorizerFunction  //The ARN of the Lambda function.
[--token-key-name MyAuthorizerToken //The key used to extract the token from headers.
[--token-signing-public-keys FirstKey=
 "-----BEGIN PUBLIC KEY-----
  [...insert your public key here...]
  -----END PUBLIC KEY-----"
[--status ACTIVE]
[--tags <value>]
[--signing-disabled | --no-signing-disabled]
```

Puoi utilizzare il parametro `signing-disabled` per disattivare la convalida della firma per ogni chiamata dell'autorizzatore. Si consiglia vivamente di non disattivare la firma a meno che non sia necessario. La convalida della firma ti protegge da invocazioni eccessive della funzione Lambda da dispositivi sconosciuti. Non è possibile aggiornare lo status `signing-disabled` di un autorizzatore dopo averlo creata. Per modificare questo comportamento, è necessario creare un altro autorizzatore personalizzato con un valore diverso per il parametro `signing-disabled`.

I valori per i parametri `tokenKeyName` e `tokenSigningPublicKeys` sono facoltativi se la firma è stata disabilitata. Sono valori obbligatori se la firma è abilitata.

Dopo aver creato la funzione Lambda e l'autorizzatore personalizzato, devi concedere esplicitamente al AWS IoT Core servizio l'autorizzazione a richiamare la funzione per tuo conto. Puoi farlo con il seguente comando.

> **Note**
>
> L'endpoint IoT predefinito potrebbe non supportare l'utilizzo di autorizzazioni personalizzate con funzioni Lambda. Puoi invece utilizzare le configurazioni di dominio per definire un nuovo endpoint e quindi specificare quell'endpoint per l'autorizzatore personalizzato.

```
aws lambda add-permission --function-name <lambda_function_name>
--principal iot.amazonaws.com --source-arn <authorizer_arn>
--statement-id Id-123 --action "lambda:InvokeFunction"
```

Autorizzazione AWS IoT a richiamare la funzione Lambda

In questa sezione, concederai l'autorizzazione della risorsa di autorizzazione personalizzata che hai appena creato per eseguire la funzione Lambda. Per concedere l'autorizzazione, è possibile utilizzare il comando dell'interfaccia a riga di comando add-permission.

Concedi l'autorizzazione alla tua funzione Lambda utilizzando il AWS CLI

1. Una volta inseriti i valori, inserisci il comando seguente. Attenzione: il valore `statement-id` deve essere univoco. Sostituisci *`Id-1234`* con il valore esatto che hai, altrimenti potresti ricevere un `ResourceConflictException` errore.

   ```
   aws lambda add-permission  \
   --function-name "custom-auth-function" \
   --principal "iot.amazonaws.com" \
   --action "lambda:InvokeFunction" \
   --statement-id "Id-1234" \
   --source-arn authorizerArn
   ```

2. Se il comando viene completato correttamente, restituisce un'istruzione di autorizzazione come questa. È possibile passare alla sezione successiva per testare l'autorizzazione ad hoc.

   ```
   {
       "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
   \":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
   \",\"Resource\":\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-
   auth-function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":
   \"arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function\"}}}"
   }
   ```

   Se il comando non ha esito positivo, restituisce un errore come questo. Dovrai esaminare e correggere l'errore prima di continuare.

   ```
   An error occurred (AccessDeniedException) when calling the AddPermission operation:
    User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
    lambda:AddPer
   ```

```
mission on resource: arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-
function
```

Verificare le autorizzazioni

Puoi utilizzare l'TestInvokeAuthorizerAPI per testare l'invocazione e restituire i valori del tuo autorizzatore. Questa API consente di specificare i metadati del protocollo e di testare la convalida della firma nell'autorizzatore.

Le seguenti schede mostrano come utilizzare per testare l'autorizzatore AWS CLI .

Unix-like

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER \
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Windows CMD

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER ^
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Windows PowerShell

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER `
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Il valore del parametro `token-signature` è il token firmato. Per ulteriori informazioni su come modificare questo valore, consulta the section called “Firma del token”.

Se l'autorizzatore accetta un nome utente e una password, puoi trasferire queste informazioni utilizzando il parametro `--mqtt-context`. Nelle seguenti schede viene illustrato come utilizzare l'API `TestInvokeAuthorizer` per inviare un oggetto JSON che contiene un nome utente, una password e un nome client all'autorizzatore personalizzato.

Unix-like

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  \
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Windows CMD

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  ^
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Windows PowerShell

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  `
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

La password deve essere con codifica base64. Nell'esempio seguente viene illustrato come codificare una password in un ambiente simile a UNIX.

```
echo -n PASSWORD | base64
```

Gestione degli autorizzatori personalizzati

Puoi gestire i tuoi autorizzatori utilizzando quanto segue. APIs

- ListAuthorizers: Mostra tutti gli autorizzatori presenti nel tuo account.
- DescribeAuthorizer: visualizza le proprietà dell'autorizzatore specificato. Questi valori includono data di creazione, data ultima modifica e altri attributi.
- SetDefaultAuthorizer: specifica l'autorizzatore predefinito per gli endpoint di dati AWS IoT Core . AWS IoT Core utilizza questo autorizzatore se un dispositivo non trasmette AWS IoT Core le credenziali e non specifica un autorizzatore. Per ulteriori informazioni sull'utilizzo delle AWS IoT Core credenziali, vedere. the section called “Autenticazione client”
- UpdateAuthorizer: modifica lo stato, il nome della chiave del token o le chiavi pubbliche per l'autorizzatore specificato.
- DeleteAuthorizer: elimina l'autorizzatore specificato.

> **Note**
>
> Non è possibile aggiornare i requisiti di firma di un autorizzatore. Ciò significa che non è possibile disabilitare la firma in un autorizzatore esistente che la richiede. Inoltre, non è possibile richiedere la firma a un autorizzatore esistente che non la richiede.

Autenticazione personalizzata con certificati client X.509

Quando si collegano dispositivi a AWS IoT Core, sono disponibili più tipi di autenticazione. È possibile utilizzare certificati client X.509 che possono essere utilizzati per autenticare le connessioni di client e dispositivi o definire autorizzatori personalizzati per gestire la propria logica di autenticazione e autorizzazione dei client. Questo argomento spiega come utilizzare l'autenticazione personalizzata con i certificati client X.509.

L'utilizzo dell'autenticazione personalizzata con i certificati X.509 può essere utile se hai già autenticato i tuoi dispositivi utilizzando certificati X.509 e desideri eseguire ulteriori convalide e autorizzazioni personalizzate. Ad esempio, se si archiviano i dati dei dispositivi, come i numeri di serie, nel certificato client X.509, dopo aver AWS IoT Core autenticato il certificato client X.509, è possibile utilizzare un autorizzatore personalizzato per identificare dispositivi specifici in base alle informazioni memorizzate nel campo del certificato. CommonName L'utilizzo dell'autenticazione personalizzata con certificati X.509 può migliorare la gestione della sicurezza dei dispositivi durante la connessione dei dispositivi AWS IoT Core e offre maggiore flessibilità nella gestione della logica di autenticazione e autorizzazione. AWS IoT Core supporta l'autenticazione personalizzata con certificati X.509 utilizzando il certificato X.509 e il tipo di autenticazione con autorizzazione personalizzata, che funziona sia con il protocollo MQTT che con il protocollo HTTPS. Per ulteriori informazioni sui tipi di autenticazione e sui protocolli applicativi supportati dagli endpoint dei dispositivi, vedere Protocolli di comunicazione dei AWS IoT Core dispositivi.

> **Note**
>
> L'autenticazione personalizzata con certificati client X.509 non è supportata nelle regioni. AWS GovCloud (US)

> **Important**
>
> È necessario utilizzare un endpoint creato utilizzando configurazioni di dominio. Inoltre, i client devono fornire l'estensione SNI (Server Name Indication) durante la connessione a. AWS IoT Core

Il processo di autenticazione dei dispositivi utilizzando l'autenticazione personalizzata con certificati client X.509 prevede i seguenti passaggi.

- Fase 1: Registrare i certificati client X.509 con AWS IoT Core
- Passaggio 2: creazione di una funzione Lambda
- Fase 3: Creare un autorizzatore personalizzato
- Fase 4: Impostare il tipo di autenticazione e il protocollo applicativo in una configurazione di dominio

Fase 1: Registrare i certificati client X.509 con AWS IoT Core

Se non l'hai già fatto, registra e attiva i certificati client X.509 con. AWS IoT Core Altrimenti, passare alla fase successiva.

Per registrare e attivare i certificati client con AWS IoT Core, procedi nel seguente modo:

1. Se crei certificati client direttamente con AWS IoT. Questi certificati client verranno registrati automaticamente con AWS IoT Core.
2. Se crei i tuoi certificati client, segui queste istruzioni per registrarli AWS IoT Core.
3. Per attivare i certificati client, segui queste istruzioni.

Passaggio 2: creazione di una funzione Lambda

AWS IoT Core utilizza autorizzatori personalizzati per implementare schemi di autenticazione e autorizzazione personalizzati. Un autorizzatore personalizzato è associato a una funzione Lambda che determina se un dispositivo è autenticato e quali operazioni il dispositivo è autorizzato a eseguire. Quando un dispositivo si connette a AWS IoT Core, AWS IoT Core recupera i dettagli dell'autorizzatore, incluso il nome dell'autorizzatore e la funzione Lambda associata, e richiama la funzione Lambda. La funzione Lambda riceve un evento che contiene un oggetto JSON con i dati del

certificato client X.509 del dispositivo. La tua funzione Lambda utilizza questo oggetto JSON di eventi per valutare la richiesta di autenticazione, decidere le azioni da intraprendere e inviare una risposta.

Esempio di evento della funzione Lambda

L'oggetto JSON di esempio seguente contiene tutti i campi possibili che possono essere inclusi. L'oggetto JSON effettivo conterrà solo i campi relativi alla richiesta di connessione specifica.

```
{
 "token": "aToken",
 "signatureVerified": true,
 "protocols": [
  "tls",
  "mqtt"
 ],
 "protocolData": {
  "tls": {
   "serverName": "serverName",
   "x509CertificatePem": "x509CertificatePem",
   "principalId": "principalId"
  },
  "mqtt": {
   "clientId": "myClientId",
                    "username": "myUserName",
                    "password": "myPassword"
  }
 },
 "connectionMetadata": {
  "id": "UUID"
 }
}
```

`signatureVerified`

Un valore booleano che indica se la firma del token configurata nell'autorizzatore è verificata o meno prima di richiamare la funzione Lambda dell'autorizzatore. Se l'autorizzatore è configurato per disabilitare la firma tramite token, questo campo sarà falso.

`protocols`

Un array che contiene i protocolli previsti per la richiesta.

`protocolData`

Un oggetto che contiene informazioni sui protocolli utilizzati nella connessione. Fornisce dettagli specifici del protocollo che possono essere utili per l'autenticazione, l'autorizzazione e altro ancora.

`tls`- Questo oggetto contiene informazioni relative al protocollo TLS (Transport Layer Security).

- `serverName`- La stringa del nome host SNI (Server Name Indication). AWS IoT Core richiede che i dispositivi inviino l'estensione SNI al protocollo Transport Layer Security (TLS) e forniscano l'indirizzo completo dell'endpoint sul campo. `host_name`
- `x509CertificatePem`- Il certificato X.509 in formato PEM, utilizzato per l'autenticazione del client nella connessione TLS.
- `principalId`- L'identificatore principale associato al client nella connessione TLS.

`mqtt`- Questo oggetto contiene informazioni relative al protocollo MQTT.

- `clientId`- È necessario includere una stringa solo nel caso in cui il dispositivo invii questo valore.
- `username`- Il nome utente fornito nel pacchetto MQTT Connect.
- `password`- La password fornita nel pacchetto MQTT Connect.

`connectionMetadata`

Metadati della connessione.

`id`- L'ID di connessione, che è possibile utilizzare per la registrazione e la risoluzione dei problemi.

> **Note**
>
> In questo caso, oggetto JSON, `x509CertificatePem` e `principalId` ci sono due nuovi campi nella richiesta. Il valore di `principalId` è uguale al valore di`certificateId`. Per ulteriori informazioni, consulta Certificato.

Esempio di risposta alla funzione Lambda

La funzione Lambda deve utilizzare le informazioni dell'oggetto JSON dell'evento per autenticare la connessione in ingresso e decidere quali azioni sono consentite nella connessione.

Il seguente oggetto JSON contiene un esempio di risposta che la funzione Lambda può inviare.

```
{
 "isAuthenticated": true,
 "principalId": "xxxxxxxx",
 "disconnectAfterInSeconds": 86400,
 "refreshAfterInSeconds": 300,
 "policyDocuments": [
  {
   "Version": "2012-10-17",
   "Statement": [
    {
     "Effect": "Allow",
     "Action": "iot:Publish",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/customauthtesting"
    }
   ]
  }
 ]
}
```

In questo esempio, questa funzione dovrebbe inviare una risposta che contenga i seguenti valori.

`isAuthenticated`

Un valore booleano che indica se la richiesta è autenticata.

`principalId`

Una stringa alfanumerica che funge da identificatore per il token inviato dalla richiesta di autorizzazione personalizzata. Il valore deve essere una stringa alfanumerica con almeno uno e non più di 128 caratteri. Identifica la connessione nei registri. Il valore di `principalId` deve essere uguale al valore dell'oggetto JSON dell'`principalId`evento (ad esempio CertificateID del certificato X.509).

`policyDocuments`

Un elenco di documenti di policy in formato JSON. AWS IoT Core Il valore è facoltativo e supporta le variabili thing policy e le variabili certificate policy. Il numero massimo di documenti relativi alle politiche è 10. Ogni documento di policy può contenere un massimo di 2.048 caratteri. Se hai più politiche collegate al certificato client e alla funzione Lambda, l'autorizzazione è una raccolta di tutte le politiche. Per ulteriori informazioni sulla creazione di AWS IoT Core politiche, consulta Politiche.

`disconnectAfterInSeconds`

Un numero intero che specifica la durata massima (in secondi) della connessione al AWS IoT Core gateway. Il valore minimo è 300 secondi e il valore massimo è 86.400 secondi. `disconnectAfterInSeconds`è valido per tutta la durata di una connessione e non viene aggiornato in caso di aggiornamenti consecutivi delle policy.

`refreshAfterInSeconds`

Un numero intero che specifica l'intervallo tra gli aggiornamenti delle policy. Al termine di questo intervallo, AWS IoT Core richiama la funzione Lambda per consentire gli aggiornamenti delle policy. Il valore minimo è 300 secondi e il valore massimo è 86400 secondi.

Funzione Lambda di esempio

Di seguito è riportato un esempio di funzione Lambda di Node.js. La funzione esamina il certificato X.509 del client ed estrae le informazioni pertinenti come il numero di serie, l'impronta digitale e il nome del soggetto. Se le informazioni estratte corrispondono ai valori previsti, al client viene concesso l'accesso per la connessione. Questo meccanismo garantisce che solo i client autorizzati con certificati validi possano stabilire una connessione.

```
const crypto = require('crypto');

exports.handler = async (event) => {

    // Extract the certificate PEM from the event
    const certPem = event.protocolData.tls.x509CertificatePem;

    // Parse the certificate using Node's crypto module
    const cert = new crypto.X509Certificate(certPem);

    var effect = "Deny";
    // Allow permissions only for a particular certificate serial, fingerprint, and
 subject
    if (cert.serialNumber === "7F8D2E4B9C1A5036DE8F7C4B2A91E5D80463BC9A1257" // This is
 a random serial
       && cert.fingerprint ===
 "F2:9A:C4:1D:B5:E7:08:3F:6B:D0:4E:92:A7:C1:5B:8D:16:0F:E3:7A" // This is a random
 fingerprint
       && cert.subject === "allow.example.com") {
      effect = "Allow";
    }
```

```
    return generateAuthResponse(event.protocolData.tls.principalId, effect);
};


// Helper function to generate the authorization response.
function generateAuthResponse(principalId, effect) {
    const authResponse = {
        isAuthenticated: true,
        principalId,
        disconnectAfterInSeconds: 3600,
        refreshAfterInSeconds: 300,
        policyDocuments: [
          {
            Version: "2012-10-17",
            Statement: [
              {
                Action: ["iot:Connect"],
                Effect: effect,
                Resource: [
                  "arn:aws:iot:us-east-1:123456789012:client/myClientName"
                ]
              },
              {
                Action: ["iot:Publish"],
                Effect: effect,
                Resource: [
                  "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
                ]
              },
              {
                Action: ["iot:Subscribe"],
                Effect: effect,
                Resource: [
                   "arn:aws:iot:us-east-1:123456789012:topicfilter/telemetry/
myClientName"
                ]
              },
              {
                Action: ["iot:Receive"],
                Effect: effect,
                Resource: [
                   "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
                ]
```

```
              }
            ]
          }
        ]
      };

  return authResponse;
}
```

La precedente funzione Lambda restituisce il seguente codice JSON quando riceve un certificato con il numero di serie, l'impronta digitale e l'oggetto previsti. Il valore di `x509CertificatePem` sarà il certificato client fornito nell'handshake TLS. Per ulteriori informazioni, consulta Definizione della funzione Lambda.

```
{
 "isAuthenticated": true,
 "principalId": "principalId in the event JSON object",
 "policyDocuments": [
  {
   "Version": "2012-10-17",
   "Statement": [
    {
     "Action": "iot:Connect",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:client/myClientName"
    },
    {
     "Action": "iot:Publish",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
    },
    {
     "Action": "iot:Subscribe",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/telemetry/
myClientName"
    },
    {
     "Action": "iot:Receive",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
    }
   ]
```

```
  }
 ],
 "disconnectAfterInSeconds": 3600,
 "refreshAfterInSeconds": 300
}
```

Fase 3: Creare un autorizzatore personalizzato

Dopo aver definito la funzione Lambda, crea un autorizzatore personalizzato per gestire la logica di autenticazione e autorizzazione del client. Puoi seguire le istruzioni dettagliate nella Fase 3: Creare una risorsa di autorizzazione del cliente e la relativa autorizzazione. Per ulteriori informazioni, consulta Creazione di un autorizzatore.

Nel processo di creazione dell'autorizzatore personalizzato, è necessario concedere l' AWS IoT autorizzazione a richiamare la funzione Lambda dopo la sua creazione. Per istruzioni dettagliate, consulta Autorizzazione AWS IoT a richiamare la funzione Lambda.

Fase 4: Impostare il tipo di autenticazione e il protocollo applicativo in una configurazione di dominio

Per autenticare i dispositivi utilizzando l'autenticazione personalizzata con certificati client X.509, è necessario impostare il tipo di autenticazione e il protocollo applicativo in una configurazione di dominio e inviare l'estensione SNI. Il valore di `authenticationType` deve essere e `CUSTOM_AUTH_X509` il valore di `applicationProtocol` può essere o. `SECURE_MQTT` `HTTPS`

Imposta il tipo di autenticazione e il protocollo dell'applicazione nella configurazione del dominio (CLI)

Se non disponi di una configurazione di dominio, usa il create-domain-configurationcomando per crearne una. Il valore di `authenticationType` deve essere `CUSTOM_AUTH_X509` e il valore di `applicationProtocol` può essere `SECURE_MQTT` o`HTTPS`.

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT \
    --authorizer-config '{
        "defaultAuthorizerName": my-custom-authorizer
    }'
```

Se hai già una configurazione di dominio, usa il update-domain-configurationcomando update `authenticationType` e `applicationProtocol` se necessario. Tieni presente che non puoi modificare il tipo o il protocollo di autenticazione sull'endpoint predefinito (`iot:Data-ATS`).

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT \
    --authorizer-config '{
        "defaultAuthorizerName": my-custom-authorizer
    }'
```

`domain-configuration-name`

Il nome della configurazione di dominio.

`authentication-type`

Il tipo di autenticazione della configurazione del dominio. Per ulteriori informazioni, vedi Scelta del tipo di autenticazione.

`application-protocol`

Il protocollo applicativo con cui i dispositivi comunicano AWS IoT Core. Per ulteriori informazioni, vedere Scelta di un protocollo applicativo.

`--authorizer-config`

Un oggetto che specifica la configurazione dell'autorizzatore in una configurazione di dominio.

`defaultAuthorizerName`

Il nome dell'autorizzatore per una configurazione di dominio.

Per ulteriori informazioni, consulta CreateDomainConfiguratione UpdateDomainConfigurationconsulta l'AWS IoT API Reference. Per ulteriori informazioni sulla configurazione del dominio, consulta Configurazioni del dominio.

Connessione a AWS IoT Core tramite autenticazione personalizzata

I dispositivi possono connettersi AWS IoT Core utilizzando l'autenticazione personalizzata con qualsiasi protocollo che AWS IoT Core supporti la messaggistica dei dispositivi. Per ulteriori informazioni sui protocolli di comunicazione supportati, consulta the section called “Protocolli di dispositivo di comunicazione”. I dati di connessione che passi all'autorizzatore della funzione Lambda dipendono dal protocollo utilizzato. Per ulteriori informazioni sulla creazione dell'autorizzatore della funzione Lambda, consulta the section called “Scrittura della funzione

Lambda”. Nelle sezioni seguenti viene illustrato come connettersi per l'autenticazione utilizzando ogni protocollo supportato.

HTTPS

I dispositivi che inviano dati AWS IoT Core tramite l'API HTTP Publish possono passare le credenziali tramite le intestazioni di richiesta o i parametri di query nelle richieste HTTP POST. I dispositivi possono specificare un'autorizzatore da richiamare utilizzando l'intestazione `x-amz-customauthorizer-name` o i parametri di query. Se nella tua autorizzazione è abilitata la firma dei token, devi passare *`token-key-name`* e `x-amz-customauthorizer-signature` nelle intestazioni di richiesta o nei parametri di query. Tieni presente che il *`token-signature`* valore deve essere codificato come URL quando viene utilizzato JavaScript dall'interno del browser.

> **Note**
>
> La funzione delle autorizzazioni del cliente per il protocollo HTTPS supporta solo le operazioni di pubblicazione. Per ulteriori informazioni sul protocollo HTTP, consultare the section called “Protocolli di dispositivo di comunicazione”.

Le seguenti richieste di esempio mostrano come passare questi parametri sia nelle intestazioni di richiesta che nei parametri di query.

```
//Passing credentials via headers
POST /topics/topic?qos=qos HTTP/1.1
Host: your-endpoint
x-amz-customauthorizer-signature: token-signature
token-key-name: token-value
x-amz-customauthorizer-name: authorizer-name

//Passing credentials via query parameters
POST /topics/topic?qos=qos&x-amz-customauthorizer-signature=token-signature&token-key-name=token-value HTTP/1.1
```

MQTT

I dispositivi a cui si connettono AWS IoT Core tramite una connessione MQTT possono trasmettere le credenziali attraverso i campi `username` e `password` dei messaggi MQTT. Il valore `username` può facoltativamente contenere una stringa di query che invia valori aggiuntivi (tra cui un token, una firma

e il nome di un autorizzatore) al tuo autorizzatore. Puoi utilizzare questa stringa di query se desideri utilizzare uno schema di autenticazione basato su token anziché i valori `username` e `password`.

> **Note**
>
> I dati nel campo della password sono codificati in base64 da. AWS IoT Core Devono essere decodificati dalla funzione Lambda.

L'esempio seguente contiene una stringa `username` che contiene parametri aggiuntivi che specificano un token e una firma.

```
username?x-amz-customauthorizer-name=authorizer-name&x-amz-customauthorizer-
signature=token-signature&token-key-name=token-value
```

Per richiamare un autorizzatore, i dispositivi che si connettono tramite MQTT e l' AWS IoT Core autenticazione personalizzata devono connettersi alla porta 443. Devono inoltre passare l'estensione TLS Application Layer Protocol Negotiation (ALPN) con un valore di `mqtt` e l'estensione Server Name Indication (SNI) con il nome host del loro endpoint di dati. AWS IoT Core Per evitare potenziali errori, il valore per `x-amz-customauthorizer-signature` deve essere codificato in formato URL. Inoltre, è raccomandabile che anche i valori di `x-amz-customauthorizer-name` e `token-key-name` siano codificati nello stesso formato. Per ulteriori informazioni su questi valori, consulta the section called “Protocolli di dispositivo di comunicazione”. AWS IoT Client per dispositivi SDKs SDKs, dispositivi mobili e AWS IoT dispositivi mobili V2 può configurare entrambe queste estensioni.

MQTT over WebSockets

I dispositivi che si connettono AWS IoT Core tramite MQTT over WebSockets possono passare le credenziali in uno dei due modi seguenti.

- Tramite le intestazioni di richiesta o i parametri di query nella richiesta HTTP UPGRADE per stabilire la connessione. WebSockets
- Attraverso i campi `username` e `password` nel messaggio MQTT CONNECT.

Se si passano le credenziali tramite il messaggio di connessione MQTT, sono necessarie le estensioni ALPN e SNI TLS. Per ulteriori informazioni su queste estensioni, consulta the section called “MQTT”. Nell'esempio seguente viene illustrato come trasferire le credenziali tramite la richiesta di aggiornamento HTTP.

```
GET /mqtt HTTP/1.1
Host: your-endpoint
Upgrade: WebSocket
Connection: Upgrade
x-amz-customauthorizer-signature: token-signature
token-key-name: token-value
sec-WebSocket-Key: any random base64 value
sec-websocket-protocol: mqtt
sec-WebSocket-Version: websocket version
```

Firma del token

Devi firmare il token con la chiave privata della coppia di chiavi pubblica-privata che hai utilizzato nella chiamata `create-authorizer`. Gli esempi seguenti mostrano come creare la firma del token utilizzando un comando simile a Unix e. JavaScript Utilizzano l'algoritmo hash SHA-256 per codificare la firma.

Command line

```
echo -n TOKEN_VALUE | openssl dgst -sha256 -sign PEM encoded RSA private key |
 openssl base64
```

JavaScript

```
const crypto = require('crypto')

const key = "PEM encoded RSA private key"

const k = crypto.createPrivateKey(key)
let sign = crypto.createSign('SHA256')
sign.write(t)
sign.end()
const s = sign.sign(k, 'base64')
```

Risoluzione dei problemi relativi agli autorizzatori

In questo argomento vengono illustrati i problemi comuni che possono causare problemi nei flussi di lavoro di autenticazione personalizzata e i passaggi per risolverli. Per risolvere i problemi nel modo più efficace, abilita CloudWatch i log per AWS IoT Core e imposta il livello di registro su DEBUG. È possibile abilitare CloudWatch i log nella console (). AWS IoT Core https://console.aws.amazon.com/iot/ Per ulteriori informazioni sull'attivazione e l'utilizzo dei registri per AWS IoT Core, consulta the section called “Configurare la registrazione AWS IoT”.

> **Note**
>
> Se si lascia il livello di registro su DEBUG per lunghi periodi di tempo, CloudWatch potrebbe memorizzare grandi quantità di dati di registrazione. Ciò può aumentare i costi. CloudWatch Prendi in considerazione l'utilizzo della registrazione basata sulle risorse per aumentare la verbosità solo per i dispositivi di un particolare gruppo di oggetti. Per ulteriori informazioni sulle registrazioni basate sulle risorse, consulta the section called “Configurare la registrazione AWS IoT”. Inoltre, al termine della risoluzione dei problemi, ridurre il livello di registro a un livello meno verboso.

Prima di iniziare la risoluzione dei problemi, consulta the section called “Informazioni sul flusso di lavoro di autenticazione personalizzato” per una visualizzazione generale del processo di autenticazione personalizzata. Questo ti aiuta a capire dove cercare la fonte di un problema.

In questo argomento vengono illustrate le due seguenti aree da esaminare.

- Problemi relativi alla funzione Lambda dell’autorizzatore.
- Problemi relativi al tuo dispositivo.

Controllo della presenza di problemi nella funzione Lambda dell'autorizzatore

Esegui la procedura seguente per assicurarti che i tentativi di connessione dei dispositivi richiamino la funzione Lambda.

1. Verifica quale funzione Lambda è associata al tuo autorizzatore.

   Puoi farlo chiamando l'DescribeAuthorizerAPI o facendo clic sull'autorizzatore desiderato nella sezione Secure della AWS IoT Core console.

2. Controlla i parametri di chiamata per la funzione Lambda. Esegui i seguenti passaggi.

   a. Apri la AWS Lambda console (https://console.aws.amazon.com/lambda/) e seleziona la funzione associata al tuo autorizzatore.

   b. Scegli la scheda Monitor (Monitorare) e visualizza i parametri relativi al periodo di tempo rilevante per il problema.

3. Se non vedi alcuna chiamata, verifica di AWS IoT Core avere il permesso di richiamare la tua funzione Lambda. Se visualizzi i richiami, vai al passaggio successivo. Esegui la procedura seguente per verificare che la funzione Lambda disponga delle autorizzazioni richieste.

   a. Scegli la scheda Autorizzazioni per la tua funzione nella console. AWS Lambda

   b. Trova la sezione Resource-based Policy (Policy basata su risorse) nella parte inferiore della pagina. Se la funzione Lambda dispone delle autorizzazioni richieste, la policy sarà simile all'esempio seguente.

```
{
  "Version":"2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "Id123",
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-
east-1:111111111111:function:FunctionName",
      "Condition": {
        "ArnLike": {
          "AWS:SourceArn": "arn:aws:iot:us-east-1:111111111111:authorizer/
AuthorizerName"
        },
        "StringEquals": {
          "AWS:SourceAccount": "111111111111"
        }
      }
    }
  ]
}
```

c. Questa politica concede l'`InvokeFunction`autorizzazione sulla tua funzione al AWS IoT Core preside. Se non la vedi, dovrai aggiungerla utilizzando l'AddPermissionAPI. Nell'esempio seguente viene illustrato come eseguire questa operazione utilizzando l' AWS CLI.

```
aws lambda add-permission --function-name FunctionName --principal
 iot.amazonaws.com --source-arn AuthorizerARn --statement-id Id-123 --action
 "lambda:InvokeFunction"
```

4. Se visualizzi i richiami, verifica che non vi siano errori. Un errore potrebbe indicare che la funzione Lambda non gestisce correttamente l'evento di connessione che la AWS IoT Core invia.

   Per informazioni sulla gestione dell'evento nella funzione Lambda, consulta the section called "Scrittura della funzione Lambda". È possibile utilizzare la funzionalità test nella AWS Lambda console (https://console.aws.amazon.com/lambda/) per codificare i valori di test della funzione per assicurarsi che la funzione gestisca gli eventi correttamente.

5. Se visualizzi i richiami senza errori, ma i tuoi dispositivi non sono in grado di connettersi (o pubblicare, sottoscrivere e ricevere messaggi), il problema potrebbe essere che la policy restituita dalla funzione Lambda non fornisce autorizzazioni per le operazioni che i tuoi dispositivi stanno tentando di eseguire. Eseguire la procedura seguente per determinare se qualcosa non è corretto nella policy restituita dalla funzione.

   a. Utilizza una query Amazon CloudWatch Logs Insights per scansionare i log in un breve periodo di tempo per verificare eventuali errori. La seguente query di esempio ordina gli eventi in base al timestamp e cerca errori.

   ```
   display clientId, eventType, status, @timestamp | sort @timestamp desc | filter
    status = "Failure"
   ```

   b. Aggiorna la tua funzione Lambda per registrare i dati a cui ritorna AWS IoT Core e l'evento che attiva la funzione. È possibile utilizzare questi registri per esaminare le policy create dalla funzione.

6. Se visualizzi invocazioni senza errori ma i tuoi dispositivi non sono in grado di connettersi (o pubblicare, sottoscrivere e ricevere messaggi), un altro motivo potrebbe essere che la funzione Lambda supera il limite di timeout. Il limite di timeout della funzione Lambda per l'autorizzatore personalizzato è di 5 secondi. Puoi controllare la durata della funzione nei CloudWatch log o nelle metriche.

Analisi dei problemi relativi ai dispositivi

Se non riscontri problemi con il richiamo della funzione Lambda o con la policy restituita dalla funzione, cerca i problemi con la connessione dei tuoi dispositivi. Le richieste di connessione AWS IoT Core non corrette possono impedire l'attivazione dell'autorizzatore. I problemi di connessione possono verificarsi sia a livello TLS che a livello di applicazione.

Possibili problemi di livello TLS:

- I clienti devono inserire un'intestazione del nome host (HTTP, MQTT over WebSockets) o l'estensione TLS Server Name Indication (HTTP, MQTT over WebSockets, MQTT) in tutte le richieste di autenticazione personalizzate. In entrambi i casi, il valore passato deve corrispondere a uno degli endpoint di dati del tuo account. AWS IoT Core Questi sono gli endpoint restituiti quando si eseguono i seguenti comandi CLI.
  - `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
  - `aws iot describe-endpoint --endpoint-type iot:Data`(per gli VeriSign endpoint precedenti)
- I dispositivi che utilizzano l'autenticazione personalizzata nelle connessioni MQTT devono anche inviare l'estensione TLS Application Layer Protocol Negotiation (ALPN) con un valore pari a `mqtt`.
- L'autenticazione personalizzata è attualmente disponibile solo sulla porta 443.

Possibili problemi relativi al livello dell'applicazione:

- Se la firma è abilitata (il campo `signingDisabled` è false nel tuo autorizzatore), cerca i seguenti problemi di firma.
  - Assicurati di passare la firma del token nell'intestazione`x-amz-customauthorizer-signature` o in un parametro di stringa di query.
  - Assicurati che il servizio non firmi un valore diverso dal token.
  - Assicurati di passare il token nel parametro di intestazione o query specificato nel campo `token-key-name` dell'autorizzatore.
- Assicurati che il nome dell'autorizzatore che si passa nell'intestazione `x-amz-customauthorizer-name` o nel parametro della stringa di query sia valido o che disponga di un autorizzatore predefinito definito per il proprio account.

Autorizzazione

L'autorizzazione è il processo di concessione delle autorizzazioni a un'identità autenticata. Concedi le autorizzazioni per AWS IoT Core l'utilizzo AWS IoT Core delle politiche IAM. In questo argomento vengono illustrate le policy AWS IoT Core . Per ulteriori informazioni sulla creazione di policy IAM, consulta Gestione delle identità e degli accessi per AWS IoT e Come AWS IoT funziona con IAM.

AWS IoT Core le politiche determinano cosa può fare un'identità autenticata. Un'identità autenticata viene usata da dispositivi, applicazioni per dispositivi mobili, applicazioni Web e applicazioni desktop. Un'identità autenticata può anche essere un utente che digita i comandi CLI AWS IoT Core . Un'identità può eseguire AWS IoT Core operazioni solo se dispone di una politica che le concede l'autorizzazione per tali operazioni.

Sia AWS IoT Core le policy che le policy IAM vengono utilizzate AWS IoT Core per controllare le operazioni che un'identità (chiamata anche principale) può eseguire. Il tipo di policy utilizzato dipende dal tipo di identità con AWS IoT Core cui si effettua l'autenticazione.

AWS IoT Core le operazioni sono suddivise in due gruppi:

- L'API control-plane ti permette di eseguire attività di amministrazione come la creazione o l'aggiornamento di certificati, oggetti, regole e così via.
- L'API Data Plane consente di inviare e ricevere dati da AWS IoT Core.

Il tipo di policy usato dipende dall'API in uso, control-plane o data-plane.

La tabella seguente mostra i tipi di identità, i protocolli usati da ciascuno e i tipi di policy che possono essere usati per l'autorizzazione.

AWS IoT Core API del piano dati e tipi di policy

Protocollo e meccanismo di autenticazione	SDK	Tipo di identità	Tipo di policy		
MQTT su TLS/TCP, autenticazione	AWS IoT SDK del dispositivo	Certificati X.509	AWS IoT Core politica		

Protocollo e meccanismo di autenticazione	SDK	Tipo di identità	Tipo di policy		
reciproca TLS (porta 8883 o 443)[†])					
MQTT su HTTPS/ WebSocket, autenticazione AWS SigV4 (porta 443)	AWS SDK per dispositivi mobili	Amazon Cognito Identity autenticato	IAM e politiche AWS IoT Core		
		Amazon Cognito Identity non autenticato	Policy IAM		
		Identità IAM o federata	Policy IAM		
HTTPS, autenticazione AWS Signature versione 4 (porta 443)	AWS CLI	Amazon Cognito, IAM o identità federata	Policy IAM		
Autenticazione reciproca HTTPS, TLS (porta 8443)	Nessun SDK supportato	Certificati X.509	AWS IoT Core politica		

Protocollo e meccanismo di autentica zione	SDK	Tipo di identità	Tipo di policy		
HTTPS tramite autentica zione personali zzata (porta 443)	AWS IoT SDK del dispositivo	Autorizza tore personali zzato	Policy dell'auto rizzatore personali zzato		

AWS IoT Core API e tipi di policy del piano di controllo

Protocollo e meccanismo di autentica zione	SDK	Tipo di identità	Tipo di policy		
Autentica zione HTTPS AWS Signature versione 4 (porta 443)	AWS CLI	Identità Amazon Cognito	Policy IAM		
		Identità IAM o federata	Policy IAM		

AWS IoT Core le politiche sono allegate ai certificati X.509, alle identità di Amazon Cognito o ai gruppi di oggetti. Le policy IAM sono collegate a un utente, un gruppo o un ruolo IAM. Se utilizzi la AWS IoT console o la AWS IoT Core CLI per allegare la policy (a un certificato, Amazon Cognito Identity o Thing Group), utilizzi una policy. AWS IoT Core Altrimenti, utilizzi una policy IAM. AWS IoT Core le politiche allegate a un gruppo di cose si applicano a qualsiasi cosa all'interno di quel gruppo di cose. Affinché la AWS IoT Core policy abbia effetto, il nome `clientId` e il nome dell'oggetto devono corrispondere.

L'autorizzazione basata sulle policy è uno strumento potente Ti offre il controllo completo sulle operazioni che dispositivi, utenti o applicazioni possono eseguire in AWS IoT Core. Ad esempio, si consideri un dispositivo a cui si AWS IoT Core connette un certificato. Puoi permettere al dispositivo di accedere a tutti gli argomenti MQTT oppure puoi limitarne l'accesso a un singolo argomento. In un altro esempio supponi che un utente digiti comandi nella riga di comando. Utilizzando una policy, è possibile consentire o negare l'accesso a qualsiasi comando o AWS IoT Core risorsa per l'utente. Puoi inoltre controllare l'accesso di un'applicazione alle risorse AWS IoT Core .

Le modifiche apportate a una policy possono richiedere alcuni minuti per diventare effettive a causa della modalità in cui AWS IoT memorizza nella cache i documenti delle policy. In particolare, potrebbero essere necessari alcuni minuti per accedere a una risorsa a cui è stato concesso l'accesso di recente e una risorsa potrebbe essere accessibile per alcuni minuti dopo la revoca dell'accesso.

AWS formazione e certificazione

Per informazioni sull'autorizzazione AWS IoT Core, segui il corso Deep Dive into AWS IoT Core Authentication and Authorization sul sito web di AWS Training and Certification.

AWS IoT Core politiche

AWS IoT Core le politiche sono documenti JSON. Seguono le stesse convenzioni delle politiche IAM. AWS IoT Core supporta politiche denominate in modo che molte identità possano fare riferimento allo stesso documento di policy. Le policy denominate hanno più versioni in modo da semplificarne il rollback.

AWS IoT Core le politiche consentono di controllare l'accesso al piano AWS IoT Core dati. Il piano dati AWS IoT Core è costituito dalle operazioni che ti permettono di connetterti al broker di messaggi AWS IoT Core , inviare e ricevere messaggi MQTT e ottenere o aggiornare il Device Shadow di un oggetto.

Una AWS IoT Core policy è un documento JSON che contiene una o più dichiarazioni di policy. Ogni dichiarazione contiene:

- `Effect`, specifica se l'operazione sarà permessa o negata.
- `Action`, specifica l'operazione permessa o negata dalla policy.
- `Resource`, specifica la risorsa o le risorse in cui l'operazione è permessa o negata.

Le modifiche apportate a una policy possono richiedere dai 6 agli 8 minuti per diventare effettive, a causa del modo in cui i documenti relativi alle policy vengono AWS IoT memorizzati nella cache. In particolare, potrebbero essere necessari alcuni minuti per accedere a una risorsa a cui è stato concesso l'accesso di recente e una risorsa potrebbe essere accessibile per alcuni minuti dopo la revoca dell'accesso.

AWS IoT Core le politiche possono essere allegate ai certificati X.509, alle identità di Amazon Cognito e ai gruppi di oggetti. Le policy associate a un gruppo di oggetti si applicano a qualsiasi elemento all'interno di quel gruppo. Affinché la policy diventi effettiva, il `clientId` e il nome dell'oggetto devono corrispondere. Le policy AWS IoT Core seguono la stessa logica di valutazione delle policy IAM. Per impostazione predefinita, tutte le policy vengono negate implicitamente. Un'autorizzazione esplicita in una policy basata su identità o su risorse sostituisce questo comportamento predefinito. Un rifiuto esplicito in una policy sostituisce qualsiasi permesso. Per ulteriori informazioni, consulta Logica di valutazione delle policy nella AWS Identity and Access Management Guida per l'utente.

Argomenti

- AWS IoT Core azioni politiche
- AWS IoT Core risorse d'azione
- AWS IoT Core variabili politiche
- Prevenzione del confused deputy tra servizi
- AWS IoT Core esempi di politiche
- Autorizzazione con identità Amazon Cognito

AWS IoT Core azioni politiche

Le operazioni di policy seguenti sono definite da AWS IoT Core:

Operazioni di policy MQTT

`iot:Connect`

Rappresenta l'autorizzazione a connettersi al broker di AWS IoT Core messaggi. L'autorizzazione `iot:Connect` viene controllata ogni volta che viene inviata una richiesta `CONNECT` al broker. Il broker di messaggi non permette a due client con lo stesso ID client di restare connessi contemporaneamente. Dopo la connessione del secondo client, il broker chiude la connessione

esistente. L'autorizzazione `iot:Connect` può essere usata per garantire che solo i client autorizzati possano connettersi usando un ID client specifico.

`iot:DeleteConnection`

Rappresenta l'autorizzazione a disconnettere un client MQTT connesso da. AWS IoT Core L'`iot:DeleteConnection`autorizzazione viene verificata ogni volta che viene effettuata una richiesta di disconnessione forzata di un client. Quando si disconnette un client, AWS IoT Core chiude la connessione di rete del client e, facoltativamente, pulisce lo stato della sessione.

`iot:GetRetainedMessage`

Rappresenta l'autorizzazione per ottenere il contenuto di un singolo messaggio conservato. I messaggi conservati sono i messaggi che sono stati pubblicati con il flag RETAIN impostato e archiviati da. AWS IoT Core Per ottenere un elenco di tutti i messaggi conservati dell'account, consulta `iot:ListRetainedMessages`.

`iot:ListRetainedMessages`

Rappresenta l'autorizzazione per recuperare informazioni di riepilogo sui messaggi conservati dell'account, ma non sul contenuto dei messaggi. I messaggi conservati sono i messaggi che sono stati pubblicati con il flag RETAIN impostato e archiviati da. AWS IoT Core La risorsa ARN specificata per questa azione deve essere `*`. Per ottenere l'autorizzazione per ottenere il contenuto di un singolo messaggio conservato, consulta `iot:GetRetainedMessage`.

`iot:Publish`

Rappresenta l'autorizzazione per pubblicare in un argomento MQTT. Questa autorizzazione viene controllata ogni volta che viene inviata una richiesta PUBLISH al broker. Questa può essere usata per permettere ai client di pubblicare in modelli di argomento specifici.

> **Note**
>
> Per concedere l'autorizzazione `iot:Publish`, devi concedere anche l'autorizzazione `iot:Connect`.

`iot:Receive`

Rappresenta l'autorizzazione a ricevere un messaggio da AWS IoT Core. L'autorizzazione `iot:Receive` viene confermata ogni volta che viene recapitato un messaggio a un client.

Poiché questa autorizzazione viene controllata a ogni recapito, può essere usata per revocare le autorizzazioni ai client che hanno attualmente sottoscritto un argomento.

`iot:RetainPublish`

Rappresenta l'autorizzazione per pubblicare un messaggio MQTT con il set di flag RETAIN.

> **Note**
>
> Per concedere l'autorizzazione `iot:RetainPublish`, devi concedere anche l'autorizzazione `iot:Publish`.

`iot:Subscribe`

Rappresenta l'autorizzazione a sottoscrivere un filtro di argomenti. Questa autorizzazione viene controllata ogni volta che viene inviata una richiesta SUBSCRIBE al broker. Questa può essere usata per permettere ai client di sottoscrivere argomenti corrispondenti a modelli di argomento specifici.

> **Note**
>
> Per concedere l'autorizzazione `iot:Subscribe`, devi concedere anche l'autorizzazione `iot:Connect`.

Operazioni di policy per Device Shadow

`iot:DeleteThingShadow`

Rappresenta l'autorizzazione a eliminare il Device Shadow di un oggetto. L'autorizzazione `iot:DeleteThingShadow` viene controllata ogni volta che viene effettuata una richiesta di eliminazione dei contenuti di un Device Shadow di un oggetto.

`iot:GetThingShadow`

Rappresenta l'autorizzazione a recuperare un Device Shadow di un oggetto. L'autorizzazione `iot:GetThingShadow` viene controllata ogni volta che viene effettuata una richiesta di recupero dei contenuti di un Device Shadow di un oggetto.

`iot:ListNamedShadowsForThing`

Rappresenta l'autorizzazione per elencare le copie shadow di un oggetto. L'autorizzazione `iot:ListNamedShadowsForThing` viene controllata ogni volta che viene effettuata una richiesta per elencare la copia shadow di un oggetto.

`iot:UpdateThingShadow`

Rappresenta l'autorizzazione ad aggiornare una copia shadow di un dispositivo. L'autorizzazione `iot:UpdateThingShadow` viene controllata ogni volta che viene effettuata una richiesta di aggiornamento dei contenuti del Device Shadow di un oggetto.

> **Note**
>
> La policy sull'esecuzione delle operazioni si applica solo all'endpoint TLS HTTP. Se utilizzi l'endpoint MQTT devi utilizzare le operazioni di policy MQTT definite in questo argomento. Per un esempio di criterio di policy di esecuzione dei processi che dimostra ciò, consulta the section called “Esempi di policy di processo base” che funziona con il protocollo MQTT.

Azioni AWS IoT Core politiche sulle esecuzioni di Job Executions

`iotjobsdata:DescribeJobExecution`

Rappresenta l'autorizzazione a recuperare l'esecuzione di un processo per un dato oggetto. L'autorizzazione `iotjobsdata:DescribeJobExecution` viene controllata ogni volta che viene richiesto di ottenere l'esecuzione di un processo.

`iotjobsdata:GetPendingJobExecutions`

Rappresenta l'autorizzazione a recuperare l'elenco dei processi che non si trovano in uno stato terminale per un oggetto. L'autorizzazione `iotjobsdata:GetPendingJobExecutions` viene controllata ogni volta che viene effettuata una richiesta di recupero dell'elenco.

`iotjobsdata:UpdateJobExecution`

Rappresenta l'autorizzazione ad aggiornare l'esecuzione di un processo. L'autorizzazione `iotjobsdata:UpdateJobExecution` viene controllata ogni volta che viene effettuata una richiesta di aggiornamento dell'esecuzione di un processo.

`iotjobsdata:StartNextPendingJobExecution`

Rappresenta l'autorizzazione a ottenere e avviare la successiva esecuzione in sospeso di un processo per un oggetto (ossia, per aggiornare l'esecuzione di un processo con stato da QUEUED a IN_PROGRESS). L'autorizzazione `iotjobsdata:StartNextPendingJobExecution` viene controllata ogni volta che viene effettuata una richiesta di avvio della successiva esecuzione in sospeso di un processo.

AWS IoT Core Azione politica per i fornitori di credenziali

`iot:AssumeRoleWithCertificate`

Rappresenta l'autorizzazione a chiamare il provider di AWS IoT Core credenziali per assumere un ruolo IAM con l'autenticazione basata su certificati. L'`iot:AssumeRoleWithCertificate`autorizzazione viene verificata ogni volta che viene effettuata una richiesta al provider di AWS IoT Core credenziali per assumere un ruolo.

AWS IoT Core risorse d'azione

Per specificare una risorsa per un'azione AWS IoT Core politica, utilizza l'Amazon Resource Name (ARN) della risorsa. Tutte le risorse ARNs seguono il seguente formato:

```
arn:partition:iot:region:AWS-account-ID:Resource-type/Resource-name
```

La tabella seguente mostra la risorsa da specificare per ogni tipo di azione. Gli esempi ARN riguardano l'ID dell'account`123456789012`, nella partizione `aws` e sono specifici della regione. `us-east-1` Per ulteriori informazioni sui formati per ARNs, consulta Amazon Resource Names (ARNs) dalla AWS Identity and Access Management User Guide.

Azione	Tipo di risorsa	Nome risorsa	Esempio di ARN
`iot:Connect`	`client`	ID client del client	`arn:aws:iot:us-east-1:123456789012:client/myClientId`

Azione	Tipo di risorsa	Nome risorsa	Esempio di ARN
iot:DeleteConnection	client	ID client del client	arn:aws:iot:us-east-1:123456789012:client/myClientId
iot:DeleteThingShadow	thing	Il nome dell'oggetto e il nome della copia shadow, se applicabile	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iotjobsdata:DescribeJobExecution	thing	Nome dell'oggetto	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iotjobsdata:GetPendingJobExecutions	thing	Nome dell'oggetto	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetRetainedMessage	topic	Un argomento del messaggio conservato	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:GetThingShadow	thing	Il nome dell'oggetto e il nome della copia shadow, se applicabile	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iot:ListNamedShadowsForThing	Tutti	Tutti	*

Azione	Tipo di risorsa	Nome risorsa	Esempio di ARN
`iot:ListRetainedMessages`	Tutti	Tutti	`*`
`iot:Publish`	`topic`	Una stringa di argomenti	`arn:aws:iot:us-east-1:123456789012:topic/myTopicName`
`iot:Receive`	`topic`	Una stringa di argomenti	`arn:aws:iot:us-east-1:123456789012:topic/myTopicName`
`iot:RetainPublish`	`topic`	Argomento da pubblicare con il set di flag RETAIN	`arn:aws:iot:us-east-1:123456789012:topic/myTopicName`
`iotjobsdata:StartNextPendingJobExecution`	`thing`	Nome dell'oggetto	`arn:aws:iot:us-east-1:123456789012:thing/thingOne`
`iot:Subscribe`	`topicfilter`	Una stringa di filtro dell'argomento	`arn:aws:iot:us-east-1:123456789012:topicfilter/myTopicFilter`
`iotjobsdata:UpdateJobExecution`	`thing`	Nome dell'oggetto	`arn:aws:iot:us-east-1:123456789012:thing/thingOne`

Azione	Tipo di risorsa	Nome risorsa	Esempio di ARN
`iot:UpdateThingShadow`	`thing`	Il nome dell'oggetto e il nome della copia shadow, se applicabile	`arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne`
`iot:AssumeRoleWithCertificate`	`rolealias`	Un alias del ruolo che punta a un ARN di ruolo	`arn:aws:iot:us-east-1:123456789012:rolealias/CredentialProviderRole_alias`

AWS IoT Core variabili politiche

AWS IoT Core definisce le variabili di policy che possono essere utilizzate nelle AWS IoT Core politiche del `Condition` blocco `Resource` or. Quando una policy viene valutata, le variabili vengono sostituite dai valori effettivi. Ad esempio, se un dispositivo è connesso al broker di AWS IoT Core messaggi con un ID client di 100-234-3456, la variabile di policy viene sostituita nel documento di `iot:ClientId` policy da 100-234-3456.

AWS IoT Core le policy possono utilizzare caratteri jolly e seguire una convenzione simile alle policy IAM. L'inserimento di un * (asterisco) nella stringa può essere considerato come carattere jolly corrispondente a qualsiasi carattere. Puoi ad esempio usare * per descrivere più nomi di argomenti MQTT nell'attributo `Resource` di una policy. I caratteri + e # sono trattati come stringhe letterali in una policy. Per una policy di esempio che illustra come utilizzare i caratteri jolly, consultaUtilizzo di caratteri jolly in MQTT e nelle policy AWS IoT Core.

È inoltre possibile utilizzare variabili di policy predefinite con valori fissi per rappresentare caratteri che altrimenti hanno un significato speciale. Questi caratteri speciali sono `${*}`, `${?}` e `${$}`. Per ulteriori informazioni sulle variabili di policy e sui caratteri speciali, consulta Elementi delle policy IAM: variabili e tag e Creazione di una condizione con più chiavi o valori.

Argomenti

- Variabili AWS IoT Core politiche di base

- Variabili delle policy di oggetto
- Variabili della politica del certificato AWS IoT Core X.509

Variabili AWS IoT Core politiche di base

AWS IoT Core definisce le seguenti variabili politiche di base:

- `aws:SourceIp`: l'indirizzo IP del client connesso al broker di AWS IoT Core messaggi.
- `iot:ClientId`: ID client usato per la connessione al broker di messaggi AWS IoT Core .
- `iot:DomainName`: il nome di dominio del client a cui è connesso AWS IoT Core.

Esempi

- Esempi ClientId di variabili SourceIp politiche
- Esempi di variabili iot:DomainName politiche

Esempi **`ClientId`** di variabili **`SourceIp`** politiche

La seguente AWS IoT Core politica mostra una politica che utilizza variabili di politica. `aws:SourceIp`può essere utilizzato nell'elemento Condition della politica per consentire ai responsabili di effettuare richieste API solo all'interno di un intervallo di indirizzi specifico. Per alcuni esempi, consulta Autorizzazione di utenti e servizi cloud a utilizzare Jobs AWS IoT.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/clientid1"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
```

```
      "iot:Publish"
     ],
     "Resource": [
      "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
     ],
     "Condition": {
      "IpAddress": {
       "aws:SourceIp": "123.45.167.89"
      }
     }
    }
   ]
}
```

In questi esempi, `${iot:ClientId}` viene sostituito dall'ID del client connesso al broker di AWS IoT Core messaggi quando viene valutata la politica. Quando usi variabili delle policy come `${iot:ClientId}`, puoi aprire inavvertitamente l'accesso ad argomenti indesiderati. Ad esempio, se usi una policy che usa `${iot:ClientId}` per specificare un filtro di argomenti:

```
{
 "Effect": "Allow",
 "Action": [
  "iot:Subscribe"
 ],
 "Resource": [
  "arn:aws:iot:us-east-1:123456789012:topicfilter/my/${iot:ClientId}/topic"
 ]
}
```

Un client può connettersi usando + come ID client. Questo permette all'utente di sottoscrivere qualsiasi argomento corrispondente al filtro di argomenti `my/+/topic`. Per proteggerti da tali lacune di sicurezza, utilizza l'azione `iot:Connect` politica per controllare quale client IDs può connettersi. Ad esempio, questa policy permette di connettersi solo a quei client il cui ID client è `clientid1`:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
```

```
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/clientid"
   ]
  }
 ]
}
```

> **Note**
>
> L'utilizzo della variabile di policy `${iot:ClientId}` con `Connect` non è consigliato. Non esiste alcun controllo sul valore di `ClientId`, quindi un collegamento con un ID client diverso può superare la convalida ma causare la disconnessione. Poiché qualsiasi `ClientId` è consentito, l'impostazione di un ID client casuale può aggirare le policy del gruppo di oggetti.

Esempi di variabili **`iot:DomainName`** politiche

È possibile aggiungere la variabile di `iot:DomainName` policy per limitare i domini che possono essere utilizzati. L'aggiunta della variabile di `iot:DomainName` policy consente ai dispositivi di connettersi solo a endpoint configurati specifici.

La seguente politica consente ai dispositivi di connettersi al dominio specificato.

```
{
 "Version":"2012-10-17",
 "Statement": {
  "Sid": "AllowConnectionsToSpecifiedDomain",
  "Effect": "Allow",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "arn:aws:iot:us-east-1:123456789012:client/clientid",
  "Condition": {
   "StringEquals": {
    "iot:DomainName": "d1234567890abcdefghij-ats.iot.us-east-1.amazonaws.com"
   }
  }
```

```
 }
}
```

La seguente politica impedisce ai dispositivi di connettersi al dominio specificato.

```
{
 "Version":"2012-10-17",
 "Statement": {
  "Sid": "DenyConnectionsToSpecifiedDomain",
  "Effect": "Deny",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "arn:aws:iot:us-east-1:123456789012:client/clientid",
  "Condition": {
   "StringEquals": {
    "iot:DomainName": "d1234567890abcdefghij-ats.iot.us-east-1.amazonaws.com"
   }
  }
 }
}
```

Per ulteriori informazioni sull'operatore condizionale della policy, consulta IAM JSON Policy elements: Condition operators. Per ulteriori informazioni sulle configurazioni dei domini, consulta Cos'è una configurazione di dominio? .

Variabili delle policy di oggetto

Le variabili Thing Policy consentono AWS IoT Core di scrivere policy che concedono o negano le autorizzazioni in base alle proprietà degli oggetti, come i nomi degli oggetti, i tipi di oggetti e i valori degli attributi degli oggetti. È possibile utilizzare le variabili Thing Policy per applicare la stessa policy per controllare più dispositivi. AWS IoT Core Per ulteriori informazioni sul provisioning dei dispositivi, consulta Provisioning dei dispositivi.

Se si utilizza un'associazione di oggetti non esclusiva, è possibile allegare lo stesso certificato a più elementi. Per mantenere un'associazione chiara ed evitare potenziali conflitti, è necessario abbinare l'ID cliente al nome dell'oggetto. In questo caso, si ottiene il nome dell'oggetto dall'ID client nel `Connect` messaggio MQTT inviato quando un oggetto si connette a AWS IoT Core.

Tenere presente quanto segue quando si utilizzano le variabili delle policy dell'oggetto nelle policy AWS IoT Core .

- Usa l'AttachThingPrincipalAPI per allegare certificati o principali (identità autenticate di Amazon Cognito) a un oggetto.
- Se è attiva un'associazione di oggetti non esclusiva, quando sostituisci i nomi degli oggetti con variabili di policy degli oggetti, il valore di contenuto nel messaggio di connessione MQTT o `clientId` nella connessione TLS deve corrispondere esattamente al nome dell'oggetto.

Sono disponibili le variabili delle policy di oggetto seguenti:

- `iot:Connection.Thing.ThingName`

  Si risolve nel nome dell'elemento nel AWS IoT Core registro per il quale viene valutata la policy. AWS IoT Core utilizza il certificato che il dispositivo presenta al momento dell'autenticazione per determinare quale elemento utilizzare per verificare la connessione. Questa variabile di policy è disponibile solo quando un dispositivo si connette tramite MQTT o MQTT tramite il protocollo. WebSocket
- `iot:Connection.Thing.ThingTypeName`

  Questa variabile restituisce il tipo di oggetto associato all'oggetto per cui viene valutata la policy. L'ID client della WebSocket connessione MQTT/ deve essere lo stesso del nome dell'oggetto. Questa variabile di policy è disponibile solo quando ci si connette tramite MQTT o MQTT tramite il protocollo. WebSocket
- `iot:Connection.Thing.Attributes[`*`attributeName`*`]`

  Questa variabile restituisce il valore dell'attributo specificato associato all'oggetto per cui viene valutata la policy. A un oggetto possono essere associati fino a 50 attributi. Ogni attributo è disponibile come variabile di politica: `iot:Connection.Thing.Attributes[`*`attributeName`*`]` dove *`attributeName`* è il nome dell'attributo. L'ID client della MQTT/WebSocket connessione deve essere lo stesso del nome dell'oggetto. Questa variabile di policy è disponibile solo quando ci si connette tramite MQTT o MQTT tramite il WebSocket protocollo.
- `iot:Connection.Thing.IsAttached`

  `iot:Connection.Thing.IsAttached: ["true"]`impone che solo i dispositivi registrati AWS IoT e collegati al principale possano accedere alle autorizzazioni previste dalla policy.

Puoi utilizzare questa variabile per impedire la connessione a un dispositivo AWS IoT Core se presenta un certificato che non è collegato a un elemento IoT nel AWS IoT Core registro. Questa variabile ha dei valori `true` o `false` indica che l'elemento di connessione è collegato al certificato o all'identità di Amazon Cognito nel registro utilizzando l'API. AttachThingPrincipal Il nome dell'oggetto è preso come ID client.

Se l'ID client corrisponde al nome dell'oggetto o se alleghi il certificato esclusivamente a un oggetto, l'utilizzo di variabili di policy nella definizione della policy può semplificare la gestione delle policy. Invece di creare policy individuali per ogni elemento IoT, puoi definire una singola policy utilizzando le variabili della policy del thing. Questa policy può essere applicata a tutti i dispositivi in modo dinamico. Di seguito è riportato un esempio di policy per mostrarne il funzionamento. Per ulteriori informazioni, consulta ???.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Condition": {
    "StringLike": {
     "iot:ClientId": "*${iot:Connection.Thing.Attributes[envType]}"
    }
   },
   "Effect": "Allow",
   "Action": "iot:Connect",
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/*"
  }
 ]
}
```

Questo esempio di policy consente agli elementi a cui connettersi AWS IoT Core se il loro ID client termina con il valore del loro `envType` attributo. Potranno connettersi solo gli elementi con un modello di ID client corrispondente.

Variabili della politica del certificato AWS IoT Core X.509

Le variabili delle politiche dei certificati X.509 aiutano a scrivere le politiche. AWS IoT Core Queste politiche concedono autorizzazioni basate sugli attributi del certificato X.509. Le seguenti sezioni descrivono come utilizzare queste variabili della politica dei certificati.

> **Important**
>
> Se il certificato X.509 non include un particolare attributo di certificato ma nel documento di policy viene utilizzata la corrispondente variabile di politica del certificato, la valutazione della politica potrebbe portare a un comportamento imprevisto.

CertificateId

Nell'RegisterCertificateAPI, `certificateId` viene visualizzato nel corpo della risposta. Per ottenere informazioni sul certificato, utilizza il comando `certificateId` in DescribeCertificate.

Attributi dell'autorità emittente

Le seguenti variabili di AWS IoT Core policy supportano l'autorizzazione o la negazione delle autorizzazioni, in base agli attributi del certificato impostati dall'emittente del certificato.

- `iot:Certificate.Issuer.DistinguishedNameQualifier`
- `iot:Certificate.Issuer.Country`
- `iot:Certificate.Issuer.Organization`
- `iot:Certificate.Issuer.OrganizationalUnit`
- `iot:Certificate.Issuer.State`
- `iot:Certificate.Issuer.CommonName`
- `iot:Certificate.Issuer.SerialNumber`
- `iot:Certificate.Issuer.Title`
- `iot:Certificate.Issuer.Surname`
- `iot:Certificate.Issuer.GivenName`
- `iot:Certificate.Issuer.Initials`
- `iot:Certificate.Issuer.Pseudonym`
- `iot:Certificate.Issuer.GenerationQualifier`

Attributi soggetto

Le seguenti variabili di AWS IoT Core policy supportano la concessione o la negazione delle autorizzazioni, in base agli attributi dell'oggetto del certificato impostati dall'emittente del certificato.

- `iot:Certificate.Subject.DistinguishedNameQualifier`
- `iot:Certificate.Subject.Country`
- `iot:Certificate.Subject.Organization`
- `iot:Certificate.Subject.OrganizationalUnit`
- `iot:Certificate.Subject.State`
- `iot:Certificate.Subject.CommonName`
- `iot:Certificate.Subject.SerialNumber`
- `iot:Certificate.Subject.Title`
- `iot:Certificate.Subject.Surname`
- `iot:Certificate.Subject.GivenName`
- `iot:Certificate.Subject.Initials`
- `iot:Certificate.Subject.Pseudonym`
- `iot:Certificate.Subject.GenerationQualifier`

I certificati X.509 offrono a questi attributi la possibilità di contenere uno o più valori. Per impostazione predefinita, le variabili delle policy per ogni attributo con più valori restituiscono il primo valore. Ad esempio, l'attributo `Certificate.Subject.Country` potrebbe contenere un elenco di nomi di paese, ma quando viene valutato in una policy, `iot:Certificate.Subject.Country` viene sostituito con il nome del primo paese.

Puoi richiedere un valore di attributo specifico diverso dal primo usando un indice a base uno. Ad esempio, `iot:Certificate.Subject.Country.1` viene sostituito dal secondo nome di paese nell'attributo `Certificate.Subject.Country`. Se specifichi un valore di indice che non esiste, ad esempio se richiedi un terzo valore quando all'attributo ne sono assegnati solo due, non viene eseguita alcuna sostituzione e l'autorizzazione non riesce. Puoi anche usare il suffisso `.List` nel nome di variabile della policy per specificare tutti i valori dell'attributo.

Attributi dei nomi alternativi dell'autorità emittente

Le seguenti variabili di AWS IoT Core policy supportano la concessione o la negazione delle autorizzazioni, in base agli attributi alternativi del nome dell'emittente impostati dall'emittente del certificato.

- `iot:Certificate.Issuer.AlternativeName.RFC822Name`
- `iot:Certificate.Issuer.AlternativeName.DNSName`

- `iot:Certificate.Issuer.AlternativeName.DirectoryName`
- `iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Issuer.AlternativeName.IPAddress`

Attributi dei nomi alternativi di oggetto

Le seguenti variabili di AWS IoT Core policy supportano la concessione o la negazione delle autorizzazioni, in base agli attributi del nome alternativo del soggetto impostati dall'emittente del certificato.

- `iot:Certificate.Subject.AlternativeName.RFC822Name`
- `iot:Certificate.Subject.AlternativeName.DNSName`
- `iot:Certificate.Subject.AlternativeName.DirectoryName`
- `iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Subject.AlternativeName.IPAddress`

Altri attributi

È possibile utilizzare `iot:Certificate.SerialNumber` per consentire o negare l'accesso alle AWS IoT Core risorse, in base al numero di serie di un certificato. La variabile di policy `iot:Certificate.AvailableKeys` contiene il nome di tutte le variabili delle policy di certificato che contengono valori.

Utilizzo delle variabili di policy dei certificati X.509

Questo argomento fornisce dettagli su come utilizzare le variabili delle politiche dei certificati. Le variabili delle politiche di certificato X.509 sono essenziali quando si creano AWS IoT Core politiche che forniscono autorizzazioni basate sugli attributi del certificato X.509. Se il certificato X.509 non include un particolare attributo di certificato ma nel documento di policy viene utilizzata la corrispondente variabile di politica del certificato, la valutazione della politica potrebbe portare a un comportamento imprevisto. Questo perché la variabile di policy mancante non viene valutata nella dichiarazione politica.

In questo argomento:

- Esempio di certificato X.509
- Utilizzo degli attributi dell'emittente del certificato come variabili della politica del certificato

- Utilizzo degli attributi dell'oggetto del certificato come variabili della politica del certificato
- Utilizzo degli attributi di nome alternativi dell'emittente del certificato come variabili della politica del certificato
- Utilizzo degli attributi del nome alternativo del soggetto del certificato come variabili della politica del certificato
- Utilizzo di un altro attributo di certificato come variabile di politica del certificato
- Limitazioni per le variabili delle policy di certificato X.509
- Politiche di esempio che utilizzano variabili di policy relative ai certificati

Esempio di certificato X.509

Un tipico certificato X.509 potrebbe apparire come segue. Questo certificato di esempio include gli attributi del certificato. Durante la valutazione delle AWS IoT Core politiche, i seguenti attributi del certificato verranno compilati come variabili dei criteri di certificazione:`Serial Number`,`Issuer`, `SubjectX509v3 Issuer Alternative Name`, e`X509v3 Subject Alternative Name`.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            92:12:85:cb:b7:a5:e0:86
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=IoT Devices, OU=SmartHome, ST=WA, CN=IoT Devices Primary CA,
    GN=Primary CA1/initials=XY/dnQualifier=Example corp,
    SN=SmartHome/ title=CA1/pseudonym=Primary_CA/generationQualifier=2/serialNumber=987

        Validity
            Not Before: Mar 26 03:25:40 2024 GMT
            Not After : Apr 28 03:25:40 2025 GMT
        Subject: C=US, O=IoT Devices, OU=LightBulb, ST=NY, CN=LightBulb Device Cert,
    GN=Bulb/initials=ZZ/dnQualifier=Bulb001,
    SN=Multi Color/title=RGB/pseudonym=RGB Device/generationQualifier=4/
serialNumber=123
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    << REDACTED >>
                Exponent: 65537 (0x10001)
```

```
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Subject Alternative Name:
                DNS:example.com, IP Address:1.2.3.4, URI:ResourceIdentifier001,
email:device1@example.com, DirName:/C=US/O=IoT/OU=SmartHome/CN=LightBulbCert
            X509v3 Issuer Alternative Name:
                DNS:issuer.com, IP Address:5.6.7.8, URI:PrimarySignerCA,
email:primary@issuer.com, DirName:/C=US/O=Issuer/OU=IoT Devices/CN=Primary Issuer CA
    Signature Algorithm: sha256WithRSAEncryption
         << REDACTED >>
```

Utilizzo degli attributi dell'emittente del certificato come variabili della politica del certificato

La tabella seguente fornisce dettagli su come gli attributi dell'emittente del certificato verranno inseriti in una politica. AWS IoT Core

Attributi dell'emittente da inserire in una politica

Attributi dell'emittente del certificato	Variabili della politica dei certificati
• C=US • Dispositivi o=IoT • OU = SmartHome • ST=WA • CA principale per dispositivi CN=IoT • GN = primario CA1 • Iniziali = XY • DNQualifier=Example corp • SN= SmartHome • titolo= CA1 • pseudonimo=primary_CA • Qualificatore di generazione = 2	• `iot:Certificate.Issuer.Country = US` • `iot:Certificate.Issuer.Organization = IoT Devices` • `iot:Certificate.Issuer.OrganizationalUnit = SmartHome` • `iot:Certificate.Issuer.State = WA` • `iot:Certificate.Issuer.CommonName = IoT Devices Primary CA` • `iot:Certificate.Issuer.GivenName = Primary CA1` • `iot:Certificate.Issuer.initials = XY` • `iot:Certificate.Issuer.Distinguished NameQualifier = Example corp` • `iot:Certificate.Issuer.Surname = SmartHome`

Attributi dell'emittente del certificato	Variabili della politica dei certificati
• Numero di serie = 987	• `iot:Certificate.Issuer.Title = CA1` • `iot:Certificate.Issuer.Pseudonym = Primary_CA` • `iot:Certificate.Issuer.GenerationQualifier = 2` • `iot:Certificate.Issuer.SerialNumber = 987`

Utilizzo degli attributi dell'oggetto del certificato come variabili della politica del certificato

La tabella seguente fornisce dettagli su come gli attributi dell'oggetto del certificato verranno inseriti in una AWS IoT Core politica.

Attributi del soggetto da inserire in una politica

Attributi dell'oggetto del certificato	Variabili della politica dei certificati
• C=US • Dispositivi o=IoT • ST = NY • CN= Certificato del dispositivo LightBulb • GN=lampadina • Iniziali = ZZ • Qualificatore DN=bulb001 • sn=Multicolore • Titolo = RGB • Pseudonimo=Dispositivo RGB • GenerationQualifier = 4 • Numero di serie = 123	• `iot:Certificate.Subject.Country = US` • `iot:Certificate.Subject.Organization = IoT Devices` • `iot:Certificate.Subject.State = NY` • `iot:Certificate.Subject.CommonName = LightBulb Device Cert` • `iot:Certificate.Subject.GivenName = Bulb` • `iot:Certificate.Subject.initials = ZZ` • `iot:Certificate.Subject.DistinguishedNameQualifier = Bulb001` • `iot:Certificate.Subject.Surname = Multi Color` • `iot:Certificate.Subject.Title = RGB` • `iot:Certificate.Subject.Pseudonym = RGB Device`

Attributi dell'oggetto del certificato	Variabili della politica dei certificati
	• `iot:Certificate.Subject.GenerationQualifier = 4` • `iot:Certificate.Subject.SerialNumber = 123`

Utilizzo degli attributi di nome alternativi dell'emittente del certificato come variabili della politica del certificato

La tabella seguente fornisce dettagli su come gli attributi dei nomi alternativi dell'emittente del certificato verranno inseriti in una politica. AWS IoT Core

Attributi del nome alternativi dell'emittente da inserire in una policy

Nome alternativo dell'emittente X509v3	Attributo in una politica
• DNS: Issuer.com • Indirizzo IP: 5.6.7.8 • URI: CA PrimarySigner • e-mail: primary@issuer.com • DirName: /C= US/O=Issuer/OU=IoT Devices/CN =Emittente principale CA	• `iot:Certificate.Issuer.AlternativeName.DNSName = issuer.com` • `iot:Certificate.Issuer.AlternativeName.IPAddress = 5.6.7.8` • `iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier = PrimarySignerCA` • `iot:Certificate.Issuer.AlternativeName.RFC822Name = primary@issuer.com` • `iot:Certificate.Issuer.AlternativeName.DirectoryName = cn=Primary Issuer CA,ou=IoT Devices,o=Issuer,c=US`

Utilizzo degli attributi del nome alternativo del soggetto del certificato come variabili della politica del certificato

La tabella seguente fornisce dettagli su come gli attributi del nome alternativo del soggetto del certificato verranno inseriti in una politica. AWS IoT Core

Attributi del nome alternativo del soggetto da inserire in una policy

Nome alternativo del soggetto X509v3	Attributo in una politica
• DNS: example.com • Indirizzo IP: 1.2.3.4 • URI: 001 ResourceIdentifier • e-mail: device1@example.com • DirName: /C= = US/O=IoT/OU=SmartHome/CN LightBulbCert	• `iot:Certificate.Subject.AlternativeName.DNSName = example.com` • `iot:Certificate.Subject.AlternativeName.IPAddress = 1.2.3.4` • `iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier = ResourceIdentifier001` • `iot:Certificate.Subject.AlternativeName.RFC822Name = device1@example.com` • `iot:Certificate.Subject.AlternativeName.DirectoryName = cn=LightBulbCert,ou=SmartHome,o=IoT,c=US`

Utilizzo di un altro attributo di certificato come variabile di politica del certificato

La tabella seguente fornisce dettagli su come gli altri attributi del certificato verranno inseriti in una AWS IoT Core politica.

Altri attributi da inserire in una policy

Altro attributo del certificato	Variabile di politica del certificato
`Serial Number: 92:12:85:cb:b7:a5:e0:86`	`iot:Certificate.SerialNumber = 105256223 89124227206`

Limitazioni per le variabili delle policy di certificato X.509

Alle variabili delle policy di certificato X.509 si applicano le limitazioni seguenti:

Variabili di policy mancanti

Se il certificato X.509 non include un particolare attributo di certificato ma nel documento di policy viene utilizzata la corrispondente variabile di policy del certificato, la valutazione della policy potrebbe portare a un comportamento imprevisto. Questo perché la variabile di policy mancante non viene valutata nella dichiarazione politica.

Formato del certificato SerialNumber

AWS IoT Core considera il numero di serie del certificato come la rappresentazione in formato stringa di un numero intero decimale. Ad esempio, se una policy consente solo connessioni con ID client corrispondente al numero di serie del certificato, l'ID client deve essere il numero di serie in formato decimale.

Caratteri jolly

Se negli attributi del certificato sono presenti caratteri jolly, la variabile policy non viene sostituita dal valore dell'attributo certificate. Questo lascerà il `${policy-variable}` testo nel documento di policy. Questo comportamento può provocare un errore di autenticazione. È possibile utilizzare i seguenti caratteri jolly: `*`, `$`, `+`, `?` e `#`.

Campi di matrice

Gli attributi di certificato che contengono matrici sono limitati a cinque elementi. Gli elementi aggiuntivi vengono ignorati.

Lunghezza delle stringhe

Tutti i valori di stringa sono limitati a 1024 caratteri. Se un attributo di certificato contiene una stringa più lunga di 1024 caratteri, la variabile policy non viene sostituita dal valore dell'attributo certificate. Questo lascerà il nome `${policy-variable}` nel documento relativo alla policy. Questo comportamento può provocare un errore di autenticazione.

Caratteri speciali

Qualsiasi carattere speciale, ad esempio `,`, `'`, `\`, `+`, `=`, `<`, `>` e `;` deve essere preceduto da una barra rovesciata (`\`) quando viene utilizzato in una variabile di policy. Ad esempio `Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US` diventa `Amazon Web Service O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US`.

Politiche di esempio che utilizzano variabili di policy relative ai certificati

Il seguente documento di policy consente le connessioni con un ID client che corrisponde al numero di serie del certificato e la pubblicazione sull'argomento che corrisponde al modello:`${iot:Certificate.Subject.Organization}/device-stats/${iot:ClientId}/*`.

> **Important**
>
> Se il certificato X.509 non include un particolare attributo di certificato ma nel documento di policy viene utilizzata la corrispondente variabile di policy del certificato, la valutazione della policy potrebbe portare a un comportamento imprevisto. Questo perché la variabile di policy mancante non viene valutata nella dichiarazione politica. Ad esempio, se alleghi il seguente documento di policy a un certificato che non contiene l'`iot:Certificate.Subject.Organization`attributo, le variabili di policy del `iot:Certificate.Subject.Organization` certificato non verranno compilate durante la valutazione della policy.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Certificate.SerialNumber}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.Organization}/device-stats/${iot:ClientId}/*"
   ]
```

```
  }
 ]
}
```

È inoltre possibile utilizzare l'operatore di condizione Null per garantire che le variabili della politica dei certificati utilizzate in una politica vengano compilate durante la valutazione della politica. Il seguente documento di policy consente l'utilizzo `iot:Connect` di certificati solo quando sono presenti gli attributi Certificate Serial Number e Certificate Subject Common name.

Tutte le variabili della politica del certificato hanno valori String, quindi tutti gli operatori di condizione String sono supportati.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/*"
   ],
   "Condition": {
    "Null": {
     "iot:Certificate.SerialNumber": "false",
     "iot:Certificate.Subject.CommonName": "false"
    }
   }
  }
 ]
}
```

Prevenzione del confused deputy tra servizi

Il problema confused deputy è un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire un'azione può costringere un'entità maggiormente privilegiata a eseguire l'azione. Inoltre AWS, l'impersonificazione tra servizi può causare il confuso problema del vicesceriffo. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante)

effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare ciò, AWS fornisce alcuni strumenti che consentono di proteggere i dati per tutti i servizi che dispongono di principali del servizio a cui è stato consentito l'accesso alle risorse del tuo account.

Per limitare le autorizzazioni che AWS IoT assegnano un altro servizio alla risorsa, consigliamo di utilizzare le chiavi di contesto `aws:SourceArn`e le chiavi di contesto della condizione `aws:SourceAccount`globale nelle politiche delle risorse. Se si utilizzano entrambe le chiavi di contesto delle condizioni globali, il valore `aws:SourceAccount` e l'account nel valore `aws:SourceArn` devono utilizzare lo stesso ID account nella stessa istruzione di policy.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'Amazon Resource Name (ARN) completo della risorsa. Perché AWS IoT, è `aws:SourceArn` necessario rispettare il formato: `arn:`*`aws`*`:iot:`*`region`*`:`*`account-id`*`:`*`resource-type/resource-id`* per le autorizzazioni specifiche delle risorse o. `arn:`*`aws`*`:iot:`*`region`*`:`*`account-id`*`:*` Il resource-id può essere il nome o l'ID della risorsa consentita o una dichiarazione con caratteri jolly della risorsa consentita. IDs Assicurati che corrisponda alla tua AWS IoT regione e che *`region`* corrisponda all'ID del *`account-id`* tuo account cliente.

L'esempio seguente mostra come prevenire il problema confuso del vicesceriffo utilizzando le chiavi del contesto `aws:SourceArn` e della condizione `aws:SourceAccount` globale nella politica di fiducia dei AWS IoT ruoli. Per ulteriori esempi, consulta Esempi dettagliati di prevenzione confusa.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
        },
            "ArnLike":{
               "aws:SourceArn":"arn:aws:iot:us-east-1:123456789012:*"
```

```
        }
         }
      }
   ]
}
```

> **Note**
>
> Se si verificano errori di negazione dell'accesso, può essere dovuto al fatto che l'integrazione del servizio con AWS Security Token Service (STS) non supporta le chiavi `aws:SourceArn` e di `aws:SourceAccount` contesto.

Esempi dettagliati di prevenzione confusa

Questa sezione fornisce esempi dettagliati di come prevenire il problema confuso dei vicesceriffi utilizzando aws:SourceArn le chiavi relative al contesto della condizione aws:SourceAccount globale nella politica di fiducia nei AWS IoT ruoli.

- Provisioning del parco istanze
- JITP
- Fornitore di credenziali

Provisioning del parco istanze

È possibile configurare il provisioning della flotta utilizzando una risorsa modello di provisioning. Quando un modello di provisioning fa riferimento a un ruolo di provisioning, la policy di attendibilità di quel ruolo può includere le `aws:SourceArn` chiavi e condition. `aws:SourceAccount` Queste chiavi limitano le risorse per le quali la configurazione può richiamare la richiesta. `sts:AssumeRole`

Il ruolo con la seguente policy di trust può essere assunto solo dal principale IoT (`iot.amazonaws.com`) per il modello di provisioning specificato in. `SourceArn`

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
```

```
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
        },
            "ArnLike":{
               "aws:SourceArn":"arn:aws:iot:us-
east-1:123456789012:provisioningtemplate/example_template"
         }
          }
      }
   ]
}
```

JITP

Nel just-in-time provisioning (JITP), è possibile utilizzare il modello di provisioning come risorsa separata dalla CA o definire il corpo del modello e il ruolo come parte della configurazione del certificato CA. Il valore della policy di `aws:SourceArn` attendibilità del AWS IoT ruolo dipende da come si definisce il modello di provisioning.

Definizione del modello di provisioning come risorsa separata

Se si definisce il modello di provisioning come risorsa separata, il valore di `aws:SourceArn` può essere. `"arn:aws:iot:region:account-id:provisioningtemplate/example_template"` È possibile utilizzare lo stesso esempio di policy inProvisioning del parco istanze.

Definizione di un modello di provisioning in un certificato CA

Se si definisce il modello di provisioning all'interno di una risorsa di certificato CA, il valore di `aws:SourceArn` può essere `"arn:aws:iot:region:account-id:cacert/cert_id"` o. `"arn:aws:iot:region:account-id:cacert/*"` È possibile utilizzare un carattere jolly quando l'identificatore della risorsa, ad esempio l'ID di un certificato CA, è sconosciuto al momento della creazione.

Il ruolo con la seguente policy di fiducia può essere assunto solo dal principale IoT (`iot.amazonaws.com`) per il certificato CA specificato in`SourceArn`.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
        },
            "ArnLike":{
               "aws:SourceArn":"arn:aws:iot:us-
east-1:123456789012:cacert/
8ecde6884f3d87b1125ba31ac3fcb13d7016de7f57cc904fe1cb97c6ae98196e"
        }
         }
      }
   ]
}
```

Quando si crea un certificato CA, è possibile fare riferimento a un ruolo di provisioning nella configurazione di registrazione. La politica di fiducia del ruolo di provisioning può essere utilizzata `aws:SourceArn` per limitare le risorse per cui il ruolo può essere assunto. Tuttavia, durante la CACertificate chiamata iniziale di Register per registrare il certificato CA, non avresti l'ARN del certificato CA da specificare nella `aws:SourceArn` condizione.

Per ovviare a questo problema, ad esempio per specificare la policy di attendibilità del ruolo di provisioning per lo specifico certificato CA con cui è registrato AWS IoT Core, puoi fare quanto segue:

- Innanzitutto, chiama Register CACertificate senza fornire il `RegistrationConfig` parametro.
- Dopo aver registrato il certificato CA AWS IoT Core, chiama Update CACertificate su di esso.

  Nella CACertificate chiamata Update, fornisci una policy `RegistrationConfig` che includa la policy di attendibilità del ruolo di provisioning `aws:SourceArn` impostata sull'ARN del certificato CA appena registrato.

Fornitore di credenziali

Per il provider di AWS IoT Core credenziali, usa lo stesso Account AWS che usi per creare l'alias del ruolo in `aws:SourceAccount` e specifica un'istruzione che corrisponda all'ARN della risorsa del tipo di risorsa rolealias in. `aws:SourceArn` Quando crei un ruolo IAM da utilizzare con il provider di AWS IoT Core credenziali, devi includere nella `aws:SourceArn` condizione gli alias ARNs di ruolo che potrebbero dover assumere il ruolo, autorizzando così la richiesta interservizio. `sts:AssumeRole`

Il ruolo con la seguente politica di fiducia può essere assunto solo dal principale fornitore di AWS IoT Core credenziali (`credentials.iot.amazonaws.com`) per il RoleAlias specificato in. `SourceArn` Se un principale tenta di recuperare le credenziali per un alias di ruolo diverso da quello specificato nella `aws:SourceArn` condizione, la richiesta verrà rifiutata, anche se l'altro alias del ruolo fa riferimento allo stesso ruolo IAM.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:iot:us-
east-1:123456789012:rolealias/example_rolealias"
        }
      }
    }
  ]
}
```

AWS IoT Core esempi di politiche

I criteri di esempio in questa sezione illustrano i documenti della policy utilizzati per completare le attività comuni in AWS IoT Core. È possibile utilizzarli come esempi per iniziare dalla creazione delle policy per le tue soluzioni.

Gli esempi in questa sezione utilizzano questi elementi di policy:

- the section called “AWS IoT Core azioni politiche”
- the section called “AWS IoT Core risorse d'azione”
- the section called “Esempi di policy basate su identità”
- the section called “Variabili AWS IoT Core politiche di base”
- the section called “Variabili della politica del certificato AWS IoT Core X.509”

Esempi di policy riportati in questa sezione:

- Esempi di policy di connessione
- Esempi di policy di pubblicazione/sottoscrizione
- Esempi di policy di connessione e pubblicazione
- Esempi di policy per i messaggi conservati
- Esempi di policy di certificato
- Esempi di policy di oggetto
- Esempi di policy di processo base

Esempi di policy di connessione

La seguente politica nega l'autorizzazione al client IDs `client1` e `client2` a cui connettersi AWS IoT Core, mentre consente ai dispositivi di connettersi utilizzando un ID client. L'ID client corrisponde al nome di un elemento registrato nel AWS IoT Core registro e allegato al principale utilizzato per la connessione:

> **Note**
>
> Per i dispositivi registrati, è consigliabile utilizzare variabili delle policy di oggetto per azioni `Connect` e collegare l'oggetto al principale utilizzato per la connessione.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/client1",
    "arn:aws:iot:us-east-1:123456789012:client/client2"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  }
 ]
}
```

La seguente politica concede l'autorizzazione alla connessione AWS IoT Core con l'ID client. `client1` Questo esempio di policy è relativa a dispositivi non registrati.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
```

```
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/client1"
   ]
  }
 ]
}
```

Esempi di policy di sessioni persistenti MQTT

`connectAttributes` consente di specificare quali attributi si desidera utilizzare nel messaggio di connessione nelle policy IAM, ad esempio `PersistentConnect` e `LastWill`. Per ulteriori informazioni, consulta Utilizzo degli Attributi Connect.

La seguente policy permette di connettersi con la funzionalità `PersistentConnect`:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
    }
   }
  }
 ]
}
```

La seguente policy non consente `PersistentConnect`, sono consentite altre funzionalità:

```
{
```

```
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringNotEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
    }
   }
  }
 ]
}
```

La policy di cui sopra può essere espressa anche utilizzando `StringEquals`, è consentita qualsiasi altra funzionalità, inclusa la nuova funzionalità:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1"
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
```

```
                        "PersistentConnect"
                    ]
                }
            }
            }
    ]
}
```

La seguente policy permette la connessione sia tramite `PersistentConnect` che `LastWill`, qualsiasi altra nuova funzionalità non è consentita:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect",
      "LastWill"
     ]
    }
   }
  }
 ]
}
```

La seguente policy consente la connessione pulita da parte dei client con o senza `LastWill`, non saranno consentite altre funzionalità:

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
```

```
        "Action": [
            "iot:Connect"
        ],
        "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
        "Condition": {
            "StringEquals": {
                "iot:ConnectAttributes": "LastWill"
        }
        }
    }]
}
```

La seguente policy consente la connessione solo utilizzando le funzionalità predefinite:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": []
    }
   }
  }
 ]
}
```

La seguente policy permette di connettersi solo a `PersistentConnect`, qualsiasi nuova funzionalità è consentita fintanto che la connessione utilizza `PersistentConnect`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        }
    ]
}
```

La seguente policy indica che la connessione deve avere sia `PersistentConnect` che `LastWill`, non è consentita alcuna nuova funzionalità:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect",
      "LastWill"
     ]
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
```

```
    ],
    "Resource": "*",
    "Condition": {
     "ForAllValues:StringEquals": {
      "iot:ConnectAttributes": [
       "PersistentConnect"
      ]
     }
    }
   },
   {
    "Effect": "Deny",
    "Action": [
     "iot:Connect"
    ],
    "Resource": "*",
    "Condition": {
     "ForAllValues:StringEquals": {
      "iot:ConnectAttributes": [
       "LastWill"
      ]
     }
    }
   },
   {
    "Effect": "Deny",
    "Action": [
     "iot:Connect"
    ],
    "Resource": "*",
    "Condition": {
     "ForAllValues:StringEquals": {
      "iot:ConnectAttributes": []
     }
    }
   }
  ]
}
```

La seguente policy non deve avere `PersistentConnect` ma può avere `LastWill`, qualsiasi altra nuova funzionalità non è consentita:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "LastWill"
                    ]
            }
        }
        }
    ]
}
```

La seguente policy consente la connessione solo da client che dispongono di un `LastWill` con argomento "`my/lastwill/topicName`", qualsiasi funzionalità è consentita purché utilizzi l'argomento `LastWill`:

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ArnEquals": {
                "iot:LastWillTopic": "arn:aws:iot:us-east-1:123456789012:topic/my/
lastwill/topicName"
            }
        }
        }
    ]
}
```

La seguente policy consente solo la connessione pulita utilizzando un `LastWillTopic`, qualsiasi funzionalità è consentita purché utilizzi `LastWillTopic`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ArnEquals": {
                "iot:LastWillTopic": "arn:aws:iot:us-east-1:123456789012:topic/my/
lastwill/topicName"
            }
        }
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        }
    ]
}
```

Esempi di policy di pubblicazione/sottoscrizione

La politica che usi dipende dal modo in cui ti connetti. AWS IoT CoreÈ possibile connettersi AWS IoT Core utilizzando un client MQTT, HTTP o WebSocket. Quando ti connetti con un client MQTT, l'autenticazione viene eseguita con un certificato X.509. Quando ti connetti tramite HTTP o il WebSocket protocollo, ti autentichi con Signature Version 4 e Amazon Cognito.

> **Note**
>
> Per i dispositivi registrati, è consigliabile utilizzare variabili delle policy di oggetto per azioni `Connect` e collegare l'oggetto al principale utilizzato per la connessione.

In questa sezione:

- Utilizzo di caratteri jolly in MQTT e nelle policy AWS IoT Core
- Politiche per la pubblicazione, la sottoscrizione e la ricezione di messaggi (argomenti to/from specifici)
- Politiche per la pubblicazione, la sottoscrizione e la ricezione di messaggi, argomenti con un prefisso specifico to/from
- Politiche per la pubblicazione, la sottoscrizione e la ricezione di messaggi (argomenti specifici per ogni dispositivo) to/from
- Politiche per la pubblicazione, la sottoscrizione e la ricezione di messaggi (to/from argomenti con l'attributo thing nel nome dell'argomento)
- Policy per rifiutare la pubblicazione di messaggi in argomenti secondari di un nome di argomento
- Policy per rifiutare la ricezione di messaggi da argomenti secondari di un nome di argomento

- Policy per effettuare la sottoscrizione agli argomenti utilizzando caratteri jolly MQTT
- WebSocket Politiche per HTTP e client

Utilizzo di caratteri jolly in MQTT e nelle policy AWS IoT Core

MQTT e AWS IoT Core le policy hanno caratteri jolly diversi e dovreste sceglierli dopo un'attenta valutazione. In MQTT, i caratteri + jolly # vengono utilizzati nei filtri degli argomenti MQTT per sottoscrivere più nomi di argomenti. AWS IoT Core le politiche utilizzano * e ? come caratteri jolly e seguono le convenzioni delle politiche IAM. In un documento di policy, * rappresenta qualsiasi combinazione di caratteri e il punto interrogativo ? rappresenta qualsiasi singolo carattere. Nei documenti delle policy, i caratteri jolly MQTT, + e # sono trattati come caratteri senza alcun significato speciale. Per descrivere più nomi e filtri di argomenti nell'attributo `resource` di una policy, utilizza i caratteri jolly * e ? al posto dei caratteri jolly MQTT.

Quando scegliete i caratteri jolly da utilizzare in un documento di policy, tenete presente che il * carattere non è limitato a un singolo livello di argomento. Il + carattere è limitato a un singolo livello di argomento in un filtro per argomenti MQTT. Per contribuire a vincolare una specifica di carattere jolly a un singolo livello di filtro di argomento MQTT, considera l'utilizzo di più caratteri ?. Per ulteriori informazioni sull'uso dei caratteri jolly in una risorsa politica e altri esempi di ciò che corrispondono, consultate Using jolly in resource. ARNs

La tabella seguente mostra i diversi caratteri jolly utilizzati in MQTT e le policy AWS IoT Core per client MQTT.

Carattere jolly.	È un carattere jolly MQTT	Esempio in MQTT	È un carattere jolly AWS IoT Core di policy	Esempio AWS IoT Core di policy per i client MQTT
#	Sì	`some/#`	No	N/D
+	Sì	`some/+/topic`	No	N/D
*	No	N/D	Sì	`topicfilter/some/*/topic` `topicfilter/some/sensor*/topic`

Carattere jolly.	È un carattere jolly MQTT	Esempio in MQTT	È un carattere jolly AWS IoT Core di policy	Esempio AWS IoT Core di policy per i client MQTT

?	No	N/D	Sì	`topic/some/?????/topic`
				
				`topicfilter/some/sensor???/topic`

Politiche per la pubblicazione, la sottoscrizione e la ricezione di messaggi (argomenti to/from specifici)

Di seguito sono riportati esempi di dispositivi registrati e non registrati per pubblicare, sottoscrivere e ricevere messaggi to/from sull'argomento denominato «some_specific_topic». Gli esempi evidenziano anche che `Publish` e `Receive` utilizzano "topic" come la risorsa e che `Subscribe` utilizza "topicfilter" come la risorsa.

Registered devices

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi con ClientID che corrisponde al nome di un elemento nel registro. Inoltre, fornisce le autorizzazioni `Publish`, `Subscribe` e `Receive` per l'argomento denominato "some_specific_topic".

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
```

```
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/some_specific_topic"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
   ]
  }
 ]
}
```

Unregistered devices

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi utilizzando ClientID1, ClientID2 o ClientID3. Inoltre, fornisce le autorizzazioni `Publish`, `Subscribe` e `Receive` per l'argomento denominato "some_specific_topic".

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/
some_specific_topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
            ]
        }
    ]
}
```

Politiche per la pubblicazione, la sottoscrizione e la ricezione di messaggi, argomenti con un prefisso specifico to/from

Di seguito sono riportati esempi di dispositivi registrati e non registrati per pubblicare, sottoscrivere e ricevere messaggi to/from su argomenti con il prefisso «topic_prefix».

> **Note**
>
> Nota l'uso del carattere jolly in questo esempio. * Sebbene * sia utile fornire le autorizzazioni per più nomi di argomenti in un'unica istruzione, può portare a conseguenze indesiderate fornendo ai dispositivi più privilegi del necessario. Si consiglia quindi di utilizzare il carattere * jolly solo dopo un'attenta valutazione.

Registered devices

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi con ClientID che corrisponde al nome di un elemento nel registro. Inoltre, fornisce le autorizzazioni `Publish`, `Subscribe` e `Receive` per gli argomenti preceduti dal prefisso "topic_prefix".

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
```

```
    "iot:Publish",
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix*"
   ]
  }
 ]
}
```

Unregistered devices

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi utilizzando ClientID1, ClientID2 o ClientID3. Inoltre, fornisce le autorizzazioni `Publish`, `Subscribe` e `Receive` per gli argomenti preceduti dal prefisso "topic_prefix".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix*"
            ]
        }
    ]
}
```

Politiche per la pubblicazione, la sottoscrizione e la ricezione di messaggi (argomenti specifici per ogni dispositivo) to/from

Di seguito sono riportati esempi di dispositivi registrati e non registrati per la pubblicazione, la sottoscrizione e la ricezione di messaggi to/from su argomenti specifici per un determinato dispositivo.

Registered devices

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi con ClientID che corrisponde al nome di un elemento nel registro. Fornisce l'autorizzazione per pubblicare nell'argomento specifico dell'oggetto (`sensor/device/${iot:Connection.Thing.ThingName}`), nonché effettuare la sottoscrizione e ricevere messaggi dall'argomento specifico dell'oggetto (`command/device/${iot:Connection.Thing.ThingName}`). Se il nome dell'oggetto nel registro è «thing1", il dispositivo sarà in grado di pubblicare sull'argomento" 1". sensor/device/thing1". The device will also be able to subscribe to and receive from the topic "command/device/thing

```
{
```

```
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/sensor/device/
${iot:Connection.Thing.ThingName}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/command/device/
${iot:Connection.Thing.ThingName}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/command/device/
${iot:Connection.Thing.ThingName}"
```

```
   ]
  }
 ]
}
```

Unregistered devices

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi utilizzando ClientID1, ClientID2 o ClientID3. Fornisce l'autorizzazione per pubblicare nell'argomento specifico del client (`sensor/device/${iot:ClientId}`), nonché effettuare la sottoscrizione e ricevere messaggi dall'argomento specifico del client (`command/device/${iot:ClientId}`). Se il dispositivo si connette con ClientID come ClientID1, sarà in grado di pubblicare sull'argomento "1". sensor/device/clientId Il dispositivo sarà inoltre in grado di abbonarsi e ricevere messaggi dall'argomento. `device/clientId1/command`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/sensor/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

Politiche per la pubblicazione, la sottoscrizione e la ricezione di messaggi (to/from argomenti con l'attributo thing nel nome dell'argomento)

Di seguito viene illustrato un esempio di dispositivi registrati per la pubblicazione, la sottoscrizione e la ricezione di to/from argomenti di messaggi i cui nomi includono attributi di oggetto.

> **Note**
>
> Gli attributi degli oggetti esistono solo per i dispositivi registrati nel AWS IoT Core Registro di sistema. Non esiste un esempio corrispondente per dispositivi non registrati.

Registered devices

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi con ClientID che corrisponde al nome di un elemento nel registro. Fornisce l'autorizzazione per pubblicare nell'argomento (`sensor/${iot:Connection.Thing.Attributes[version]}`), nonché effettuare la sottoscrizione e ricevere messaggi dall'argomento (`command/`

`${iot:Connection.Thing.Attributes[location]}`) dove il nome di argomento include gli attributi dell'oggetto. Se il nome dell'oggetto nel registro è impostato su `version=v1` and`location=Seattle`, il dispositivo sarà in grado di pubblicare nell'argomento "sensor/v1", and subscribe to and receive from the topic "command/Seattle».

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/sensor/
${iot:Connection.Thing.Attributes[version]}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/command/
${iot:Connection.Thing.Attributes[location]}"
   ]
  },
```

```
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/command/
${iot:Connection.Thing.Attributes[location]}"
   ]
  }
 ]
}
```

Unregistered devices

Poiché gli attributi dell'oggetto esistono solo per i dispositivi registrati nel AWS IoT Core registro, non esiste un esempio corrispondente per gli elementi non registrati.

Policy per rifiutare la pubblicazione di messaggi in argomenti secondari di un nome di argomento

Di seguito vengono illustrati esempi di dispositivi registrati e non registrati per pubblicare messaggi in tutti gli argomenti tranne determinati argomenti secondari.

Registered devices

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi con ClientID che corrisponde al nome di un elemento nel registro. Fornisce l'autorizzazione per pubblicare in tutti gli argomenti preceduti dal prefisso "department/" ma non nell'argomento secondario "department/admins".

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
```

```
    ],
    "Condition": {
     "Bool": {
      "iot:Connection.Thing.IsAttached": "true"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": [
     "iot:Publish"
    ],
    "Resource": [
     "arn:aws:iot:us-east-1:123456789012:topic/department/*"
    ]
   },
   {
    "Effect": "Deny",
    "Action": [
     "iot:Publish"
    ],
    "Resource": [
     "arn:aws:iot:us-east-1:123456789012:topic/department/admins"
    ]
   }
 ]
}
```

Unregistered devices

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi utilizzando ClientID1, ClientID2 o ClientID3. Fornisce l'autorizzazione per pubblicare in tutti gli argomenti preceduti dal prefisso "department/" ma non nell'argomento secondario "department/admins".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/department/*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/department/admins"
            ]
        }
    ]
}
```

Policy per rifiutare la ricezione di messaggi da argomenti secondari di un nome di argomento

Di seguito vengono illustrati esempi di dispositivi registrati e non registrati per effettuare la sottoscrizione e ricevere messaggi da argomenti con prefissi specifici tranne determinati argomenti secondari.

Registered devices

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi con ClientID che corrisponde al nome di un elemento nel registro. La policy consente ai dispositivi di effettuare la sottoscrizione a qualsiasi argomento preceduto dal prefisso "topic_prefix". Utilizzando `NotResource` nell'istruzione for`iot:Receive`, consentiamo al dispositivo di ricevere messaggi da tutti gli argomenti a cui il dispositivo è abbonato, ad eccezione

degli argomenti con il prefisso «topic_». prefix/restricted". For example, with this policy, devices can subscribe to "topic_prefix/topic1" and even "topic_prefix/restricted", however, they will only receive messages from the topic "topic_prefix/topic1" and no messages from the topic "topic_prefix/restricted

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": "iot:Subscribe",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix/*"
  },
  {
   "Effect": "Allow",
   "Action": "iot:Receive",
   "NotResource": "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix/
restricted/*"
  }
 ]
}
```

Unregistered devices

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi utilizzando ClientID1, ClientID2 o ClientID3. La policy consente ai dispositivi di effettuare la sottoscrizione a qualsiasi argomento preceduto dal prefisso

"topic_prefix". Utilizzando `NotResource` nell'istruzione for`iot:Receive`, consentiamo al dispositivo di ricevere messaggi da tutti gli argomenti a cui il dispositivo è abbonato, ad eccezione degli argomenti con il prefisso «topic_». prefix/restricted". For example, with this policy, devices can subscribe to "topic_prefix/topic1" and even "topic_prefix/restricted". However, they will only receive messages from the topic "topic_prefix/topic1" and no messages from the topic "topic_prefix/restricted

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/
topic_prefix/*"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "NotResource": "arn:aws:iot:us-east-1:123456789012:topic/
topic_prefix/restricted/*"
        }
    ]
}
```

Policy per effettuare la sottoscrizione agli argomenti utilizzando caratteri jolly MQTT

I caratteri jolly MQTT + e # vengono trattati come stringhe letterali, ma non vengono trattati come caratteri jolly quando vengono utilizzati nelle policy. AWS IoT Core In MQTT, i caratteri + e # vengono

trattati come caratteri jolly solo quando si effettua la sottoscrizione a un filtro di argomenti, ma come una stringa letterale in tutti gli altri contesti. Si consiglia di utilizzare questi caratteri jolly MQTT come parte delle policy solo dopo un'attenta valutazione. AWS IoT Core

Di seguito sono riportati esempi di elementi registrati e non registrati che utilizzano i caratteri jolly MQTT nelle politiche. AWS IoT Core Questi caratteri jolly vengono trattati come stringhe letterali.

Registered devices

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi con ClientID che corrisponde al nome di un elemento nel registro. La policy consente ai dispositivi di effettuare la sottoscrizione agli argomenti "department/+/employees" e "location/#". Poiché + e # sono considerati stringhe letterali nelle AWS IoT Core policy, i dispositivi possono sottoscrivere l'argomento «reparto/+/employees» ma non anche l'argomento "". department/engineering/employees". Similarly, devices can subscribe to the topic "location/#" but not to the topic "location/Seattle". However, once the device subscribes to the topic "department/+/employees", the policy will allow them to receive messages from the topic "department/engineering/employees". Similarly, once the device subscribes to the topic "location/#", they will receive messages from the topic "location/Seattle

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": "iot:Subscribe",
```

```
   "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/department/+/
employees"
  },
  {
   "Effect": "Allow",
   "Action": "iot:Subscribe",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/location/#"
  },
  {
   "Effect": "Allow",
   "Action": "iot:Receive",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
  }
 ]
}
```

Unregistered devices

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica consente ai dispositivi di connettersi utilizzando ClientID1, ClientID2 o ClientID3. La policy consente ai dispositivi di effettuare la sottoscrizione agli argomenti "department/+/employees" e "location/#". Poiché + e # sono considerati stringhe letterali nelle AWS IoT Core politiche, i dispositivi possono sottoscrivere l'argomento «reparto/+/dipendenti» ma non anche l'argomento "". department/engineering/employees". Similarly, devices can subscribe to the topic "location/#" but not "location/Seattle". However, once the device subscribes to the topic "department/+/employees", the policy will allow them to receive messages from the topic "department/engineering/employees". Similarly, once the device subscribes to the topic "location/#", they will receive messages from the topic "location/Seattle

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
```

```
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/
department/+/employees"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/location/
#"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
        }
    ]
}
```

WebSocket Politiche per HTTP e client

Quando ti connetti tramite HTTP o il WebSocket protocollo, ti autentichi con Signature Version 4 e Amazon Cognito. Idenitità di Amazon Cognito possono essere autenticate e non autenticate. Le identità autenticate appartengono agli utenti autenticati da qualsiasi provider di identità supportato. Solitamente le identità non autenticate appartengono in genere agli utenti guest che non sono in grado di effettuare l'autenticazione con un provider di identità. Amazon Cognito fornisce un identificatore e AWS credenziali univoci per supportare identità non autenticate. Per ulteriori informazioni, consulta the section called “Autorizzazione con identità Amazon Cognito”.

Per le seguenti operazioni, AWS IoT Core utilizza AWS IoT Core le policy collegate alle identità di Amazon Cognito tramite l'API. `AttachPolicy` In questo modo vengono ridotte le autorizzazioni associate al pool di identità di Amazon Cognito con identità autenticate.

- `iot:Connect`
- `iot:Publish`
- `iot:Subscribe`

- `iot:Receive`
- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

Ciò significa che un'identità di Amazon Cognito necessita dell'autorizzazione della policy del ruolo IAM e della AWS IoT Core policy. Puoi collegare la policy del ruolo IAM al pool e la AWS IoT Core policy all'identità di Amazon Cognito tramite l' AWS IoT Core `AttachPolicy`API.

Gli utenti autenticati e non autenticati sono diversi tipi di identità. Se non alleghi una AWS IoT policy all'identità di Amazon Cognito, un utente autenticato non ottiene l'autorizzazione AWS IoT e non ha accesso a AWS IoT risorse e azioni.

> **Note**
>
> Per altre AWS IoT Core operazioni o per identità non autenticate, AWS IoT Core non limita le autorizzazioni associate al ruolo del pool di identità di Amazon Cognito. Per le identità autenticate e non autenticate, questa è la policy più permissiva che consigliamo di collegare al ruolo del pool di Amazon Cognito.

HTTP

Per permettere alle identità non autenticate di Amazon Cognito di pubblicare messaggi tramite HTTP in un argomento specifico di Amazon Cognito Identity, collega la seguente policy IAM al ruolo del pool di identità di Amazon Cognito:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/${cognito-
identity.amazonaws.com:sub}"
```

```
            ]
        }
    ]
}
```

Per consentire l'autenticazione degli utenti, collega la policy precedente al ruolo del pool di Amazon Cognito Identity e all'Amazon Cognito Identity utilizzando l'API. AWS IoT Core AttachPolicy

> **Note**
>
> Quando autorizzi le identità di Amazon Cognito AWS IoT Core , considera entrambe le politiche e concede i privilegi minimi specificati. Un'operazione è consentita solo se entrambe le policy consentono l'operazione richiesta. Se una policy non consente un'operazione, quest'ultima non è autorizzata.

MQTT

Per consentire alle identità di Amazon Cognito non autenticate di pubblicare messaggi MQTT WebSocket su un argomento specifico di Amazon Cognito Identity nel tuo account, collega la seguente policy IAM al ruolo del pool di Amazon Cognito Identity:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-
identity.amazonaws.com:sub}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/${cognito-
identity.amazonaws.com:sub}"]
```

```
        }
    ]
}
```

Per consentire l'autenticazione degli utenti, collega la policy precedente al ruolo del pool di Amazon Cognito Identity e all'Amazon Cognito Identity utilizzando l'API. AWS IoT Core AttachPolicy

> **Note**
>
> Quando autorizzi le identità di Amazon Cognito AWS IoT Core , considera entrambe e concede i privilegi minimi specificati. Un'operazione è consentita solo se entrambe le policy consentono l'operazione richiesta. Se una policy non consente un'operazione, quest'ultima non è autorizzata.

Esempi di policy di connessione e pubblicazione

Per i dispositivi registrati come oggetti nel AWS IoT Core registro, la seguente politica concede l'autorizzazione a connettersi AWS IoT Core con un ID client che corrisponde al nome dell'oggetto e limita la pubblicazione del dispositivo su un argomento MQTT specifico dell'ID client o del nome dell'oggetto. Affinché una connessione abbia esito positivo, il nome dell'oggetto deve essere registrato nel AWS IoT Core registro ed essere autenticato utilizzando un'identità o un principale associato all'oggetto:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/${iot:Connection.Thing.ThingName}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
      ]
    }
  ]
}
```

Per i dispositivi non registrati come elementi nel AWS IoT Core registro, la seguente politica concede l'autorizzazione alla connessione AWS IoT Core con l'ID client `client1` e limita il dispositivo alla pubblicazione su un argomento MQTT specifico per ClientID:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/${iot:ClientId}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/client1"
      ]
    }
  ]
}
```

Esempi di policy per i messaggi conservati

L'utilizzo di messaggi conservati richiede politiche specifiche. I messaggi conservati sono messaggi MQTT pubblicati con il flag RETAIN impostato e archiviati da. AWS IoT Core Questa sezione presenta esempi di policy che consentono l'uso comune dei messaggi conservati.

In questa sezione:

- Policy per la connessione e la pubblicazione di messaggi conservati
- Criteri per connettere e pubblicare i messaggi Will conservati
- Policy per elencare e ricevere messaggi conservati

Policy per la connessione e la pubblicazione di messaggi conservati

Affinché un dispositivo pubblichi i messaggi conservati, il dispositivo deve essere in grado di connettersi, pubblicare (qualsiasi messaggio MQTT) e pubblicare messaggi conservati MQTT. Il seguente criterio concede queste autorizzazioni per l'argomento: `device/sample/configuration` al client **`device1`**. Per un altro esempio che concede l'autorizzazione per connettersi, consultathe section called “Esempi di policy di connessione e pubblicazione”.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/device1"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:RetainPublish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/device/sample/configuration"
```

```
    ]
   }
  ]
}
```

Criteri per connettere e pubblicare i messaggi Will conservati

I client possono configurare un messaggio che AWS IoT Core verrà pubblicato quando il client si disconnette inaspettatamente. MQTT chiama un messaggio di tipo messaggio Will. Un client deve disporre di una condizione aggiuntiva aggiunta all'autorizzazione di connessione per includerli.

Il seguente documento di policy concede a tutti i client l'autorizzazione per connettersi e pubblicare un messaggio Will, identificato dal relativo argomento, `will`, di cui AWS IoT Core lo manterrà.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/device1"
   ],
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "LastWill"
     ]
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:RetainPublish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/will"
```

```
    ]
   }
  ]
}
```

Policy per elencare e ricevere messaggi conservati

Servizi e applicazioni possono accedere ai messaggi conservati senza la necessità di supportare un client MQTT chiamando `ListRetainedMessages` e `GetRetainedMessage`. I servizi e le applicazioni che chiamano queste azioni devono essere autorizzati utilizzando una policy come l'esempio seguente.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:ListRetainedMessages"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/device1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetRetainedMessage"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo"
            ]
        }
    ]
}
```

Esempi di policy di certificato

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica concede l'autorizzazione a connettersi a AWS IoT Core un ID client che corrisponde al nome di un oggetto e a pubblicare

su un argomento il cui nome è uguale al certificato utilizzato dal `certificateId` dispositivo per autenticarsi:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:CertificateId}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        }
    ]
}
```

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica concede l'autorizzazione alla connessione AWS IoT Core con il client IDs `client3` e alla pubblicazione su un argomento il cui nome è uguale al `certificateId` certificato utilizzato dal dispositivo per l'autenticazione: `client1` `client2`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:CertificateId}"]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
}
```

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica concede l'autorizzazione a AWS IoT Core connettersi a un ID client che corrisponde al nome dell'oggetto e a pubblicare su un argomento il cui nome è uguale al `CommonName` campo dell'oggetto del certificato utilizzato dal dispositivo per l'autenticazione:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.CommonName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        }
    ]
```

```
}
```

> **Note**
>
> In questo esempio, il nome comune dell'oggetto del certificato viene utilizzato come identificatore dell'argomento, presumendo che il nome comune dell'oggetto sia univoco per ogni certificato registrato. Se i certificati sono condivisi tra più dispositivi, il nome comune dell'oggetto è identico per tutti i dispositivi che condividono il certificato, concedendo quindi privilegi di pubblicazione nello stesso argomento da più dispositivi (opzione non consigliata).

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica concede l'autorizzazione alla connessione AWS IoT Core con il client IDs `client3` e alla pubblicazione su un argomento il cui nome è uguale al `CommonName` campo dell'oggetto del certificato utilizzato dal dispositivo per l'autenticazione: `client1` `client2`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.CommonName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
```

```
}
```

> **Note**
>
> In questo esempio, il nome comune dell'oggetto del certificato viene utilizzato come identificatore dell'argomento, presumendo che il nome comune dell'oggetto sia univoco per ogni certificato registrato. Se i certificati sono condivisi tra più dispositivi, il nome comune dell'oggetto è identico per tutti i dispositivi che condividono il certificato, concedendo quindi privilegi di pubblicazione nello stesso argomento da più dispositivi (opzione non consigliata).

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica concede l'autorizzazione a connettersi a AWS IoT Core un ID client che corrisponde al nome dell'oggetto e a pubblicare su un argomento il cui nome ha il prefisso `admin/` quando il certificato utilizzato per l'autenticazione del dispositivo ha il `Subject.CommonName.2` campo impostato su`Administrator`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
                "StringEquals": {
                    "iot:Certificate.Subject.CommonName.2": "Administrator"
            }
        }
        }
```

```
    ]
}
```

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica concede l'autorizzazione alla connessione AWS IoT Core con il client IDs `client1` `client3` e alla pubblicazione su un argomento il cui nome è preceduto da `admin/` quando il certificato utilizzato per l'autenticazione del dispositivo ha il `Subject.CommonName.2` campo impostato su`Administrator`: `client2`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
                "StringEquals": {
                    "iot:Certificate.Subject.CommonName.2": "Administrator"
                }
        }
        }
    ]
}
```

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica consente a un dispositivo di utilizzare il proprio nome oggetto per pubblicare su un argomento specifico, `admin/` seguito

da `ThingName` quando il certificato utilizzato per l'autenticazione del dispositivo ha uno dei suoi `Subject.CommonName` campi impostato su`Administrator`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/
${iot:Connection.Thing.ThingName}"],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:Certificate.Subject.CommonName.List": "Administrator"
            }
        }
        }
    ]
}
```

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica concede l'autorizzazione AWS IoT Core alla connessione al client IDs `client1` `client3` e alla pubblicazione sull'argomento `admin` quando uno dei `Subject.CommonName` campi del certificato utilizzato per l'autenticazione del dispositivo è impostato su: `client2` `Administrator`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin"],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:Certificate.Subject.CommonName.List": "Administrator"
            }
        }
        }
    ]
}
```

Esempi di policy di oggetto

La seguente politica consente a un dispositivo di connettersi se il certificato utilizzato per l'autenticazione AWS IoT Core è allegato all'oggetto per cui viene valutata la politica:

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":["iot:Connect"],
            "Resource":[ "*" ],
            "Condition": {
                "Bool": {
                    "iot:Connection.Thing.IsAttached": ["true"]
            }
        }
        }
```

```
    ]
}
```

La seguente policy consente a un dispositivo di pubblicare se il certificato è collegato a un oggetto con un particolare tipo di oggetto e se l'oggetto ha un attributo di `attributeName` con valore `attributeValue`. Per ulteriori informazioni sulle variabili delle policy di oggetto, consulta Variabili delle policy di oggetto.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/device/stats",
      "Condition": {
        "StringEquals": {
          "iot:Connection.Thing.Attributes[attributeName]": "attributeValue",
          "iot:Connection.Thing.ThingTypeName": "Thing_Type_Name"
        },
        "Bool": {
          "iot:Connection.Thing.IsAttached": "true"
        }
      }
    }
  ]
}
```

La seguente policy consente a un dispositivo di pubblicare su un argomento che inizia con un attributo dell'oggetto. Se il certificato del dispositivo non è associato all'oggetto, questa variabile non verrà risolta e genererà un errore di accesso negato. Per ulteriori informazioni sulle variabili delle policy di oggetto, consulta Variabili delle policy di oggetto.

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.Attributes[attributeName]}/*"
    }
  ]
}
```

Esempi di policy di processo base

Questo esempio mostra le dichiarazioni delle policy richieste per un obiettivo di processo che è un singolo dispositivo per ricevere una richiesta di processo e comunicare lo stato di esecuzione del processo con AWS IoT.

Sostituisci *us-west-2:57EXAMPLE833* con i tuoi Regione AWS, un carattere con i due punti (:) e il tuo Account AWS numero a 12 cifre, quindi sostituiscilo *uniqueThingName* con il nome della risorsa oggetto che rappresenta il dispositivo in. AWS IoT

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:client/uniqueThingName"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/pubtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/job/*",
```

```
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/jobExecution/
*",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topicfilter/test/dc/subtopic",
                "arn:aws:iot:us-west-2:123456789012:topicfilter/$aws/events/
jobExecution/*",
                "arn:aws:iot:us-west-2:123456789012:topicfilter/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/subtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotjobsdata:DescribeJobExecution",
                "iotjobsdata:GetPendingJobExecutions",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:UpdateJobExecution"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName"
            ]
        }
    ]
```

```
}
```

Autorizzazione con identità Amazon Cognito

Esistono due tipi di identità di Amazon Cognito: non autenticate e autenticate. Se l'app supporta identità Amazon Cognito non autenticate, non viene eseguita alcuna autenticazione in modo da non sapere chi è l'utente.

Identità non autenticate: per le identità Amazon Cognito non autenticate, viene concessa l'autorizzazione collegando un ruolo IAM a un pool di identità non autenticato. Consigliamo di concedere l'accesso solo alle risorse che si desidera rendere disponibili agli utenti sconosciuti.

> **Important**
>
> Per gli utenti non autenticati che si connettono AWS IoT Core ad Amazon Cognito, consigliamo di concedere l'accesso a risorse molto limitate nelle policy IAM.

Identità autenticate: per le identità Amazon Cognito autenticate, è necessario specificare autorizzazioni in due posizioni:

- Collega una policy IAM al pool di identità di Amazon Cognito autenticato e
- Allega una AWS IoT Core policy all'identità di Amazon Cognito (utente autenticato).

Esempi di policy per utenti Amazon Cognito non autenticati e autenticati che si connettono a AWS IoT Core

Nell'esempio seguente vengono illustrate le autorizzazioni nella policy IAM e nella policy IoT di un'identità Amazon Cognito. L'utente autenticato desidera pubblicare su un argomento specifico del dispositivo (ad esempio). device/DEVICE_ID/status

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/Client_ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/device/Device_ID/status"
            ]
        }
    ]
}
```

L'esempio seguente mostra le autorizzazioni in una policy IAM di un ruolo non autenticato di Amazon Cognito. L'utente non autenticato desidera pubblicare in argomenti non specifici del dispositivo che non richiedono l'autenticazione.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:topic/non_device_specific_topic"
            ]
        }
    ]
}
```

GitHub esempi

I seguenti esempi di applicazioni web GitHub mostrano come incorporare l'allegato delle policy agli utenti autenticati nel processo di registrazione e autenticazione degli utenti.

- Applicazione web MQTT publish/subscribe React che utilizza e AWS AmplifySDK per dispositivi AWS IoT per JavaScript
- Applicazione web MQTT publish/subscribe React che utilizza AWS AmplifySDK per dispositivi AWS IoT per JavaScript, la e una funzione Lambda

Amplify è un insieme di strumenti e servizi che ti aiuta a creare applicazioni web e mobili che si integrano con i servizi. AWS Per ulteriori informazioni su Amplify, consulta Amplify Framework Documentation (Documentazione Amplify Framework).

Entrambi gli esempi eseguono la seguente procedura.

1. Quando un utente si registra per un account, l'applicazione crea un bacino d'utenza e un'identità di Amazon Cognito.
2. Quando un utente si autentica, l'applicazione crea e allega una policy all'identità. Ciò consente all'utente di pubblicare e sottoscrivere le autorizzazioni.
3. L'utente può utilizzare l'applicazione per pubblicare e iscriversi ad argomenti MQTT.

Il primo esempio utilizza l'operazione API `AttachPolicy` direttamente all'interno dell'operazione di autenticazione. Nell'esempio seguente viene illustrato come implementare questa chiamata API all'interno di un'applicazione Web React che utilizza Amplify e SDK per dispositivi AWS IoT per JavaScript.

```
function attachPolicy(id, policyName) {
    var Iot = new AWS.Iot({region: AWSConfiguration.region, apiVersion:
 AWSConfiguration.apiVersion, endpoint: AWSConfiguration.endpoint});
    var params = {policyName: policyName, target: id};
```

```
    console.log("Attach IoT Policy: " + policyName + " with cognito identity id: " +
 id);
    Iot.attachPolicy(params, function(err, data) {
         if (err) {
               if (err.code !== 'ResourceAlreadyExistsException') {
                  console.log(err);
               }
          }
         else  {
            console.log("Successfully attached policy with the identity", data);
         }
     });
}
```

Questo codice viene visualizzato nel AuthDisplayfile.js.

Il secondo esempio implementa l'operazione API `AttachPolicy` in una funzione Lambda. Nell'esempio seguente viene illustrato in che modo Lambda utilizza questa chiamata API.

```
iot.attachPolicy(params, function(err, data) {
     if (err) {
           if (err.code !== 'ResourceAlreadyExistsException') {
              console.log(err);
              res.json({error: err, url: req.url, body: req.body});
           }
      }
     else  {
        console.log(data);
        res.json({success: 'Create and attach policy call succeed!', url: req.url,
 body: req.body});
     }
 });
```

Questo codice viene visualizzato all'interno della funzione `iot.GetPolicy` nel fileapp.js.

> **Note**
>
> Quando chiami la funzione con AWS le credenziali ottenute tramite i pool di identità di Amazon Cognito, l'oggetto di contesto nella funzione Lambda contiene un valore per.

`context.cognito_identity_id` Per ulteriori informazioni, consulta gli argomenti seguenti.

- AWS Lambda oggetto di contesto in Node.js
- AWS Lambda oggetto di contesto in Python
- AWS Lambda oggetto di contesto in Ruby
- AWS Lambda oggetto di contesto in Java
- AWS Lambda oggetto contestuale in Go
- AWS Lambda oggetto di contesto in C#
- AWS Lambda oggetto di contesto in PowerShell

Autorizzazione di chiamate dirette ai AWS servizi utilizzando il provider di AWS IoT Core credenziali

I dispositivi possono utilizzare i certificati X.509 a cui connettersi AWS IoT Core utilizzando i protocolli di autenticazione reciproca TLS. Altri AWS servizi non supportano l'autenticazione basata su certificati, ma possono essere richiamati utilizzando AWS credenziali in formato Signature Version 4.AWS L'algoritmo Signature Version 4 richiede normalmente che il chiamante disponga di un ID di chiave di accesso e di una chiave di accesso segreta. AWS IoT Core dispone di un provider di credenziali che consente di utilizzare il certificato X.509 integrato come identità univoca del dispositivo per autenticare le richieste. AWS Ciò elimina la necessità di archiviare un ID chiave di accesso e la chiave di accesso segreta sul dispositivo.

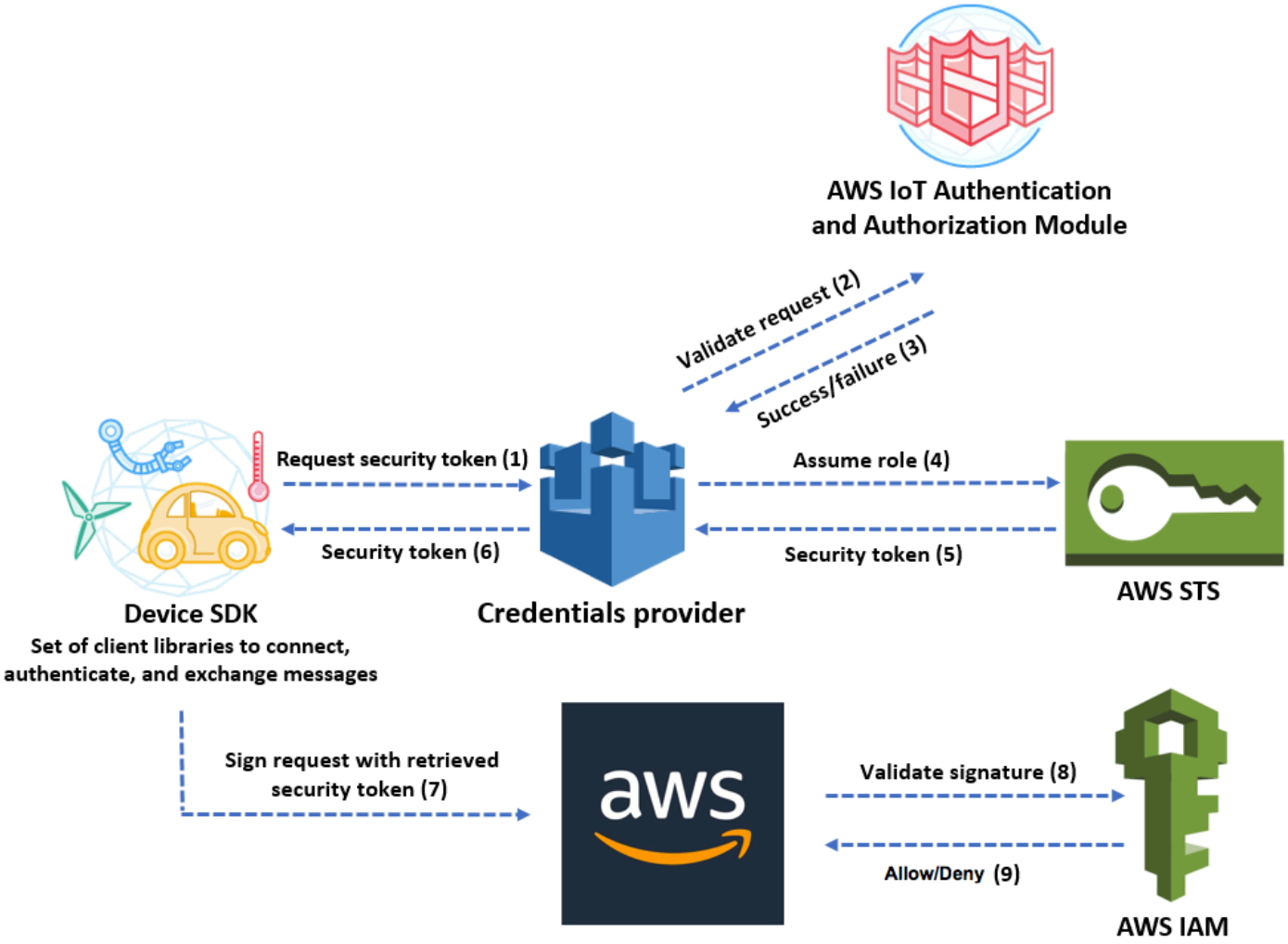
Il fornitore di credenziali autentica un intermediario usando un certificato X.509 ed emette un token temporaneo di sicurezza con privilegi limitati. Il token può essere utilizzato per firmare e autenticare qualsiasi richiesta. AWS Questo metodo di autenticazione delle AWS richieste richiede la creazione e la configurazione di un ruolo AWS Identity and Access Management (IAM) e l'associazione di politiche IAM appropriate al ruolo in modo che il fornitore delle credenziali possa assumere il ruolo per conto dell'utente. Per ulteriori informazioni su AWS IoT Core e IAM, consulta Gestione delle identità e degli accessi per AWS IoT.

AWS IoT richiede che i dispositivi inviino l'estensione SNI (Server Name Indication) al protocollo Transport Layer Security (TLS) e forniscano l'indirizzo completo dell'endpoint sul campo. `host_name` Il campo `host_name` deve contenere l'endpoint che si sta chiamando e deve essere:

- Il `endpointAddress` restituito da `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`.

Le connessioni tentate dai dispositivi senza il valore `host_name` corretto non andranno a buon fine.

Lo schema seguente illustra il flusso di lavoro del fornitore di credenziali.



1. Il AWS IoT Core dispositivo invia una richiesta HTTPS al provider di credenziali per un token di sicurezza. La richiesta include il certificato X.509 del dispositivo per l'autenticazione.
2. Il provider di credenziali inoltra la richiesta al modulo di AWS IoT Core autenticazione e autorizzazione per convalidare il certificato e verificare che il dispositivo sia autorizzato a richiedere il token di sicurezza.

3. Se il certificato è valido e dispone dell'autorizzazione per richiedere un token di sicurezza, il modulo di AWS IoT Core autenticazione e autorizzazione restituisce l'esito positivo. In caso contrario, invia un'eccezione al dispositivo.
4. Una volta completata la convalida del certificato, il fornitore di credenziali richiama il AWS Security Token Service (AWS STS) per usare il ruolo IAM creato.
5. AWS STS restituisce un token di sicurezza temporaneo con privilegi limitati al fornitore delle credenziali.
6. Il fornitore di credenziali restituisce il token di sicurezza al dispositivo.
7. Il dispositivo utilizza il token di sicurezza per firmare una AWS richiesta con AWS Signature Version 4.
8. Il servizio richiesto richiama IAM per convalidare la firma e autorizzare la richiesta in base alle policy di accesso collegate al ruolo IAM creato per il fornitore di credenziali.
9. Se IAM convalida la firma correttamente e autorizza la richiesta, questa va a buon fine. In caso contrario, IAM invia un'eccezione.

La sezione seguente descrive come utilizzare un certificato per ottenere un token di sicurezza. Si basa su presupposto che tu abbia già registrato un dispositivo e creato e attivato il relativo certificato.

Come utilizzare un certificato per ottenere un token di sicurezza

1. Configura il ruolo IAM che il fornitore di credenziali assume per conto del tuo dispositivo. Collega la seguente policy di attendibilità al ruolo.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {"Service": "credentials.iot.amazonaws.com"},
        "Action": "sts:AssumeRole"
    }
}
```

   Per ogni AWS servizio che desideri chiamare, allega una politica di accesso al ruolo. Il fornitore di credenziali supporta le seguenti variabili di policy:

   - `credentials-iot:ThingName`

- `credentials-iot:ThingTypeName`
- `credentials-iot:AwsCertificateId`

Quando il dispositivo fornisce il nome dell'oggetto nella richiesta di un servizio AWS , il fornitore di credenziali aggiunge `credentials-iot:ThingName` e `credentials-iot:ThingTypeName` come variabili di contesto al token di sicurezza. Il fornitore di credenziali fornisce `credentials-iot:AwsCertificateId` come variabile di contesto, anche se il dispositivo non fornisce il nome dell'oggetto nella richiesta. Il nome dell'oggetto viene passato come valore dell'intestazione di richiesta HTTP `x-amzn-iot-thingname`.

Queste tre variabili funzionano solo con le policy IAM, non con le policy AWS IoT Core .

2. Assicurati che l'utente che esegue la fase successiva (creando un alias del ruolo) disponga dell'autorizzazione per trasferire questo ruolo appena creato a AWS IoT Core. La seguente politica fornisce entrambe `iam:GetRole` le `iam:PassRole` autorizzazioni a un AWS utente. L'autorizzazione `iam:GetRole` consente all'utente di ottenere informazioni sul ruolo appena creato. L'`iam:PassRole`autorizzazione consente all'utente di passare il ruolo a un altro AWS servizio.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GetRole",
            "iam:PassRole"
        ],
        "Resource": "arn:aws:iam::123456789012:role/your role name"
    }
}
```

3. Crea un alias di AWS IoT Core ruolo. Il dispositivo che effettuerà chiamate dirette ai AWS servizi deve sapere a quale ruolo ARN utilizzare durante la connessione. AWS IoT Core L'impostazione hardcoded dell'ARN del ruolo non è una buona soluzione, perché richiede di aggiornare il dispositivo ogni volta che l'ARN del ruolo cambia. Una soluzione migliore consiste nell'utilizzare l'API `CreateRoleAlias` per creare un alias del ruolo che punti all'ARN del ruolo. Se l'ARN del ruolo viene modificato, è sufficiente aggiornare l'alias del ruolo. Non è richiesta alcuna modifica sul dispositivo. Questa API accetta i parametri seguenti:

`roleAlias`

Obbligatorio. Stringa arbitraria che identifica l'alias del ruolo. Opera come chiave primaria nel modello di dati dell'alias del ruolo. Contiene da 1 a 128 caratteri e deve includere solo caratteri alfanumerici e i simboli =, @ e -. Sono consentite le lettere maiuscole e minuscole. Gli alias dei ruoli fanno distinzione tra maiuscole e minuscole.

`roleArn`

Obbligatorio. ARN del ruolo cui fa riferimento l'alias del ruolo.

`credentialDurationSeconds`

Opzionale. Tempo di validità (in secondi) della credenziale. Il valore minimo è di 900 secondi (15 minuti). Il valore massimo è di 43.200 secondi (12 ore). Il valore predefinito è 3.600 secondi (1 ora).

> **Important**
>
> Il AWS IoT Core Credential Provider può emettere una credenziale con una durata massima di 43.200 secondi (12 ore). La validità della credenziale fino a 12 ore può aiutare a ridurre il numero di chiamate verso il provider di credenziali memorizzando nella cache più a lungo la credenziale.
> Il valore `credentialDurationSeconds` deve essere minore di o uguale alla durata massima della sessione del ruolo IAM a cui fa riferimento l'alias del ruolo. Per ulteriori informazioni, vedere Modifica della durata massima della sessione (AWS API) di un ruolo nella AWS Identity and Access Management User Guide.

Per ulteriori informazioni sull'API, consulta CreateRoleAlias.

4. Collega una policy al certificato del dispositivo. La policy collegata al certificato del dispositivo deve concedere al dispositivo l'autorizzazione necessaria per assumere il ruolo. A questo scopo, devi concedere l'autorizzazione per l'operazione `iot:AssumeRoleWithCertificate` sull'alias del ruolo, come indicato nel seguente esempio.

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "iot:AssumeRoleWithCertificate",
            "Resource": "arn:aws:iot:us-east-1:123456789012:rolealias/your role
 alias"
        }
    ]
}
```

5. Effettuare una richiesta HTTPS al fornitore di credenziali per ottenere un token di sicurezza. Fornire le seguenti informazioni:

   - Certificato: trattandosi di una richiesta HTTP tramite autenticazione reciproca TLS, fornisci il certificato e la chiave privata per il client durante la richiesta. Usa lo stesso certificato e la stessa chiave privata che hai usato quando hai registrato il certificato con AWS IoT Core.

     Per assicurarti che il tuo dispositivo stia comunicando con AWS IoT Core (e non un servizio che lo impersona), consulta Autenticazione del server, segui i collegamenti per scaricare i certificati CA appropriati, quindi copiali sul tuo dispositivo.
   - RoleAlias: il nome dell'alias di ruolo che hai creato per il provider di credenziali. I nomi degli alias dei ruoli fanno distinzione tra maiuscole e minuscole e devono corrispondere all'alias di ruolo creato in. AWS IoT Core
   - ThingName: Il nome dell'oggetto che hai creato quando l'hai AWS IoT Core registrato. Viene trasferito come valore dell'intestazione HTTP `x-amzn-iot-thingname`. Questo valore è richiesto solo se si utilizzano gli attributi degli oggetti come variabili di policy nelle AWS IoT Core o nelle policy IAM.

     > **Note**
     >
     > L'ThingNameinput fornito `x-amzn-iot-thingname` deve corrispondere al nome della risorsa AWS IoT Thing assegnata a un certificato. Se non corrisponde, viene restituito un errore 403.

   Esegui il comando seguente in AWS CLI per ottenere l'endpoint del provider di credenziali per il tuo. Account AWS Per ulteriori informazioni sull'API, consulta DescribeEndpoint. Per gli endpoint compatibili con FIPS, vedere. AWS IoT Core- endpoint del fornitore di credenziali

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

Il seguente oggetto JSON è l'output di esempio del comando describe-endpoint. Contiene il valore `endpointAddress` utilizzato per richiedere un token di sicurezza.

```
{
    "endpointAddress": "your_aws_account_specific_prefix.credentials.iot.your
 region.amazonaws.com"
}
```

Utilizza l'endpoint per effettuare una richiesta HTTPS al fornitore di credenziali per restituire un token di sicurezza. Il comando di esempio seguente utilizza `curl`, ma è possibile utilizzare qualsiasi client HTTP.

> **Note**
>
> Il nome RoleAlias fa distinzione tra maiuscole e minuscole e deve corrispondere all'alias del ruolo creato in. AWS IoT

```
curl --cert your certificate --key your private key -H "x-amzn-iot-thingname: your
 thing name" --cacert AmazonRootCA1.pem https://your endpoint /role-aliases/your
 role alias/credentials
```

Questo comando restituisce un oggetto token di sicurezza che contiene `accessKeyId`, `secretAccessKey`, `sessionToken` e una scadenza. Il seguente oggetto JSON è l'output di esempio del comando `curl`.

```
   {"credentials":{"accessKeyId":"access key","secretAccessKey":"secret access
 key","sessionToken":"session token","expiration":"2018-01-18T09:18:06Z"}}
```

È quindi possibile utilizzare i `sessionToken` valori `accessKeyIdsecretAccessKey`, e per firmare le richieste ai servizi. AWS Per una end-to-end dimostrazione, vedi Come eliminare la necessità di AWS credenziali codificate nei dispositivi utilizzando il post di blog AWS IoT Credential Provider sul Security Blog.AWS

Accesso tra account con IAM

AWS IoT Core consente di consentire a un principale di pubblicare o sottoscrivere un argomento definito in un Account AWS argomento che non è di proprietà del principale. Per configurare l'accesso tra account, devi creare una policy IAM; e un ruolo IAM; e quindi collegare la policy al ruolo.

Innanzitutto, crea una policy IAM gestita del cliente come descritto in Creazione di policy IAM, come faresti per altri utenti e certificati nell'account Account AWS.

Per i dispositivi registrati nel AWS IoT Core registro, la seguente politica concede l'autorizzazione ai dispositivi a cui si connette AWS IoT Core utilizzando un ID client che corrisponde al nome dell'oggetto del dispositivo e a pubblicare con il nome dell'oggetto `my/topic/`*`thing-name`* dove *`thing-name`* è il nome dell'oggetto del dispositivo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/my/topic/
${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

Per i dispositivi non registrati nel AWS IoT Core registro, la seguente politica concede l'autorizzazione a un dispositivo a utilizzare il nome oggetto `client1` registrato nel AWS IoT Core registro dell'account (123456789012) per connettersi AWS IoT Core e pubblicare su un argomento specifico dell'ID client il cui nome è preceduto da: `my/topic/`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
            ]
        }
    ]
}
```

Quindi, segui le fasi nella pagina Creazione di un ruolo per delegare le autorizzazioni a un utente IAM. Immetti l'ID dell'account Account AWS con cui desideri condividere l'accesso. Quindi, nella fase finale, collega la policy appena creata al ruolo. Se in un secondo momento devi modificare l'ID account AWS cui desideri concedere l'accesso, potrai usare il seguente formato di policy di trust:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::567890123456:user/MyUser"
        },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Protezione dei dati in AWS IoT Core

Il modello di responsabilità AWS condivisa modello di di si applica alla protezione dei dati in AWS IoT Core. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per maggiori informazioni sulla privacy dei dati, consulta le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al AWS Modello di responsabilità condivisa e GDPR nel AWS Blog sulla sicurezza.

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori AWS IoT o Servizi AWS utilizzi la console, l'API o. AWS CLI AWS SDKs I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando si fornisce un URL a un server esterno, suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la richiesta al server.

Per ulteriori informazioni sulla protezione dei dati, consulta il post del blog AWS Modello di responsabilità condivisa e GDPR su AWS Security Blog.

AWS IoT i dispositivi raccolgono dati, eseguono alcune manipolazioni su tali dati e quindi li inviano a un altro servizio Web. Potresti scegliere di archiviare alcuni dati sul tuo dispositivo per un breve periodo di tempo. L'utente è responsabile di fornire qualsiasi protezione dei dati su tali dati a riposo. Quando il dispositivo invia dati a AWS IoT, lo fa tramite una connessione TLS, come illustrato più avanti in questa sezione. AWS IoT i dispositivi possono inviare dati a qualsiasi AWS servizio. Per ulteriori informazioni sulla sicurezza dei dati di ciascun servizio, consulta la documentazione relativa al servizio. AWS IoT può essere configurato per scrivere log su CloudWatch Logs e registrare chiamate AWS IoT API su. AWS CloudTrail Per ulteriori informazioni sulla sicurezza dei dati per questi servizi, consulta Authentication and Access Control for Amazon CloudWatch e Encrypting CloudTrail Log Files with AWS KMS managed keys.

Crittografia dei dati in AWS IoT

Per impostazione predefinita, tutti i AWS IoT dati in transito e a riposo sono crittografati. I dati in transito vengono crittografati tramite TLS e i dati inattivi vengono crittografati utilizzando chiavi AWS proprietarie. AWS IoT supporta la gestione dei clienti AWS KMS keys (chiavi KMS) da AWS Key Management Service (AWS KMS). Tuttavia, Device Advisor e AWS IoT Wireless utilizzano solo un Chiave di proprietà di AWS per crittografare i dati dei clienti.

Sicurezza dei trasporti in AWS IoT Core

TLS (Transport Layer Security) è un protocollo di crittografia progettato per comunicazioni sicure su una rete di computer. Il AWS IoT Core Device Gateway richiede ai clienti di crittografare tutte le comunicazioni in transito utilizzando TLS per le connessioni dai dispositivi al Gateway. TLS viene utilizzato per garantire la riservatezza dei protocolli applicativi (MQTT, HTTP e) supportati da. WebSocket AWS IoT Core Il supporto TLS è disponibile in diversi linguaggi di programmazione e

sistemi operativi. I dati all'interno AWS sono crittografati dal servizio specifico AWS . Per ulteriori informazioni sulla crittografia dei dati su altri AWS servizi, consulta la documentazione sulla sicurezza relativa a tale servizio.

Indice

- Protocolli TLS
- Policy di sicurezza
- Note importanti per la sicurezza dei trasporti in AWS IoT Core
- Sicurezza del trasporto per dispositivi wireless LoRa WAN

Protocolli TLS

AWS IoT Core supporta le seguenti versioni del protocollo TLS:

- TLS 1.3
- TLS 1.2

Con AWS IoT Core, puoi configurare le impostazioni TLS (per TLS 1.2 e TLS 1.3) nelle configurazioni di dominio. Per ulteriori informazioni, consulta ???.

Policy di sicurezza

Una policy di sicurezza è una combinazione di protocolli TLS e delle relative crittografie che determinano quali protocolli e crittografie sono supportati durante le negoziazioni TLS tra un client e un server. È possibile configurare i dispositivi per utilizzare policy di sicurezza predefinite in base alle esigenze. Tieni presente che AWS IoT Core non supporta politiche di sicurezza personalizzate.

Puoi scegliere una delle politiche di sicurezza predefinite per i tuoi dispositivi al momento della connessione a AWS IoT Core. I nomi delle politiche di sicurezza predefinite più recenti AWS IoT Core includono informazioni sulla versione basate sull'anno e sul mese in cui sono state rilasciate. La policy di sicurezza predefinita di default è `IoTSecurityPolicy_TLS13_1_2_2022_10`. Per specificare una politica di sicurezza, è possibile utilizzare la AWS IoT console o il AWS CLI. Per ulteriori informazioni, consulta ???.

Nella seguente tabella vengono descritte le policy di sicurezza predefinite più recenti supportate da AWS IoT Core . `IotSecurityPolicy_` è stato rimosso dai nomi di policy nella riga dell'intestazione ai fini dell'adattamento.

Policy di sicurezza	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
Porta TCP	443/8443/8883	443/8443/8883	443/8443/8883	443	8443/8883	443	8443/8883
Protocolli TLS							
TLS 1.2		✓	✓	✓	✓	✓	✓
TLS 1.3	✓	✓					
Crittografie TLS							
TLS_AES_128_GCM_SHA256	✓	✓					
TLS_AES_256_GCM_SHA384	✓	✓					
TLS_CHACHA20_POLY1305_SHA256	✓	✓					
ECDHE-RSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
ECDH-RSA-AES128-SHA256		✓	✓	✓	✓	✓	✓

Policy di sicurezza	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
ECDHE-RSA-AES128 - SHA		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256 -GCM-SHA384		✓	✓	✓	✓	✓	✓
ECDH-RSA-AES256 - SHA384		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256 - SHA		✓	✓	✓	✓	✓	✓
AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
AES128-SHA256		✓	✓	✓		✓	✓
AES128-SHA		✓	✓	✓	✓	✓	✓
AES256-GCM-SHA384		✓	✓	✓	✓	✓	✓

Policy di sicurezza	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
AES256-SHA256		✓	✓	✓	✓	✓	✓
AES256-SHA		✓	✓	✓	✓	✓	✓
SHE-RSA--SHA AES256						✓	✓
ECDHE-ECDSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
ECDHE-ECSA- -AES128 SHA256		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA--SHA AES128		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA--GCM-AES256 SHA384		✓	✓	✓	✓	✓	✓

Policy di sicurezza	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
ECDHE-ECSA- -AES256 SHA384		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA- -SHA AES256		✓	✓	✓	✓	✓	✓

> **Note**
>
> `TLS12_1_0_2016_01`è disponibile solo nelle seguenti versioni Regioni AWS: ap-east-1, ap-northeast-2, ap-south-1, ap-southeast-2, ca-central-1, cn-north-1, cn-northwest-1, eu-north-1, eu-west-2, eu-west-3, me-3 -sud-1, sa-est-1, us-est-2, -1, -2, us-west-1. us-gov-west us-gov-west
> `TLS12_1_0_2015_01`è disponibile solo nelle seguenti versioni Regioni AWS: ap-northeast-1, ap-southeast-1, eu-central-1, eu-west-1, us-east-1, us-west-2.

Note importanti per la sicurezza dei trasporti in AWS IoT Core

Per i dispositivi che si connettono AWS IoT Core tramite MQTT, TLS crittografa la connessione tra i dispositivi e il broker e AWS IoT Core utilizza l'autenticazione del client TLS per identificare i dispositivi. Per ulteriori informazioni, consultare Autenticazione client. Per i dispositivi che si connettono AWS IoT Core tramite HTTP, TLS crittografa la connessione tra i dispositivi e il broker e l'autenticazione è delegata alla versione 4 di Signature. AWS Per ulteriori informazioni, consultare Firma delle richieste con Signature Version 4 nella documentazione generale di riferimento di AWS .

Quando si connettono dispositivi a AWS IoT Core, l'invio dell'estensione Server Name Indication (SNI) non è necessario, ma è altamente consigliato. Per utilizzare funzionalità come la registrazione di più account, i domini personalizzati, gli endpoint VPC e le politiche TLS configurate, è necessario utilizzare l'estensione SNI e fornire l'indirizzo completo dell'endpoint sul campo. `host_name` Il campo

`host_name` deve contenere l'endpoint che si sta chiamando. Tale endpoint deve essere uno dei seguenti:

- Il valore `endpointAddress` restituito da `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
- Il valore `domainName` restituito da `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"`

I tentativi di connessione effettuati da dispositivi con un valore errato o non valido falliranno. `host_name` AWS IoT Core registrerà gli errori utilizzando il tipo CloudWatch di autenticazione Custom Authentication.

AWS IoT Core non supporta l'estensione SessionTicket TLS.

Sicurezza del trasporto per dispositivi wireless LoRa WAN

LoRaI dispositivi WAN seguono le pratiche di sicurezza descritte in LoRaWAN™ SECURITY: A White Paper Prepared for the LoRa Alliance™ di Gemalto, Actility e Semtech.

Per ulteriori informazioni sulla sicurezza del trasporto con i dispositivi LoRa WAN, consulta Sicurezza dei dati e del trasporto LoRa WAN.

Crittografia dei dati in AWS IoT

La protezione dei dati si riferisce alla protezione dei dati durante il transito (mentre viaggiano da e verso AWS IoT Core) e quando sono inattivi (mentre sono archiviati su dispositivi o da altri AWS servizi). Tutti i dati inviati AWS IoT Core vengono inviati tramite una connessione TLS utilizzando MQTT, HTTPS e WebSocket protocolli, rendendoli sicuri per impostazione predefinita durante il transito. AWS IoT Core raccoglie i dati dai dispositivi e li invia ad altri AWS servizi per un'ulteriore elaborazione. Per ulteriori informazioni sulla crittografia dei dati su altri servizi AWS , consultare la documentazione di sicurezza per tale servizio. Per ulteriori informazioni, consulta Data encryption at rest.

FreeRTOS fornisce una libreria PKCS #11 che riassume l'archiviazione delle chiavi, accedendo agli oggetti crittografici e gestendo le sessioni. È responsabilità dell'utente utilizzare questa libreria per crittografare i dati inattivi sui dispositivi. Per ulteriori informazioni, consulta FreeRTOS Libreria Public Key Cryptography Standard (PKCS) #11.

Crittografia dei dati a riposo in AWS IoT Core

Per impostazione predefinita, tutti i AWS IoT Core dati inattivi vengono crittografati utilizzando chiavi AWS proprietarie. AWS IoT Core supporta anche chiavi simmetriche gestite dai clienti di AWS Key Management Service ()AWS KMS. Con le chiavi gestite dai clienti, puoi creare, possedere e gestire le AWS KMS chiavi del tuo AWS account. AWS IoT Core utilizzerà le tue chiavi KMS per crittografare i tuoi dati inattivi. Hai il pieno controllo su queste chiavi KMS, inclusa la creazione e la manutenzione delle relative politiche chiave. Puoi anche configurare le politiche IAM per i ruoli che accedono per AWS KMS controllare le autorizzazioni relative a queste chiavi.

AWS chiavi possedute

AWS le chiavi di proprietà sono una raccolta di chiavi KMS possedute e gestite da un AWS servizio per l'utilizzo in più AWS account. AWS i servizi possono utilizzare chiavi AWS di proprietà per proteggere i dati. Per impostazione predefinita, AWS IoT Core crittografa i dati inattivi utilizzando chiavi AWS proprietarie. Queste chiavi sono gestite dal servizio. Non è possibile visualizzare, gestire o utilizzare chiavi AWS di proprietà. Tuttavia, non è necessario intraprendere alcuna azione per proteggere queste chiavi.

Per ulteriori informazioni sulle chiavi AWS possedute, consulta le chiavi AWS possedute nella Guida per gli AWS Key Management Service sviluppatori.

Chiavi gestite dal cliente

Le chiavi gestite dal cliente sono chiavi KMS del tuo AWS account che crei, possiedi e gestisci. Hai il pieno controllo su queste AWS KMS chiavi, inclusa la creazione e la manutenzione delle relative politiche chiave. Puoi anche configurare le policy IAM per i ruoli che accedono per AWS KMS controllare le autorizzazioni relative a queste chiavi. Puoi configurare l'utilizzo AWS IoT Core di chiavi KMS gestite dal cliente per crittografare i tuoi dati.

Per ulteriori informazioni sulle chiavi gestite dal cliente, consulta Chiavi gestite dal cliente nella Guida per gli sviluppatori di AWS Key Management Service .

Per attivare le chiavi gestite dal cliente AWS IoT Core, procedi nel seguente modo:

Argomenti

- Fase 1: creare una chiave gestita dal cliente
- Passaggio 2: crea un ruolo IAM per concedere AWS IoT Core le autorizzazioni all'uso della chiave KMS
- Passaggio 3: Attiva le chiavi gestite dal cliente AWS IoT Core

- Fase 4: Sono necessarie autorizzazioni aggiuntive per le operazioni AWS IoT Core del piano di controllo
- Fase 5: Gestione delle chiavi
- Fase 6: Monitoraggio dello stato delle chiavi

Fase 1: creare una chiave gestita dal cliente

È possibile creare una chiave simmetrica gestita dal cliente utilizzando la AWS KMS console o i comandi CLI AWS KMS . Il `keySpec` deve essere `SYMMETRIC_DEFAULT` e il `keyUsage` deve essere. `ENCRYPT_DECRYPT`

> **Note**
>
> AWS IoT Core supporta solo AWS KMS chiavi con specifiche `SYMMETRIC_DEFAULT` chiave e `ENCRYPT_DECRYPT` utilizzo delle chiavi per le chiavi gestite dal cliente.

Di seguito è riportato un AWS CLI comando di esempio per creare una chiave KMS che può essere utilizzata con AWS IoT Core le chiavi gestite dai clienti.

```
aws kms create-key --key-spec SYMMETRIC_DEFAULT --key-usage ENCRYPT_DECRYPT --region
 us-west-2
```

Di seguito è riportato un esempio di output del comando.

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2024-09-19T11:45:23.982000-07:00",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
```

```
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
}
```

Per ulteriori informazioni, consulta Creazione di una chiave simmetrica gestita dal cliente nella Guida per gli AWS Key Management Service sviluppatori.

Policy della chiave

Quando si crea una chiave gestita dal cliente, è possibile specificare una politica chiave. Le policy della chiave controllano l'accesso alla chiave gestita dal cliente. Ogni chiave gestita dal cliente deve avere esattamente una policy della chiave, che contiene istruzioni che determinano chi può usare la chiave e come la possono usare. Per ulteriori informazioni, consulta le politiche chiave nella Guida per gli AWS Key Management Service sviluppatori.

AWS IoT Core utilizza un ruolo IAM nel tuo account per accedere alla tua chiave gestita dal cliente. Se utilizzi una policy di chiave personalizzata, assicurati che il ruolo IAM creato su questa chiave disponga delle seguenti autorizzazioni:

- `kms:DescribeKey`
- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:ReEncryptTo`
- `kms:ReEncryptFrom`

Passaggio 2: crea un ruolo IAM per concedere AWS IoT Core le autorizzazioni all'uso della chiave KMS

AWS IoT Core Per utilizzare la chiave KMS che hai creato per crittografare i tuoi dati inattivi, devi anche creare un ruolo IAM nel tuo account, che AWS IoT Core può presupporre di accedere alla chiave KMS.

Il ruolo deve avere la seguente politica di fiducia per consentire AWS IoT Core l'assunzione del ruolo.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "iot.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111122223333"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:iot:us-west-2:111122223333:*"
            }
        }
    }
}
```

Assicurati che le policy IAM allegate al ruolo IAM abbiano le seguenti autorizzazioni sulla chiave KMS:

- `kms:DescribeKey`
- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:ReEncryptTo`
- `kms:ReEncryptFrom`

Di seguito è riportato un esempio di policy IAM con le autorizzazioni richieste per le chiavi gestite dai clienti.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIoTToAccessKMSResource",
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:Decrypt",
```

```
                "kms:Encrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": [
                "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:aws-crypto-ec:vendor": "iot.amazonaws.com"
                }
            }
        }
    ]
}
```

Per ulteriori informazioni, consulta Creare un ruolo per delegare le autorizzazioni a un utente IAM nella Guida per l'utente.AWS Identity and Access Management

Passaggio 3: Attiva le chiavi gestite dal cliente AWS IoT Core

Dopo aver completato tutti i passaggi precedenti, esegui il comando `update-encryption-configuration` CLI per attivare l'utilizzo delle chiavi gestite dal cliente. AWS IoT Core Quando attivi le chiavi gestite dal cliente, tutte le AWS IoT Core risorse del tuo AWS account verranno crittografate utilizzando la AWS KMS chiave specificata.

1. Per attivare l' AWS IoT Core utilizzo delle chiavi gestite dal cliente AWS CLI, esegui il comando `update-encryption-configuration` CLI.

   ```
   aws iot update-encryption-configuration --encryption-type "CUSTOMER_MANAGED_KMS_KEY"
    \
   --kms-access-role-arn "arn:aws:iam::111122223333:role/myrole" \
   --kms-key-arn "arn:aws:kms:us-
   west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" --region us-west-2
   ```

2. Per verificare l' AWS IoT Core utilizzo delle chiavi gestite dal cliente AWS CLI, esegui il comando `describe-encryption-configuration` CLI:

   ```
   aws iot describe-encryption-configuration --region us-west-2
   ```

Se hai abilitato le chiavi gestite dal cliente AWS IoT Core, l'output può essere simile al seguente:

```
{
    "encryptionType": "CUSTOMER_MANAGED_KMS_KEY",
    "kmsKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "kmsAccessRoleArn": "arn:aws:iam::111122223333:role/myrole",
    "configurationDetails": {
        "configurationStatus": "HEALTHY"
    },
    "lastModifiedDate": "2024-09-26T22:01:02.365000-07:00"
}
```

Il `lastModifiedDate` campo indica la data dell'ultimo aggiornamento della configurazione di crittografia.

Se non hai abilitato le chiavi gestite dal cliente, l'output può essere simile al seguente:

```
{
    "encryptionType": "AWS_OWNED_KMS_KEY",
    "lastModifiedDate": "2024-09-26T22:01:02.365000-07:00"
}
```

Fase 4: Sono necessarie autorizzazioni aggiuntive per le operazioni AWS IoT Core del piano di controllo

Dopo aver attivato le chiavi gestite dal cliente, tutte le AWS IoT Core risorse appartenenti al tuo AWS account vengono crittografate con la chiave KMS fornita. Tutte le operazioni del piano di controllo ora richiedono che il chiamante `kms:Decrypt` disponga delle autorizzazioni sulla chiave KMS oltre alle autorizzazioni richieste per l'operazione specifica sulla risorsa. AWS IoT Core Se il chiamante non dispone dell'`kms:Decrypt`autorizzazione ed effettua una chiamata API che richiede la crittografia o la decrittografia dei dati (ad esempio,), `GetPolicy` riceverà un. `UnauthorizedException`

Ad esempio, quando si chiama`GetPolicy`, sono necessari entrambi `iot:GetPolicy` e le `kms:Decrypt` autorizzazioni sulla chiave KMS gestita dal cliente affinché la chiamata API abbia esito positivo.

> **Note**
>
> Quando aggiorni utenti o ruoli IAM per concedere AWS KMS le autorizzazioni sulla chiave utilizzata per la configurazione di crittografia, assicurati che la policy delle chiavi KMS conceda anche le autorizzazioni richieste ai rispettivi utenti o ruoli IAM.

AWS KMS autorizzazioni per **`UpdateEncryptionConfiguration`**

La chiamata `UpdateEncryptionConfiguration` API richiede le seguenti AWS KMS autorizzazioni sulla chiave KMS per poter attivare le chiavi gestite dal cliente o modificare la configurazione delle chiavi:

- `kms:DescribeKey`
- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:ReEncryptTo`
- `kms:ReEncryptFrom`

AWS KMS autorizzazioni per tutti gli altri piani di controllo APIs

La maggior parte dei piani di controllo APIs richiede `kms:Decrypt` le autorizzazioni quando le chiavi gestite dal cliente sono abilitate. Tuttavia, alcuni APIs non richiedono queste autorizzazioni aggiuntive:

APIs che non richiedono AWS KMS autorizzazioni

Le `List*` mani `Delete*` APIs non cadono in questo secchio. I clienti possono sempre richiamare qualsiasi API `List*` o `Delete*` Control Plane e tali chiamate API avrebbero esito positivo anche se il chiamante non dispone dell'autorizzazione. `kms:Decrypt` Queste chiamate API avranno esito positivo anche se la chiave gestita dal cliente non è integra `List*` e `Delete*` APIs non esegue alcuna decrittografia.

- Lista* APIs: tutte le operazioni di inserzione (ad esempio,,) `ListThings` `ListPolicies` `ListCertificates`
- Elimina* APIs: tutte le operazioni di eliminazione (ad esempio,,`DeleteThing`) `DeletePolicy` `DeleteCertificate`

Fase 5: Gestione delle chiavi

AWS IoT Core esegue controlli periodici sulla configurazione delle chiavi gestite dal cliente per garantire che le operazioni di crittografia e decrittografia non siano influenzate. Questi controlli di integrità vengono eseguiti una volta al minuto e verificano la capacità AWS IoT Core di accedere e utilizzare sia la AWS KMS chiave che il ruolo IAM associato per le operazioni di crittografia e decrittografia.

HEALTHY

AWS IoT Core può accedere con successo alla AWS KMS chiave tramite il ruolo IAM specificato ed eseguire operazioni. encryption/decryption Tutti i componenti funzionano correttamente.

UNHEALTHY

AWS IoT Core non può accedere o utilizzare la AWS KMS chiave. Ciò impedisce nuove operazioni di crittografia e può influire sulla funzionalità del servizio. Il `errorCode` campo indica se il problema riguarda la chiave o il ruolo IAM.

Azioni dei clienti che possono influire sulla salute dei clienti

Diverse azioni dei clienti possono far sì che lo stato di salute chiave cambi `HEALTHY` da`UNHEALTHY`:

Azioni relative alle chiavi

- Eliminazione di una AWS KMS chiave: quando pianifichi l'eliminazione di una chiave, questa è in uno `Pending deletion` stato e non può essere utilizzata
- Disabilitazione di una AWS KMS chiave: quando disabiliti una chiave KMS, questa non può più essere utilizzata per operazioni di crittografia/decrittografia
- Chiave di pianificazione per l'eliminazione: la chiave diventa inutilizzabile al termine dell'eliminazione
- Modifica della politica delle chiavi: rimozione delle autorizzazioni necessarie per l'accesso AWS IoT Core
- Modifica delle autorizzazioni di utilizzo delle chiavi: limitazione delle azioni richieste AWS KMS

Azioni relative al ruolo IAM

- Eliminazione del ruolo IAM: non è AWS IoT Core possibile assumere il ruolo di accesso alla chiave
- Modifica delle autorizzazioni dei ruoli: rimozione delle autorizzazioni richieste dalla AWS KMS politica del ruolo

- Modifica della politica di fiducia: impedire al AWS IoT Core servizio di assumere il ruolo
- Aggiungere condizioni restrittive: condizioni che AWS IoT Core impediscono l'utilizzo del ruolo

Azioni a livello di account

- Modifiche di accesso alle chiavi tra account: modifica delle autorizzazioni per le chiavi in diversi account
- Policy di controllo dei servizi (SCPs): politiche a livello di organizzazione che limitano l'accesso AWS KMS
- Policy IAM a livello di account: politiche che hanno la precedenza o sono in conflitto con l'accesso chiave

> **Important**
>
> Qualsiasi modifica alle AWS KMS chiavi, ai ruoli IAM o alle policy utilizzate da AWS IoT Core deve essere prima testata negli ambienti di sviluppo. Monitora attentamente lo stato di integrità delle chiavi dopo aver apportato modifiche per garantire che AWS IoT Core le funzionalità non ne risentano.

Aggiornamento della configurazione di crittografia

Aggiorna la configurazione di crittografia AWS IoT Core per passare da una chiave gestita dal cliente a un'altra o tra chiavi AWS di proprietà e chiavi gestite dal cliente.

Per modificare la configurazione con una chiave gestita dal cliente diversa:

1. Crea una nuova chiave gestita dal cliente seguendo i passaggi seguentithe section called “Fase 1: creare una chiave gestita dal cliente”.
2. Aggiorna la tua policy sui ruoli IAM per includere le autorizzazioni sia per la vecchia che per la nuova chiave durante il periodo di aggiornamento.
3. Aggiorna la configurazione di crittografia per utilizzare la nuova chiave:

```
aws iot update-encryption-configuration --encryption-type "CUSTOMER_MANAGED_KMS_KEY"
 \
--kms-access-role-arn "arn:aws:iam::111122223333:role/myrole" \
--kms-key-arn "arn:aws:kms:us-west-2:111122223333:key/new-key-id"
```

Per modificare la configurazione da chiavi gestite dal cliente a chiavi AWS di proprietà:

```
aws iot update-encryption-configuration --encryption-type "AWS_OWNED_KMS_KEY"
```

> **Note**
>
> Quando aggiorni la configurazione di crittografia per le nuove chiavi gestite dal cliente, assicurati che sia la vecchia che la nuova chiave rimangano accessibili affinché l'operazione abbia successo.

Scenari di errore e impatti comuni

La tabella seguente descrive gli scenari di errore più comuni quando le chiavi vengono eliminate o disattivate:

Scenario	Impatto immediato	Le conseguenze a lungo termine
Chiave disattivata	Tutte le nuove encryption/decryption operazioni falliscono immediatamente	Interruzione del servizio fino alla riattivazione o alla sostituzione della chiave
Chiave pianificata per l'eliminazione	Lo stato della chiave viene modificato in In attesa di eliminazione e tutte le encryption/decryption operazioni avranno esito negativo	Errore automatico del servizio al termine dell'eliminazione
Chiave eliminata definitivamente	Fallimento immediato e permanente di tutte le operazioni	Perdita permanente dei dati e impossibilità di recuperare i dati crittografati
Politica chiave modificata in modo errato	AWS IoT Core perde i permessi di accesso alla chiave	Guasti del servizio fino alla correzione della politica
Ruolo IAM eliminato	AWS IoT Core non può assumere il ruolo di chiave di accesso	Errore completo del servizio di crittografia

Scenario	Impatto immediato	Le conseguenze a lungo termine
Il ruolo IAM è stato modificato in modo errato	AWS IoT Core non può assumere il ruolo o utilizzare il ruolo per accedere alla chiave	Guasti del servizio fino alla correzion e del ruolo IAM

Prevenzione e best practice

Per prevenire l'eliminazione o la disattivazione accidentale delle chiavi e ridurre al minimo il rischio di guasti del servizio:

Implementa le politiche chiave relative al ciclo di vita

Stabilisci procedure chiare per la creazione, la rotazione e il ritiro delle chiavi. Documenta quali chiavi vengono utilizzate da quali AWS IoT Core risorse e mantieni un inventario delle chiavi attive.

Utilizza le policy IAM per limitare l'eliminazione delle chiavi

Crea policy IAM che impediscano agli utenti non autorizzati di eliminare o disabilitare le chiavi di crittografia critiche. Utilizza le condizioni per richiedere un'ulteriore approvazione per le operazioni di eliminazione delle chiavi.

Abilita la CloudTrail registrazione

Monitora tutte le operazioni AWS KMS chiave CloudTrail per rilevare attività di gestione delle chiavi non autorizzate o accidentali. Imposta avvisi per l'eliminazione, la disabilitazione o la modifica delle politiche delle chiavi.

Prova le procedure di sostituzione delle chiavi

Verifica regolarmente le tue procedure di sostituzione delle chiavi in ambienti non di produzione per assicurarti di poter ripristinare rapidamente i guasti relativi alle chiavi.

Mantenete i backup chiave

Sebbene non sia possibile esportare il materiale AWS KMS chiave, conservate registrazioni dettagliate delle chiavi ARNs, delle politiche e delle AWS IoT Core configurazioni associate per facilitare la sostituzione rapida delle chiavi, se necessario.

Monitora lo stato delle chiavi

Monitora continuamente la `CMK.Health` metrica e imposta avvisi automatici per le principali modifiche dello stato di salute. Implementa risposte automatiche per risolvere rapidamente i problemi principali.

> **Important**
>
> Verifica sempre le procedure di aggiornamento chiave negli ambienti di sviluppo prima di implementarle in produzione. Predisponi di un piano di rollback documentato e assicurati che le procedure di sostituzione delle chiavi possano essere eseguite rapidamente in caso di emergenza.

Fase 6: Monitoraggio dello stato delle chiavi

Nell'ambito dei controlli periodici, AWS IoT Core vengono emessi CloudWatch parametri e registri per fornire visibilità sullo stato di integrità della configurazione chiave gestita dal cliente

AWS IoT Core emette la `CMK.Health` metrica almeno una volta al minuto. CloudWatch La metrica fornisce informazioni sullo stato di integrità delle chiavi gestite dal cliente utilizzate AWS IoT Core per crittografare e decrittografare i dati.

La `CMK.Health` metrica può avere i seguenti valori:

- Il valore è`1`: AWS IoT Core è in grado di utilizzare correttamente le chiavi di crittografia per crittografare e decrittografare i dati.
- Il valore è`0`: non AWS IoT Core è in grado di utilizzare le chiavi di crittografia per crittografare e decrittografare i dati.

AWS IoT Core emette anche i log AWS IoT V2 quando cambia lo stato di integrità delle chiavi di crittografia. Questi registri forniscono ulteriori dettagli sull'aggiornamento dello stato di salute. Per visualizzare questi registri, è necessario abilitare i registri AWS IoT V2. I `HEALTHY` log vengono emessi a `INFO` livello e i `UNHEALTHY` log vengono emessi a livello. `ERROR` Per ulteriori informazioni sui livelli di registro, vedere Livelli di registro.

Gli esempi seguenti sono voci di CloudWatch registro emesse da AWS IoT Core per indicare l'aggiornamento dello stato di integrità delle chiavi gestite dal cliente.

Per monitorare e rispondere efficacemente ai principali cambiamenti dello stato di salute:

1. Imposta gli CloudWatch allarmi per la `CMK.Health` metrica:

```
aws cloudwatch put-metric-alarm --region us-west-2 \
  --alarm-name "IoTCore-CMK-Health-Alert" \
  --alarm-description "Alert when IoT Core CMK health is unhealthy" \
  --metric-name "CMK.Health" \
  --namespace "AWS/IoT" \
  --statistic "Minimum" \
  --period 300 \
  --evaluation-periods 1 \
  --threshold 1 \
  --comparison-operator "LessThanThreshold" \
  --alarm-actions "arn:aws:sns:us-west-2:111122223333:iot-alerts"
```

2. Abilita la registrazione AWS IoT V2 per acquisire eventi dettagliati di modifica dello stato di salute con codici e messaggi di errore.
3. Controlla lo stato della configurazione per la risoluzione dei problemi:

```
aws iot describe-encryption-configuration --region us-west-2
```

4. Analizza lo stato NON SANO esaminando il `errorCode` campo:
   - `KMS_KEY_VALIDATION_ERROR`— Problema con la AWS KMS chiave (disattivata, eliminata o problemi relativi alle policy)
   - `ROLE_VALIDATION_ERROR`— Problema relativo al ruolo IAM (eliminazione, problemi relativi alle policy o problemi di fiducia)

Da MALSANO a SANO

Quando lo stato delle chiavi di crittografia viene aggiornato da `UNHEALTHY` a`HEALTHY`, AWS IoT Core emetterà un messaggio di registro AWS IoT V2 nel seguente formato.

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "8421693b-f4f0-4e4a-9235-0cff8bab897d",
    "accountId": "111122223333",
    "status": "SUCCESS",
    "cmkStatus": "HEALTHY",
```

```
    "kmsKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "kmsAccessRoleArn": "arn:aws:iam::111122223333:role/myrole",
    "eventType": "CmkHealthCheck"
}
```

Da HEALTHY a UNHEALTHY

Quando lo stato delle chiavi di crittografia viene aggiornato da `HEALTHY` a`UNHEALTHY`, AWS IoT Core emetterà un messaggio di registro AWS IoT V2 nel seguente formato.

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "ERROR",
    "traceId": "8421693b-f4f0-4e4a-9235-0cff8bab897d",
    "accountId": "111122223333",
    "status": "FAILURE",
    "cmkStatus": "UNHEALTHY",
    "errorCode": "KMS_KEY_VALIDATION_ERROR / ROLE_VALIDATION_ERROR",
    "errorMessage": "Error message on why there was a failure",
    "kmsKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "kmsAccessRoleArn": "arn:aws:iam::111122223333:role/myrole",
    "eventType": "CmkHealthCheck"
}
```

> **Warning**
>
> Quando la chiave diventa integra`UNHEALTHY`, AWS IoT Core le operazioni falliscono immediatamente. In tal caso, esamina le configurazioni chiave, le autorizzazioni dei ruoli IAM e le policy. Monitora la `CMK.Health` metrica per le modifiche allo stato. Se le operazioni continuano a fallire dopo aver esaminato le configurazioni, contatta il tuo account manager o il AWS Support Center per ulteriore assistenza.

AWS CloudTrail eventi

È inoltre possibile monitorare l'utilizzo AWS IoT Core della chiave KMS per le operazioni di crittografia e decrittografia. AWS IoT Core `DescribeKey`effettuerà e eseguirà `GenerateDataKeyWithoutPlaintext` operazioni sulla tua chiave KMS per crittografare/

decrittografare i dati che appartengono al tuo account archiviati in archivio. `Decrypt` `ReEncrypt` AWS

Ci sono CloudTrail eventi per`DescribeKey`,, e`Decrypt`. `ReEncrypt` `GenerateDataKeyWithoutPlaintext` Questi eventi monitorano AWS KMS le operazioni chiamate AWS IoT Core ad accedere ai dati crittografati dalla chiave gestita dal cliente.

`Decrypt` Esempio


```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "**********************",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "*****"
            },
            "attributes": {
                "creationDate": "2024-09-16T20:23:39Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "iot.amazonaws.com"
    },
    "eventTime": "2024-09-16T20:32:48Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "iot.amazonaws.com",
    "userAgent": "iot.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "kms-arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "aws-crypto-ec:vendor": "iot.amazonaws.com",
```

```
            "branch-key-id": "111122223333",
            "type": "branch:ACTIVE"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "1afb6d98-8388-455d-8b48-e62c9e0cf7f4",
    "eventID": "b59a5f16-0d98-46d8-a590-0e040a48b39b",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Gestione delle identità e degli accessi per AWS IoT

AWS Identity and Access Management (IAM) è uno strumento Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse. AWS IoT IAM è uno Servizio AWS strumento che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità IAM
- Gestione dell’accesso tramite policy
- Come AWS IoT funziona con IAM
- AWS IoT esempi di politiche basate sull'identità

- AWS politiche gestite per AWS IoT
- Risoluzione dei problemi di AWS IoT identità e accesso

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all'amministratore se non riesci ad accedere alle funzionalità (consulta Risoluzione dei problemi di AWS IoT identità e accesso)
- Amministratore del servizio: determina l'accesso degli utenti e invia le richieste di autorizzazione (consulta Come AWS IoT funziona con IAM)
- Amministratore IAM: scrivi policy per gestire l'accesso (consulta AWS IoT esempi di politiche basate sull'identità)

Autenticazione con identità IAM

AWS IoT Le identità possono essere certificati del dispositivo (X.509), identità Amazon Cognito o utenti o gruppi IAM. In questo argomento vengono illustrate solo le identità IAM. Per ulteriori informazioni sulle altre identità supportate, consulta. AWS IoT Autenticazione client

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come AWS IAM Identity Center (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/ Facebook Per ulteriori informazioni sull'accesso, consulta Come accedere all' Account AWS nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali come

utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Utenti e gruppi IAM

Un utente IAM è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ti consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo passando da un ruolo utente a un ruolo IAM (console) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta Metodi per assumere un ruolo nella Guida per l'utente di IAM.

I ruoli IAM sono utili per l'accesso degli utenti federati, le autorizzazioni utente IAM temporanee, l'accesso multi-account, l'accesso multi-servizio e le applicazioni in esecuzione su Amazon EC2. Per maggiori informazioni, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile collegare a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una policy basata su identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Le policy basate su identità possono essere policy in linea (con embedding direttamente in una singola identità) o policy gestite (policy autonome collegate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scegliere tra policy gestite e policy in linea nella Guida per l'utente di IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio specificare un'entità principale.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Elenchi di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano. AWS WAF ACLs Per ulteriori informazioni ACLs, consulta la panoramica della lista di controllo degli accessi (ACL) nella Amazon Simple Storage Service Developer Guide.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo del servizio (SCPs): specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa in. AWS Organizations Per ulteriori informazioni, consultare Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Politiche di controllo delle risorse (RCPs): imposta le autorizzazioni massime disponibili per le risorse nei tuoi account. Per ulteriori informazioni, consulta Politiche di controllo delle risorse (RCPs) nella Guida per l'AWS Organizations utente.
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Come AWS IoT funziona con IAM

Prima di utilizzare IAM per gestire l'accesso a AWS IoT, è necessario comprendere con quali funzionalità IAM è disponibile l'uso AWS IoT. Per avere una visione di alto livello di come AWS IoT e altri AWS servizi funzionano con IAM, consulta AWS Services That Work with IAM nella IAM User Guide.

Argomenti

- AWS IoT politiche basate sull'identità
- AWS IoT politiche basate sulle risorse
- Autorizzazione basata su tag AWS IoT
- AWS IoT Ruoli IAM

AWS IoT politiche basate sull'identità

Con le policy basate su identità IAM, puoi specificare operazioni e risorse consentite o rifiutate, nonché le condizioni in base alle quali le operazioni sono consentite o rifiutate. AWS IoT supporta operazioni, risorse e chiavi di condizione specifiche. Per informazioni su tutti gli elementi utilizzati in una policy JSON, consulta Documentazione di riferimento degli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Azioni

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Includere le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

La tabella seguente elenca le azioni IAM IoT, l' AWS IoT API associata e la risorsa manipolata dall'azione.

Operazioni di policy	AWS IoT API	Resources
IoT: AcceptCertificateTransfer	AcceptCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** Il Account AWS valore specificato nell'ARN deve essere l'account su cui viene trasferito il certificato.
IoT: AddThingToThingGroup	AddThingToThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*

Operazioni di policy	AWS IoT API	Resources
IoT: Associate TargetsWithJob	Associate TargetsWithJob	nessuno
IoT: AttachPolicy	AttachPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* or `arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: AttachPrincipalPolicy	AttachPrincipalPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: AttachSecurityProfile	AttachSecurityProfile	`arn:aws:iot:` *`region`*`:`*`account-id`* `:securityprofile/` *`security-profile-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:dimension/` *`dimension-name`*
IoT: AttachThingPrincipal	AttachThingPrincipal	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: CancelCertificateTransfer	CancelCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** Il Account AWS valore specificato nell'ARN deve essere l'account su cui viene trasferito il certificato.
IoT: CancelJob	CancelJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*

Operazioni di policy	AWS IoT API	Resources
IoT: CancelJob Execution	CancelJob Execution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: ClearDefaultAuthorizer	ClearDefaultAuthorizer	Nessuno
IoT: CreateAuthorizer	CreateAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name*
IoT: CreateCertificateFromCsr	CreateCertificateFromCsr	*
IoT: CreateDimension	CreateDimension	arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: CreateJob	CreateJob	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateJobTemplate	CreateJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateKeysAndCertificate	CreateKeysAndCertificate	*

Operazioni di policy	AWS IoT API	Resources
IoT: CreatePolicy	CreatePolicy	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT: CreatePolicyVersion	CreatePolicyVersion	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name* **Note** Questa deve essere una AWS IoT politica, non una politica IAM.
IoT: CreateRoleAlias	CreateRoleAlias	(parametro: roleAlias) `arn:aws:iot:` *region*`:`*account-id* `:rolealias/` *role-alias-name*
IoT: CreateSecurityProfile	CreateSecurityProfile	`arn:aws:iot:` *region*`:`*account-id* `:securityprofile/` *security-profile-name* `arn:aws:iot:` *region*`:`*account-id* `:dimension/` *dimension-name*
IoT: CreateThing	CreateThing	`arn:aws:iot:` *region*`:`*account-id* `:thing/`*thing-name*
IoT: CreateThingGroup	CreateThingGroup	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name* Per il gruppo in fase di creazione e per il gruppo padre, se usato
IoT: CreateThingType	CreateThingType	`arn:aws:iot:` *region*`:`*account-id* `:thingtype/` *thing-type-name*
IoT: CreateTopicRule	CreateTopicRule	`arn:aws:iot:` *region*`:`*account-id* `:rule/`*rule-name*

Operazioni di policy	AWS IoT API	Resources
IoT: DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-name*
IoT: elimina CACertificate	Eliminare CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT: DeleteCertificate	DeleteCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DeleteDimension	DeleteDimension	arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: DeleteJob	DeleteJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT: DeleteJobTemplate	DeleteJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-template-id*
IoT: DeleteJobExecution	DeleteJobExecution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeletePolicy	DeletePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeletePolicyVersion	DeletePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeleteRegistrationCode	DeleteRegistrationCode	*
IoT: DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*

Operazioni di policy	AWS IoT API	Resources
IoT: DeleteSecurityProfile	DeleteSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: DeleteThing	DeleteThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeleteThingGroup	DeleteThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DeleteThingType	DeleteThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DeleteTopicRule	DeleteTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: elimina V2 LoggingLevel	Elimina V2 LoggingLevel	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DeprecateThingType	DeprecateThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DescribeAuthorizer	DescribeAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name* (parametro: authorizerName) nessuno
IoT: descrivi CACertificate	Descriva CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT: DescribeCertificate	DescribeCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*

Operazioni di policy	AWS IoT API	Resources
IoT: DescribeDefaultAuthorizer	DescribeDefaultAuthorizer	Nessuno
IoT: DescribeEndpoint	DescribeEndpoint	*
IoT: DescribeEventConfigurations	DescribeEventConfigurations	nessuno
IoT: DescribeIndex	DescribeIndex	`arn:aws:iot:` *`region`*`:`*`account-id`* `:index/`*`index-name`*
IoT: DescribeJob	DescribeJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*
IoT: DescribeJobExecution	DescribeJobExecution	Nessuno
IoT: DescribeJobTemplate	DescribeJobTemplate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-template-id`*
IoT: DescribeRoleAlias	DescribeRoleAlias	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rolealias/` *`role-alias-name`*
IoT: DescribeThing	DescribeThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: DescribeThingGroup	DescribeThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT: DescribeThingRegistrationTask	DescribeThingRegistrationTask	Nessuno
IoT: DescribeThingType	DescribeThingType	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thingtype/` *`thing-type-name`*

Operazioni di policy	AWS IoT API	Resources
IoT: DetachPolicy	DetachPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* or `arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT: DetachPrincipalPolicy	DetachPrincipalPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: DetachSecurityProfile	DetachSecurityProfile	`arn:aws:iot:` *`region`*`:`*`account-id`* `:securityprofile/` *`security-profile-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:dimension/` *`dimension-name`*
IoT: DetachThingPrincipal	DetachThingPrincipal	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: DisableTopicRule	DisableTopicRule	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rule/`*`rule-name`*
IoT: EnableTopicRule	EnableTopicRule	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rule/`*`rule-name`*
IoT: GetEffectivePolicies	GetEffectivePolicies	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: GetIndexingConfiguration	GetIndexingConfiguration	Nessuno
IoT: GetJobDocument	GetJobDocument	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*
IoT: GetLoggingOptions	GetLoggingOptions	*

Operazioni di policy	AWS IoT API	Resources
IoT: GetPolicy	GetPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: GetPolicy Version	GetPolicyVersion	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: GetRegistrationCode	GetRegistrationCode	*
IoT: GetTopicRule	GetTopicRule	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rule/`*`rule-name`*
IoT: ListAttachedPolicies	ListAttachedPolicies	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* or `arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: ListAuthorizers	ListAuthorizers	Nessuno
IoT: elencoCACertificates	Elenco CACertificates	*
IoT: ListCertificates	ListCertificates	*
iot: ListCertificatesBy CA	ListCertificatesByCA	*
IoT: ListIndices	ListIndices	Nessuno
IoT: ListJobExecutionsForJob	ListJobExecutionsForJob	Nessuno

Operazioni di policy	AWS IoT API	Resources
IoT: ListJobExecutionsForThing	ListJobExecutionsForThing	Nessuno
IoT: ListJobs	ListJobs	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* se il thingGroupName parametro è usato
iot: ListJobTemplates	ListJobs	Nessuno
IoT: ListOutgoingCertificates	ListOutgoingCertificates	*
IoT: ListPolicies	ListPolicies	*
IoT: ListPolicyPrincipals	ListPolicyPrincipals	*
IoT: ListPolicyVersions	ListPolicyVersions	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: ListPrincipalPolicies	ListPrincipalPolicies	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: ListPrincipalThings	ListPrincipalThings	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: ListRoleAliases	ListRoleAliases	Nessuno
IoT: ListTargetsForPolicy	ListTargetsForPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: ListThingGroups	ListThingGroups	Nessuno

Operazioni di policy	AWS IoT API	Resources
IoT: ListThing GroupsForThing	ListThing GroupsForThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: ListThing Principals	ListThing Principals	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: ListThing RegistrationTaskReports	ListThing RegistrationTaskReports	Nessuno
IoT: ListThing RegistrationTasks	ListThing RegistrationTasks	Nessuno
IoT: ListThing Types	ListThingTypes	*
IoT: ListThings	ListThings	*
IoT: ListThing sInThingGroup	ListThing sInThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT: ListTopic Rules	ListTopicRules	*
IoT: listv2 LoggingLevels	Elenco V2 LoggingLevels	Nessuno
IoT: registrazione CACertificate	RegistrationCACertificate	*
IoT: RegisterCertificate	RegisterCertificate	*

Operazioni di policy	AWS IoT API	Resources
IoT: RegisterThing	RegisterThing	Nessuno
IoT: RejectCertificateTransfer	RejectCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: RemoveThingFromThingGroup	RemoveThingFromThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: ReplaceTopicRule	ReplaceTopicRule	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rule/`*`rule-name`*
IoT: SearchIndex	SearchIndex	`arn:aws:iot:` *`region`*`:`*`account-id`* `:index/`*`index-id`*
IoT: SetDefaultAuthorizer	SetDefaultAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-function-name`*
IoT: SetDefaultPolicyVersion	SetDefaultPolicyVersion	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: SetLoggingOptions	SetLoggingOptions	`arn:aws:iot:` *`region`*`:`*`account-id`* `:role/`*`role-name`*
IoT: setV2LoggingLevel	Impostare V2LoggingLevel	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT: setV2LoggingOptions	Impostare V2LoggingOptions	`arn:aws:iot:` *`region`*`:`*`account-id`* `:role/`*`role-name`*
IoT: StartThingRegistrationTask	StartThingRegistrationTask	Nessuno

Operazioni di policy	AWS IoT API	Resources
IoT: StopThing RegistrationTask	StopThing RegistrationTask	Nessuno
IoT: TestAutho rization	TestAutho rization	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: TestInvok eAuthorizer	TestInvok eAuthorizer	Nessuno
IoT: TransferC ertificate	TransferCertificat e	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: UpdateAut horizer	UpdateAut horizer	`arn:aws:iot:` *region*`:`*account-id* `:authoriz erfunction/` *authorizer-function-name*
IoT: aggiornam ento CACertifi cate	Aggiorna CACertificate	`arn:aws:iot:` *region*`:`*account-i d* `:cacert/`*cert-id*
IoT: UpdateCer tificate	UpdateCer tificate	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: UpdateDim ension	UpdateDim ension	`arn:aws:iot:` *region*`:`*account-id* `:dimensio n/` *dimension-name*
IoT: UpdateEve ntConfigurations	UpdateEve ntConfigurations	Nessuno
IoT: UpdateInd exingConf iguration	UpdateInd exingConf iguration	Nessuno
IoT: UpdateRol eAlias	UpdateRoleAlias	`arn:aws:iot:` *region*`:`*account-id* `:rolealia s/` *role-alias-name*

Operazioni di policy	AWS IoT API	Resources
IoT: UpdateSecurityProfile	UpdateSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: UpdateThing	UpdateThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: UpdateThingGroup	UpdateThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: UpdateThingGroupsForThing	UpdateThingGroupsForThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name* arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*

Le azioni politiche AWS IoT utilizzano il seguente prefisso prima dell'azione:`iot:`. Ad esempio, per concedere a qualcuno l'autorizzazione a elencare tutti gli oggetti IoT registrati nell'`ListThings`API, includi l'`iot:ListThings`azione nella sua policy. Account AWS Le dichiarazioni politiche devono includere un `NotAction` elemento `Action` or. AWS IoT definisce il proprio set di azioni che descrivono le attività che è possibile eseguire con questo servizio.

Per specificare più azioni in una sola istruzione, separa ciascuna di esse con una virgola come mostrato di seguito:

```
"Action": [
      "ec2:action1",
      "ec2:action2"
```

È possibile specificare più azioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le azioni che iniziano con la parola `Describe`, includi la seguente azione:

```
"Action": "iot:Describe*"
```

Per visualizzare un elenco di AWS IoT azioni, consulta Actions Defined by AWS IoT nella IAM User Guide.

Operazioni di Device Advisor

Nella tabella seguente sono elencate le operazioni di IAM IoT Device Advisor, ovvero l'API AWS IoT Device Advisor associata e la risorsa manipolata dall'operazione.

Operazioni di policy	AWS IoT API	Resources
consulente per dispositivi IoT: CreateSuiteDefinition	CreateSuiteDefinition	Nessuno
consulente per dispositivi IoT: DeleteSuiteDefinition	DeleteSuiteDefinition	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`* `:suitedefinition/` *`suite-definition-id`*
consulente per dispositivi IoT: GetSuiteDefinition	GetSuiteDefinition	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`* `:suitedefinition/` *`suite-definition-id`*
consulente per dispositivi IoT: GetSuiteRun	GetSuiteRun	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`* `:suitedefinition/` *`suite-run-id`*
consulente per dispositivi IoT: GetSuiteRunReport	GetSuiteRunReport	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`* `:suiterun/` *`suite-definition-id`* `/`*`suite-run-id`*
consulente per dispositivi IoT: ListSuiteDefinitions	ListSuiteDefinitions	Nessuno

Operazioni di policy	AWS IoT API	Resources
consulente per dispositivi IoT: ListSuiteRuns	ListSuiteRuns	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*
consulente per dispositivi IoT: ListTagsForResource	ListTagsForResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
consulente per dispositivi IoT: StartSuiteRun	StartSuiteRun	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*
consulente per dispositivi IoT: TagResource	TagResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
consulente per dispositivi IoT: UntagResource	UntagResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
consulente per dispositivi IoT: UpdateSuiteDefinition	UpdateSuiteDefinition	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*

Operazioni di policy	AWS IoT API	Resources
consulente per dispositivi IoT: StopSuiteRun	StopSuiteRun	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`* `:suiterun/suite-definition-id/` *`suite-run-id`*

Le azioni politiche in AWS IoT Device Advisor utilizzano il seguente prefisso prima dell'azione:. `iotdeviceadvisor:` Ad esempio, per concedere a qualcuno il permesso di elencare tutte le definizioni di suite registrate nella sua Account AWS ListSuiteDefinitions API, includi l'`iotdeviceadvisor:ListSuiteDefinitions`azione nella sua politica.

Resources

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

AWS IoT risorse

Operazioni di policy	AWS IoT API	Resources
IoT: AcceptCertificateTransfer	AcceptCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** Il Account AWS valore specificato nell'ARN deve essere l'account su cui viene trasferito il certificato.

Operazioni di policy	AWS IoT API	Resources
IoT: AddThingToThingGroup	AddThingToThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: AssociateTargetsWithJob	AssociateTargetsWithJob	Nessuno
IoT: AttachPolicy	AttachPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* or `arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: AttachPrincipalPolicy	AttachPrincipalPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: AttachThingPrincipal	AttachThingPrincipal	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: CancelCertificateTransfer	CancelCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** Il Account AWS valore specificato nell'ARN deve essere l'account su cui viene trasferito il certificato.
IoT: CancelJob	CancelJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*

Operazioni di policy	AWS IoT API	Resources
IoT: CancelJob Execution	CancelJob Execution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: ClearDefaultAuthorizer	ClearDefaultAuthorizer	Nessuno
IoT: CreateAuthorizer	CreateAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name*
IoT: CreateCertificateFromCsr	CreateCertificateFromCsr	*
IoT: CreateJob	CreateJob	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateJobTemplate	CreateJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateKeysAndCertificate	CreateKeysAndCertificate	*
IoT: CreatePolicy	CreatePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*

Operazioni di policy	AWS IoT API	Resources
CreatePolicyVersion	IoT: CreatePolicyVersion	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`* **Note** Questa deve essere una AWS IoT politica, non una politica IAM.
IoT: CreateRoleAlias	CreateRoleAlias	(parametro: roleAlias) `arn:aws:iot:` *`region`*`:`*`account-id`* `:rolealias/` *`role-alias-name`*
IoT: CreateThing	CreateThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: CreateThingGroup	CreateThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* Per il gruppo in fase di creazione e per il gruppo padre, se usato
IoT: CreateThingType	CreateThingType	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thingtype/` *`thing-type-name`*
IoT: CreateTopicRule	CreateTopicRule	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rule/`*`rule-name`*
IoT: DeleteAuthorizer	DeleteAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-name`*
IoT: elimina CACertificate	Eliminare CACertificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cacert/`*`cert-id`*
IoT: DeleteCertificate	DeleteCertificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*

Operazioni di policy	AWS IoT API	Resources
IoT: DeleteJob	DeleteJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT: DeleteJob Execution	DeleteJob Execution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeleteJob Template	DeleteJob Template	arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: DeletePolicy	DeletePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeletePolicyVersion	DeletePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeleteRegistrationCode	DeleteRegistrationCode	*
IoT: DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT: DeleteThing	DeleteThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeleteThingGroup	DeleteThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DeleteThingType	DeleteThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DeleteTopicRule	DeleteTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: elimina V2 LoggingLevel	Elimina V2 LoggingLevel	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*

Operazioni di policy	AWS IoT API	Resources
IoT: Deprecate ThingType	Deprecate ThingType	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thingtype/` *`thing-type-name`*
IoT: DescribeAuthorizer	DescribeAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-function-name`* (parametro: authorizerName) nessuno
IoT: descrivi CACertificate	Descriva CACertificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cacert/`*`cert-id`*
IoT: DescribeCertificate	DescribeCertificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: DescribeDefaultAuthorizer	DescribeDefaultAuthorizer	Nessuno
IoT: DescribeEndpoint	DescribeEndpoint	*
IoT: DescribeEventConfigurations	DescribeEventConfigurations	nessuno
IoT: DescribeIndex	DescribeIndex	`arn:aws:iot:` *`region`*`:`*`account-id`* `:index/`*`index-name`*
IoT: DescribeJob	DescribeJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*
IoT: DescribeJobExecution	DescribeJobExecution	Nessuno
IoT: DescribeJobTemplate	DescribeJobTemplate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:jobtemplate/` *`job-template-id`*

Operazioni di policy	AWS IoT API	Resources
IoT: DescribeRoleAlias	DescribeRoleAlias	`arn:aws:iot:` *region*`:`*account-id* `:rolealias/` *role-alias-name*
IoT: DescribeThing	DescribeThing	`arn:aws:iot:` *region*`:`*account-id* `:thing/`*thing-name*
IoT: DescribeThingGroup	DescribeThingGroup	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name*
IoT: DescribeThingRegistrationTask	DescribeThingRegistrationTask	Nessuno
IoT: DescribeThingType	DescribeThingType	`arn:aws:iot:` *region*`:`*account-id* `:thingtype/` *thing-type-name*
IoT: DetachPolicy	DetachPolicy	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id* or `arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name*
IoT: DetachPrincipalPolicy	DetachPrincipalPolicy	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: DetachThingPrincipal	DetachThingPrincipal	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: DisableTopicRule	DisableTopicRule	`arn:aws:iot:` *region*`:`*account-id* `:rule/`*rule-name*
IoT: EnableTopicRule	EnableTopicRule	`arn:aws:iot:` *region*`:`*account-id* `:rule/`*rule-name*

Operazioni di policy	AWS IoT API	Resources
IoT: GetEffectivePolicies	GetEffectivePolicies	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: GetIndexingConfiguration	GetIndexingConfiguration	Nessuno
IoT: GetJobDocument	GetJobDocument	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*
IoT: GetLoggingOptions	GetLoggingOptions	*
IoT: GetPolicy	GetPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: GetPolicyVersion	GetPolicyVersion	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: GetRegistrationCode	GetRegistrationCode	*
IoT: GetTopicRule	GetTopicRule	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rule/`*`rule-name`*
IoT: ListAttachedPolicies	ListAttachedPolicies	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* or `arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: ListAuthorizers	ListAuthorizers	Nessuno
IoT: elencoCACertificates	Elenco CACertificates	*

Operazioni di policy	AWS IoT API	Resources
IoT: ListCertificates	ListCertificates	*
iot: ListCertificatesByCA	ListCertificatesByCA	*
IoT: ListIndices	ListIndices	Nessuno
IoT: ListJobExecutionsForJob	ListJobExecutionsForJob	Nessuno
IoT: ListJobExecutionsForThing	ListJobExecutionsForThing	Nessuno
IoT: ListJobs	ListJobs	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* se il thingGroupName parametro è usato
iot: ListJobTemplates	ListJobTemplates	Nessuno
IoT: ListOutgoingCertificates	ListOutgoingCertificates	*
IoT: ListPolicies	ListPolicies	*
IoT: ListPolicyPrincipals	ListPolicyPrincipals	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: ListPolicyVersions	ListPolicyVersions	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: ListPrincipalPolicies	ListPrincipalPolicies	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*

Operazioni di policy	AWS IoT API	Resources
IoT: ListPrincipalThings	ListPrincipalThings	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: ListRoleAliases	ListRoleAliases	Nessuno
IoT: ListTargetsForPolicy	ListTargetsForPolicy	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT: ListThingGroups	ListThingGroups	Nessuno
IoT: ListThingGroupsForThing	ListThingGroupsForThing	`arn:aws:iot:` *region*`:`*account-id* `:thing/`*thing-name*
IoT: ListThingPrincipals	ListThingPrincipals	`arn:aws:iot:` *region*`:`*account-id* `:thing/`*thing-name*
IoT: ListThingRegistrationTaskReports	ListThingRegistrationTaskReports	Nessuno
IoT: ListThingRegistrationTasks	ListThingRegistrationTasks	Nessuno
IoT: ListThingTypes	ListThingTypes	*
IoT: ListThings	ListThings	*
IoT: ListThingsInThingGroup	ListThingsInThingGroup	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name*
IoT: ListTopicRules	ListTopicRules	*

Operazioni di policy	AWS IoT API	Resources
IoT: listv2 LoggingLevels	Elenco V2 LoggingLevels	Nessuno
IoT: registrazione CACertificate	RegistratiCACertificate	*
IoT: RegisterCertificate	RegisterCertificate	*
IoT: RegisterThing	RegisterThing	Nessuno
IoT: RejectCertificateTransfer	RejectCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: RemoveThingFromThingGroup	RemoveThingFromThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: ReplaceTopicRule	ReplaceTopicRule	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rule/`*`rule-name`*
IoT: SearchIndex	SearchIndex	`arn:aws:iot:` *`region`*`:`*`account-id`* `:index/`*`index-id`*
IoT: SetDefaultAuthorizer	SetDefaultAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-function-name`*
IoT: SetDefaultPolicyVersion	SetDefaultPolicyVersion	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: SetLoggingOptions	SetLoggingOptions	*

Operazioni di policy	AWS IoT API	Resources
IoT: setV2 LoggingLevel	Impostare V2 LoggingLevel	*
IoT: setV2 LoggingOptions	Impostare V2 LoggingOptions	*
IoT: StartThin gRegistra tionTask	StartThin gRegistra tionTask	Nessuno
IoT: StopThing RegistrationTask	StopThing RegistrationTask	Nessuno
IoT: TestAutho rization	TestAutho rization	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: TestInvok eAuthorizer	TestInvok eAuthorizer	Nessuno
IoT: TransferC ertificate	TransferCertificat e	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: UpdateAut horizer	UpdateAut horizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authoriz erfunction/` *`authorizer-function-name`*
IoT: aggiornam ento CACertifi cate	Aggiorna CACertificate	`arn:aws:iot:` *`region`*`:`*`account-i d`* `:cacert/`*`cert-id`*
IoT: UpdateCer tificate	UpdateCer tificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: UpdateEve ntConfigurations	UpdateEve ntConfigurations	Nessuno

Operazioni di policy	AWS IoT API	Resources
IoT: UpdateIndexingConfiguration	UpdateIndexingConfiguration	Nessuno
IoT: UpdateRoleAlias	UpdateRoleAlias	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rolealias/` *`role-alias-name`*
IoT: UpdateThing	UpdateThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: UpdateThingGroup	UpdateThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT: UpdateThingGroupsForThing	UpdateThingGroupsForThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*

Per ulteriori informazioni sul formato di ARNs, consulta Amazon Resource Names (ARNs) e AWS Service Namespaces.

Alcune AWS IoT azioni, come quelle per la creazione di risorse, non possono essere eseguite su una risorsa specifica. In questi casi, è necessario utilizzare il carattere jolly (*).

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di AWS IoT risorse e relativi ARNs, consulta Resources Defined by AWS IoT nella IAM User Guide. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consulta Operazioni definite da AWS IoT.

Risorse di Device Advisor

Per definire le restrizioni a livello di risorsa per le politiche IAM di AWS IoT Device Advisor, utilizza i seguenti formati ARN di risorse per le definizioni e le esecuzioni delle suite.

Formato dell'ARN della risorsa di definizione della suite

```
arn:aws:iotdeviceadvisor:region:account-id:suitedefinition/suite-definition-id
```

Formato dell'ARN della risorsa di esecuzione della suite

```
arn:aws:iotdeviceadvisor:region:account-id:suiterun/suite-definition-id/suite-run-id
```

Chiavi di condizione

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

AWS IoT definisce il proprio set di chiavi di condizione e supporta anche l'utilizzo di alcune chiavi di condizione globali. Per visualizzare tutte le chiavi di condizione AWS globali, consulta AWS Global Condition Context Keys nella IAM User Guide.

AWS IoT chiavi di condizione

AWS IoT chiavi di condizione	Descrizione	Tipo
`aws:RequestTag/${tag-key}`	Una chiave del tag che è presente nella richiesta effettuata dall'utente verso AWS IoT.	Stringa
`aws:ResourceTag/${tag-key}`	Il componente chiave del tag di un tag allegato	Stringa

AWS IoT chiavi di condizione	Descrizione	Tipo
	a una AWS IoT risorsa.	
`aws:TagKeys`	L'elenco di tutti i nomi delle chiavi di tag associati alla risorsa nella richiesta.	Stringa

Per visualizzare un elenco di chiavi di AWS IoT condizione, consulta Condition Keys for AWS IoT nella IAM User Guide. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, consulta Azioni definite da AWS IoT.

Esempi

Per visualizzare esempi di politiche AWS IoT basate sull'identità, vedere. AWS IoT esempi di politiche basate sull'identità

AWS IoT politiche basate sulle risorse

Le politiche basate sulle risorse sono documenti di policy JSON che specificano quali azioni uno specifico principale può eseguire sulla risorsa e in quali condizioni. AWS IoT

AWS IoT non supporta le politiche basate sulle risorse IAM. Tuttavia, supporta politiche basate sulle risorse. AWS IoT Per ulteriori informazioni, consulta AWS IoT Core politiche.

Autorizzazione basata su tag AWS IoT

È possibile allegare tag alle AWS IoT risorse o passare tag in una richiesta a AWS IoT. Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `iot:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`. Per ulteriori informazioni, consulta Utilizzo dei tag con policy IAM. Per ulteriori informazioni sull'etichettatura AWS IoT delle risorse, consultaTaggare le tue risorse AWS IoT.

Per visualizzare un esempio di policy basata su identità per limitare l'accesso a una risorsa in base ai tag di tale risorsa, consulta Visualizzazione delle AWS IoT risorse in base ai tag.

AWS IoT Ruoli IAM

Un ruolo IAM è un'entità interna all'utente Account AWS che dispone di autorizzazioni specifiche.

Utilizzo di credenziali temporanee con AWS IoT

È possibile utilizzare credenziali temporanee per effettuare l'accesso con la federazione, assumere un ruolo IAM o un ruolo multi-account. È possibile ottenere credenziali di sicurezza temporanee chiamando operazioni AWS STS API come AssumeRoleo. GetFederationToken

AWS IoT supporta l'utilizzo di credenziali temporanee.

Ruoli collegati ai servizi

I ruoli collegati ai AWS servizi consentono ai servizi di accedere alle risorse di altri servizi per completare un'azione per conto dell'utente. I ruoli collegati ai servizi sono visualizzati nell'account IAM e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non può modificarle.

AWS IoT non supporta ruoli collegati ai servizi.

Ruoli dei servizi

Questa funzionalità consente a un servizio di assumere un ruolo di servizio per conto dell'utente. Questo ruolo consente al servizio di accedere alle risorse in altri servizi per completare un'azione per conto dell'utente. I ruoli dei servizi sono visualizzati nell'account IAM e sono di proprietà dell'account. Ciò significa che un amministratore IAM può modificare le autorizzazioni per questo ruolo. Tuttavia, questo potrebbe pregiudicare la funzionalità del servizio.

AWS IoT esempi di politiche basate sull'identità

Per impostazione predefinita, gli utenti e i ruoli IAM non dispongono dell'autorizzazione per creare o modificare risorse AWS IoT . Inoltre, non possono eseguire attività utilizzando l'API Console di gestione AWS AWS CLI, o. AWS Un amministratore IAM deve creare policy IAM che concedono a utenti e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse specificate di cui hanno bisogno. L'amministratore devi quindi collegare queste policy a utenti o gruppi che richiedono tali autorizzazioni.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy nella scheda JSON nella Guida per l'utente IAM.

Argomenti

- Best practice delle policy
- Utilizzo della console AWS IoT
- Consentire agli utenti di visualizzare le loro autorizzazioni
- Visualizzazione delle AWS IoT risorse in base ai tag
- Visualizzazione delle risorse di AWS IoT Device Advisor in base ai tag

Best practice delle policy

Le politiche basate sull'identità determinano se qualcuno può creare, accedere o eliminare AWS IoT risorse nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche gestite che concedono le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.
- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy

nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente di IAM.

- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare Protezione dell'accesso API con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console AWS IoT

Per accedere alla AWS IoT console, è necessario disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle AWS IoT risorse del tuo. Account AWS Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Per garantire che tali entità possano ancora utilizzare la AWS IoT console, allega anche la seguente politica AWS gestita alle entità:`AWSIoTFullAccess`. Per ulteriori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente IAM.

Non è necessario consentire le autorizzazioni minime della console per gli utenti che effettuano chiamate solo verso AWS CLI o l' AWS API. Al contrario, è possibile accedere solo alle operazioni che soddisfano l'operazione API che stai cercando di eseguire.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o a livello di codice. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Visualizzazione delle AWS IoT risorse in base ai tag

È possibile utilizzare le condizioni nella policy basata sulle identità per controllare l'accesso alle risorse di AWS IoT in base ai tag. Questo esempio mostra come creare una policy che consente di visualizzare un oggetto. Tuttavia, l'autorizzazione viene concessa solo se il valore del tag dell'oggetto `0wner` corrisponde a quello del nome utente. Questa policy concede anche le autorizzazioni necessarie per completare questa azione nella console.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ListBillingGroupsInConsole",
```

```
            "Effect": "Allow",
            "Action": "iot:ListBillingGroups",
            "Resource": "*"
        },
        {
            "Sid": "ViewBillingGroupsIfOwner",
            "Effect": "Allow",
           "Action": "iot:DescribeBillingGroup",
            "Resource": "arn:aws:iot:*:*:billinggroup/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
        }
        }
    ]
}
```

Puoi allegare questa policy agli utenti IAM nel tuo account. Se un utente denominato `richard-roe` tenta di visualizzare un gruppo di AWS IoT fatturazione, il gruppo di fatturazione deve essere taggato `Owner=richard-roe` o. `owner=richard-roe` In caso contrario, gli viene negato l'accesso. La chiave di tag di condizione `Owner` corrisponde a `Owner` e `owner` perché i nomi delle chiavi di condizione non effettuano la distinzione tra maiuscole e minuscole. Per ulteriori informazioni, consulta la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.

Visualizzazione delle risorse di AWS IoT Device Advisor in base ai tag

Puoi utilizzare le condizioni nella policy basata sulle identità per controllare l'accesso alle risorse di AWS IoT Device Advisor in base ai tag. Nell'esempio seguente viene illustrato come creare una policy che consente di visualizzare una determinata definizione di suite. Tuttavia, l'autorizzazione viene concessa solo se il tag di definizione della suite `SuiteType` è impostato sul valore di `MQTT`. Questa policy concede anche le autorizzazioni necessarie per completare questa azione nella console.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ViewSuiteDefinition",
            "Effect": "Allow",
            "Action": "iotdeviceadvisor:GetSuiteDefinition",
            "Resource": "arn:aws:iotdeviceadvisor:*:*:suitedefinition/*",
```

```
            "Condition": {
                "StringEquals": {"aws:ResourceTag/SuiteType": "MQTT"}
        }
        }
    ]
}
```

AWS politiche gestite per AWS IoT

Per aggiungere autorizzazioni a utenti, gruppi e ruoli, è più facile utilizzare le politiche AWS gestite che scrivere le politiche da soli. La creazione di policy gestite dai clienti IAM che forniscono al team solo le autorizzazioni di cui ha bisogno richiede tempo e competenza. Per iniziare rapidamente, puoi utilizzare le nostre politiche AWS gestite. Queste policy coprono i casi d'uso comuni e sono disponibili nell'account Account AWS. Per ulteriori informazioni sulle policy AWS gestite, consulta le policy AWS gestite nella IAM User Guide.

AWS i servizi mantengono e aggiornano le politiche AWS gestite. Non è possibile modificare le autorizzazioni nelle politiche AWS gestite. I servizi occasionalmente aggiungono altre autorizzazioni a una policy gestita da AWS per supportare nuove funzionalità. Questo tipo di aggiornamento interessa tutte le identità (utenti, gruppi e ruoli) a cui è collegata la policy. È più probabile che i servizi aggiornino una policy gestita da AWS quando viene avviata una nuova funzionalità o quando diventano disponibili nuove operazioni. I servizi non rimuovono le autorizzazioni da una policy AWS gestita, quindi gli aggiornamenti delle policy non comprometteranno le autorizzazioni esistenti.

Inoltre, AWS supporta politiche gestite per le funzioni lavorative che si estendono su più servizi. Ad esempio, la policy ReadOnlyAccess AWS gestita fornisce l'accesso in sola lettura a tutti i AWS servizi e le risorse. Quando un servizio avvia una nuova funzionalità, AWS aggiunge autorizzazioni di sola lettura per nuove operazioni e risorse. Per l'elenco e la descrizione delle policy di funzione dei processi, consultare la sezione Policy gestite da AWS per funzioni di processi nella Guida per l'utente di IAM.

> **Note**
>
> AWS IoT funziona sia con le politiche IAM che con quelle AWS IoT di IAM. Questo argomento è dedicato solo alle policy IAM, che definiscono un'operazione di policy per operazioni API piano di controllo (control-plane) e del piano dati. Consulta anche AWS IoT Core politiche.

AWS politica gestita: AWSIo TConfig accesso

È possibile allegare la policy `AWSIoTConfigAccess` alle identità IAM.

Questa policy concede all'identità associata le autorizzazioni per l'accesso a tutte le operazioni di configurazione AWS IoT . Questa policy può influenzare l'elaborazione e lo storage dei dati. Per visualizzare questa politica in Console di gestione AWS, vedere AWSIoTConfigAccess.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `iot`— Recupera AWS IoT dati ed esegui azioni di configurazione IoT.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AcceptCertificateTransfer",
                "iot:AddThingToThingGroup",
                "iot:AssociateTargetsWithJob",
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
                "iot:CancelCertificateTransfer",
                "iot:CancelJob",
                "iot:CancelJobExecution",
                "iot:ClearDefaultAuthorizer",
                "iot:CreateAuthorizer",
                "iot:CreateCertificateFromCsr",
                "iot:CreateJob",
                "iot:CreateKeysAndCertificate",
                "iot:CreateOTAUpdate",
```

```
                "iot:CreatePolicy",
                "iot:CreatePolicyVersion",
                "iot:CreateRoleAlias",
                "iot:CreateStream",
                "iot:CreateThing",
                "iot:CreateThingGroup",
                "iot:CreateThingType",
                "iot:CreateTopicRule",
                "iot:DeleteAuthorizer",
                "iot:DeleteCACertificate",
                "iot:DeleteCertificate",
                "iot:DeleteJob",
                "iot:DeleteJobExecution",
                "iot:DeleteOTAUpdate",
                "iot:DeletePolicy",
                "iot:DeletePolicyVersion",
                "iot:DeleteRegistrationCode",
                "iot:DeleteRoleAlias",
                "iot:DeleteStream",
                "iot:DeleteThing",
                "iot:DeleteThingGroup",
                "iot:DeleteThingType",
                "iot:DeleteTopicRule",
                "iot:DeleteV2LoggingLevel",
                "iot:DeprecateThingType",
                "iot:DescribeAuthorizer",
                "iot:DescribeCACertificate",
                "iot:DescribeCertificate",
                "iot:DescribeDefaultAuthorizer",
                "iot:DescribeEndpoint",
                "iot:DescribeEventConfigurations",
                "iot:DescribeIndex",
                "iot:DescribeJob",
                "iot:DescribeJobExecution",
                "iot:DescribeRoleAlias",
                "iot:DescribeStream",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingRegistrationTask",
                "iot:DescribeThingType",
                "iot:DetachPolicy",
                "iot:DetachPrincipalPolicy",
                "iot:DetachThingPrincipal",
                "iot:DisableTopicRule",
```

```
                "iot:EnableTopicRule",
                "iot:GetEffectivePolicies",
                "iot:GetIndexingConfiguration",
                "iot:GetJobDocument",
                "iot:GetLoggingOptions",
                "iot:GetOTAUpdate",
                "iot:GetPolicy",
                "iot:GetPolicyVersion",
                "iot:GetRegistrationCode",
                "iot:GetTopicRule",
                "iot:GetV2LoggingOptions",
                "iot:ListAttachedPolicies",
                "iot:ListAuthorizers",
                "iot:ListCACertificates",
                "iot:ListCertificates",
                "iot:ListCertificatesByCA",
                "iot:ListIndices",
                "iot:ListJobExecutionsForJob",
                "iot:ListJobExecutionsForThing",
                "iot:ListJobs",
                "iot:ListOTAUpdates",
                "iot:ListOutgoingCertificates",
                "iot:ListPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPolicyVersions",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListRoleAliases",
                "iot:ListStreams",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroups",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:ListThingRegistrationTaskReports",
                "iot:ListThingRegistrationTasks",
                "iot:ListThings",
                "iot:ListThingsInThingGroup",
                "iot:ListThingTypes",
                "iot:ListTopicRules",
                "iot:ListV2LoggingLevels",
                "iot:RegisterCACertificate",
                "iot:RegisterCertificate",
                "iot:RegisterThing",
                "iot:RejectCertificateTransfer",
```

```
                "iot:RemoveThingFromThingGroup",
                "iot:ReplaceTopicRule",
                "iot:SearchIndex",
                "iot:SetDefaultAuthorizer",
                "iot:SetDefaultPolicyVersion",
                "iot:SetLoggingOptions",
                "iot:SetV2LoggingLevel",
                "iot:SetV2LoggingOptions",
                "iot:StartThingRegistrationTask",
                "iot:StopThingRegistrationTask",
                "iot:TestAuthorization",
                "iot:TestInvokeAuthorizer",
                "iot:TransferCertificate",
                "iot:UpdateAuthorizer",
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:UpdateEventConfigurations",
                "iot:UpdateIndexingConfiguration",
                "iot:UpdateRoleAlias",
                "iot:UpdateStream",
                "iot:UpdateThing",
                "iot:UpdateThingGroup",
                "iot:UpdateThingGroupsForThing",
                "iot:UpdateAccountAuditConfiguration",
                "iot:DescribeAccountAuditConfiguration",
                "iot:DeleteAccountAuditConfiguration",
                "iot:StartOnDemandAuditTask",
                "iot:CancelAuditTask",
                "iot:DescribeAuditTask",
                "iot:ListAuditTasks",
                "iot:CreateScheduledAudit",
                "iot:UpdateScheduledAudit",
                "iot:DeleteScheduledAudit",
                "iot:DescribeScheduledAudit",
                "iot:ListScheduledAudits",
                "iot:ListAuditFindings",
                "iot:CreateSecurityProfile",
                "iot:DescribeSecurityProfile",
                "iot:UpdateSecurityProfile",
                "iot:DeleteSecurityProfile",
                "iot:AttachSecurityProfile",
                "iot:DetachSecurityProfile",
                "iot:ListSecurityProfiles",
                "iot:ListSecurityProfilesForTarget",
```

```
                "iot:ListTargetsForSecurityProfile",
                "iot:ListActiveViolations",
                "iot:ListViolationEvents",
                "iot:ValidateSecurityProfileBehaviors"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS politica gestita: AWSIo TConfig ReadOnlyAccess

È possibile allegare la policy `AWSIoTConfigReadOnlyAccess` alle identità IAM.

Questa policy concede all'identità associata le autorizzazioni per l'accesso in sola lettura a tutte le operazioni di configurazione AWS IoT . Per visualizzare questa politica in Console di gestione AWS, vedere AWSIoTConfigReadOnlyAccess.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `iot`: esegue operazioni di sola lettura di configurazione IoT.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeAuthorizer",
                "iot:DescribeCACertificate",
                "iot:DescribeCertificate",
                "iot:DescribeDefaultAuthorizer",
                "iot:DescribeEndpoint",
                "iot:DescribeEventConfigurations",
```

```
                "iot:DescribeIndex",
                "iot:DescribeJob",
                "iot:DescribeJobExecution",
                "iot:DescribeRoleAlias",
                "iot:DescribeStream",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingRegistrationTask",
                "iot:DescribeThingType",
                "iot:GetEffectivePolicies",
                "iot:GetIndexingConfiguration",
                "iot:GetJobDocument",
                "iot:GetLoggingOptions",
                "iot:GetOTAUpdate",
                "iot:GetPolicy",
                "iot:GetPolicyVersion",
                "iot:GetRegistrationCode",
                "iot:GetTopicRule",
                "iot:GetV2LoggingOptions",
                "iot:ListAttachedPolicies",
                "iot:ListAuthorizers",
                "iot:ListCACertificates",
                "iot:ListCertificates",
                "iot:ListCertificatesByCA",
                "iot:ListIndices",
                "iot:ListJobExecutionsForJob",
                "iot:ListJobExecutionsForThing",
                "iot:ListJobs",
                "iot:ListOTAUpdates",
                "iot:ListOutgoingCertificates",
                "iot:ListPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPolicyVersions",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListRoleAliases",
                "iot:ListStreams",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroups",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:ListThingRegistrationTaskReports",
                "iot:ListThingRegistrationTasks",
                "iot:ListThings",
```

```
                "iot:ListThingsInThingGroup",
                "iot:ListThingTypes",
                "iot:ListTopicRules",
                "iot:ListV2LoggingLevels",
                "iot:SearchIndex",
                "iot:TestAuthorization",
                "iot:TestInvokeAuthorizer",
                "iot:DescribeAccountAuditConfiguration",
                "iot:DescribeAuditTask",
                "iot:ListAuditTasks",
                "iot:DescribeScheduledAudit",
                "iot:ListScheduledAudits",
                "iot:ListAuditFindings",
                "iot:DescribeSecurityProfile",
                "iot:ListSecurityProfiles",
                "iot:ListSecurityProfilesForTarget",
                "iot:ListTargetsForSecurityProfile",
                "iot:ListActiveViolations",
                "iot:ListViolationEvents",
                "iot:ValidateSecurityProfileBehaviors"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS politica gestita: AWSIo TData Accesso

È possibile allegare la policy `AWSIoTDataAccess` alle identità IAM.

Questa politica concede le autorizzazioni di identità associate che consentono l'accesso a tutte le operazioni AWS IoT sui dati. Le operazioni sui dati inviano dati tramite il protocollo MQTT o HTTP. Per visualizzare questa policy nella Console di gestione AWS, consulta `AWSIoTDataAccess`.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `iot`— Recupera AWS IoT i dati e consenti l'accesso completo alle AWS IoT azioni di messaggistica.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect",
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow",
                "iot:ListNamedShadowsForThing"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS politica gestita: accesso AWSIo TFull

È possibile allegare la policy `AWSIoTFullAccess` alle identità IAM.

Questa policy concede all'identità associata le autorizzazioni per l'accesso a tutte le operazioni di configurazione e messaggistica AWS IoT . Per visualizzare questa politica in Console di gestione AWS, vedere `AWSIoTFullAccess`.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `iot`— Recupera AWS IoT i dati e consenti l'accesso completo alle azioni AWS IoT di configurazione e messaggistica.
- `iotjobsdata`— Recupera i dati AWS IoT di Jobs e consenti l'accesso completo alle operazioni dell'API del piano dati AWS IoT di Jobs.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:*",
                "iotjobsdata:*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS politica gestita: AWSIo TLogging

È possibile allegare la policy `AWSIoTLogging` alle identità IAM.

Questa politica concede le autorizzazioni di identità associate che consentono l'accesso alla creazione di gruppi Amazon CloudWatch Logs e ai log di streaming verso i gruppi. Questa politica è allegata al tuo ruolo di registrazione. CloudWatch Per visualizzare questa politica in Console di gestione AWS, vedere `AWSIoTLogging`.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `logs`— Recupera i CloudWatch log. Consente inoltre la creazione di gruppi di CloudWatch log e di log di streaming verso i gruppi.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DeleteLogStream"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS politica gestita: AWSIo TOTAUpdate

È possibile allegare la policy `AWSIoTOTAUpdate` alle identità IAM.

Questa politica concede le autorizzazioni di identità associate che consentono l'accesso alla creazione di AWS IoT lavori, ai lavori di firma AWS IoT del codice e alla descrizione dei lavori di firma AWS del codice. Per visualizzare questa politica in, vedere. Console di gestione `AWSAWSIoTOTAUpdate`

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `iot`— Crea AWS IoT offerte di lavoro e lavori di firma del codice.
- `signer`— Esegui la creazione di lavori di firma del AWS codice.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iot:CreateJob",
            "signer:DescribeSigningJob"
        ],
        "Resource": "*"
    }
}
```

AWS politica gestita: AWSIo TRule azioni

È possibile allegare la policy `AWSIoTRuleActions` alle identità IAM.

Questa politica concede le autorizzazioni di identità associate che consentono l'accesso a tutti Servizio AWS i messaggi supportati nelle azioni delle AWS IoT regole. Per visualizzare questa politica in Console di gestione AWS, vedere. `AWSIoTRuleActions`

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `iot`: esegue operazioni per la pubblicazione di messaggi delle operazioni delle regole.
- `dynamodb`: inserisce un messaggio in una tabella DynamoDB o suddivide un messaggio in più colonne di una tabella DynamoDB.

- `s3`: archivia un oggetto in un bucket Amazon S3.
- `kinesis`: invia un messaggio a un oggetto di flusso Amazon Kinesis.
- `firehose`- Inserire un record in un oggetto stream Firehose.
- `cloudwatch`- Modifica lo stato CloudWatch dell'allarme o invia i dati del messaggio a CloudWatch metric.
- `sns`: esegue l'operazione di pubblicazione di una notifica con Amazon SNS. Questa operazione è limitata agli argomenti AWS IoT SNS.
- `sqs`: inserisce un messaggio da aggiungere alla coda SQS.
- `es`- Invia un messaggio al OpenSearch servizio di assistenza.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "dynamodb:PutItem",
            "kinesis:PutRecord",
            "iot:Publish",
            "s3:PutObject",
            "sns:Publish",
            "sqs:SendMessage*",
            "cloudwatch:SetAlarmState",
            "cloudwatch:PutMetricData",
            "es:ESHttpPut",
            "firehose:PutRecord"
        ],
        "Resource": "*"
    }
}
```

AWS politica gestita: AWSIo TThings registrazione

È possibile allegare la policy `AWSIoTThingsRegistration` alle identità IAM.

Questa policy concede all'identità associata le autorizzazioni per l'accesso per registrare oggetti in blocco usando l'API `StartThingRegistrationTask`. Questa policy può influenzare l'elaborazione e lo storage dei dati. Per visualizzare questa politica nel Console di gestione AWS, vedere `AWSIoTThingsRegistration`.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `iot`: esegue operazioni per la creazione di oggetti e il collegamento di policy e certificati durante la registrazione in blocco.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AddThingToThingGroup",
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
                "iot:CreateCertificateFromCsr",
                "iot:CreatePolicy",
                "iot:CreateThing",
                "iot:DescribeCertificate",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingType",
                "iot:DetachPolicy",
                "iot:DetachThingPrincipal",
                "iot:GetPolicy",
                "iot:ListAttachedPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroupsForThing",
```

```
                "iot:ListThingPrincipals",
                "iot:RegisterCertificate",
                "iot:RegisterThing",
                "iot:RemoveThingFromThingGroup",
                "iot:UpdateCertificate",
                "iot:UpdateThing",
                "iot:UpdateThingGroupsForThing",
                "iot:AddThingToBillingGroup",
                "iot:DescribeBillingGroup",
                "iot:RemoveThingFromBillingGroup"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS IoT aggiornamenti alle politiche AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite AWS IoT da quando questo servizio ha iniziato a tenere traccia di queste modifiche. Per ricevere avvisi automatici sulle modifiche a questa pagina, iscriviti al feed RSS nella pagina della cronologia dei AWS IoT documenti.

Modifica	Descrizione	Data
AWSIoTFullAccesso: aggiornamento a una politica esistente	AWS IoT ha aggiunto nuove autorizzazioni per consentire agli utenti di accedere alle operazioni dell'API del piano dati di AWS IoT Jobs utilizzando il protocollo HTTP. Un nuovo prefisso della policy IAM,`iotjobsdata:` , offre un controllo di accesso più dettagliato per accedere agli	11 maggio 2022

Modifica	Descrizione	Data
	endpoint del piano dati di AWS IoT Jobs. Per le operazioni API piano di controllo (control-plane), viene usato ancora il prefisso `iot:`. Per ulteriori informazioni, consulta AWS IoT Core politiche per il protocollo HTTPS.	
AWS IoT ha iniziato a tenere traccia delle modifiche	AWS IoT ha iniziato a tenere traccia delle modifiche per le sue politiche AWS gestite.	11 maggio 2022

Risoluzione dei problemi di AWS IoT identità e accesso

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con un AWS IoT IAM.

Argomenti

- Non sono autorizzato a eseguire alcuna azione in AWS IoT
- Non sono autorizzato a eseguire iam: PassRole
- Voglio consentire a persone esterne a me di accedere Account AWS alle mie AWS IoT risorse

Non sono autorizzato a eseguire alcuna azione in AWS IoT

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM, `mateojackson`, tenta di utilizzare la console per visualizzare i dettagli relativi a una risorsa oggetto, ma non dispone delle autorizzazioni `iot:DescribeThing`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iot:DescribeThing on resource: MyIoTThing
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa oggetto utilizzando l'azione `iot:DescribeThing`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Utilizzo di AWS IoT Device Advisor

Se utilizzi AWS IoT Device Advisor, l'errore di esempio seguente si verifica quando l'utente `mateojackson` tenta di utilizzare la console per visualizzare i dettagli sulla definizione di una suite ma non dispone delle `iotdeviceadvisor:GetSuiteDefinition` autorizzazioni.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotdeviceadvisor:GetSuiteDefinition on resource: MySuiteDefinition
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa `MySuiteDefinition` utilizzando l'azione `iotdeviceadvisor:GetSuiteDefinition`.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo a AWS IoT.

Alcuni Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in AWS IoT. Tuttavia, l'operazione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne a me di accedere Account AWS alle mie AWS IoT risorse

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per concedere alle persone l'accesso alle tue risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se AWS IoT supporta queste funzionalità, consulta. Come AWS IoT funziona con IAM
- Per scoprire come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.

Registrazione e monitoraggio

Il monitoraggio è un elemento importante per mantenere l'affidabilità, la disponibilità e le prestazioni delle AWS IoT vostre AWS soluzioni. È necessario raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica. Per informazioni sulle procedure di registrazione e monitoraggio, consulta Monitoraggio AWS IoT

Strumenti di monitoraggio

AWS fornisce strumenti che è possibile utilizzare per il monitoraggio AWS IoT. Alcuni di questi strumenti possono essere configurati per il monitoraggio automatico delle applicazioni. Alcuni degli

strumenti richiedono l'intervento manuale. Si consiglia di automatizzare il più possibile i processi di monitoraggio.

Strumenti di monitoraggio automatici

Puoi utilizzare i seguenti strumenti di monitoraggio automatizzato per osservare AWS IoT e segnalare quando qualcosa non va:

- Amazon CloudWatch Alarms: monitora una singola metrica in un periodo di tempo specificato ed esegui una o più azioni in base al valore della metrica rispetto a una determinata soglia in diversi periodi di tempo. L'azione è una notifica inviata a un argomento di Amazon Simple Notification Service (Amazon SNS) o a una policy di Amazon EC2 Auto Scaling. CloudWatch gli allarmi non richiamano azioni semplicemente perché si trovano in uno stato particolare. Lo stato deve essere cambiato e restare costante per un numero specificato di periodi. Per ulteriori informazioni, consulta Monitora AWS IoT allarmi e metriche con Amazon CloudWatch.
- Amazon CloudWatch Logs: monitora, archivia e accedi ai tuoi file di registro da AWS CloudTrail o altre fonti. Amazon CloudWatch Logs ti consente anche di visualizzare i passaggi critici eseguiti dai test case di AWS IoT Device Advisor, gli eventi generati e i messaggi MQTT inviati dai tuoi dispositivi o AWS IoT Core durante l'esecuzione del test. Questi registri consentono di eseguire il debug e intraprendere operazioni correttive sui dispositivi. Per ulteriori informazioni, consulta Monitora AWS IoT utilizzando i log CloudWatch Per ulteriori informazioni sull'uso di Amazon CloudWatch, consulta Monitoring Log Files nella Amazon CloudWatch User Guide.
- Amazon CloudWatch Events: abbina gli eventi e li indirizza a una o più funzioni o stream di destinazione per apportare modifiche, acquisire informazioni sullo stato e intraprendere azioni correttive. Per ulteriori informazioni, consulta What Is Amazon CloudWatch Events nella Amazon CloudWatch User Guide.
- AWS CloudTrail Monitoraggio dei log: condividi i file di CloudTrail registro tra account, monitora i file di registro in tempo reale inviandoli a CloudWatch Logs, scrivi applicazioni di elaborazione dei log in Java e verifica che i file di registro non siano cambiati dopo la consegna da parte di. CloudTrail Per ulteriori informazioni, vedere Registrazione delle chiamate AWS IoT API utilizzando AWS CloudTrail e anche Lavorare con i file di CloudTrail registro nella Guida per l'AWS CloudTrail utente.

Strumenti di monitoraggio manuali

Un'altra parte importante del monitoraggio AWS IoT consiste nel monitorare manualmente gli elementi che gli CloudWatch allarmi non coprono. Le dashboard della console di AWS servizio AWS

IoT CloudWatch, e altre, forniscono una at-a-glance panoramica dello stato dell'ambiente AWS . Ti consigliamo di controllare anche i file di registro. AWS IoT

- AWS IoT il pannello di controllo mostra:
  - Certificati CA
  - Certificati
  - Policy
  - Regole
  - Oggetti
- CloudWatch la home page mostra:
  - Lo stato e gli allarmi attuali.
  - I grafici degli allarmi e delle risorse.
  - Lo stato dei servizi.

  Puoi usare CloudWatch per fare quanto segue:
  - Creare pannelli di controllo personalizzati per monitorare i servizi di interesse.
  - Creare grafici dei dati dei parametri per la risoluzione di problemi e il rilevamento di tendenze.
  - Cerca e sfoglia tutte le metriche AWS delle tue risorse.
  - Crea e modifica gli allarmi per ricevere le notifiche dei problemi.

Convalida della conformità per Core AWS IoT

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. Per ulteriori informazioni sulla responsabilità di conformità durante l'utilizzo Servizi AWS, consulta la Documentazione AWS sulla sicurezza.

Resilienza in AWS IoT Core

L'infrastruttura AWS globale è costruita attorno a Regione AWS s e alle zone di disponibilità. Regione AWS Le s forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Per ulteriori informazioni su Regione AWS s e sulle zone di disponibilità, consulta Global Infrastructure.AWS

AWS IoT Core memorizza le informazioni sui dispositivi nel registro dei dispositivi. Memorizza inoltre i certificati CA, i certificati dei dispositivi e i dati Device Shadow. In caso di errori hardware o di rete, questi dati vengono replicati automaticamente attraverso le zone di disponibilità ma non attraverso le regioni.

AWS IoT Core pubblica eventi MQTT quando il registro del dispositivo viene aggiornato. È possibile utilizzare questi messaggi per eseguire il backup dei dati del registro di sistema e salvarli in una ubicazione, ad esempio in una tabella Dynamo DB. L'utente è responsabile del salvataggio dei certificati AWS IoT Core creati per sé o di quelli creati dall'utente. Device Shadow memorizza i dati sullo stato dei dispositivi e può essere inviato nuovamente quando un dispositivo torna online. AWS IoT Device Advisor memorizza le informazioni sulla configurazione della suite di test. Questi dati vengono replicati automaticamente in caso di errori hardware o di rete.

AWS IoT Core le risorse sono specifiche della regione e non vengono replicate a Regioni AWS meno che l'utente non lo faccia specificamente.

Per ulteriori informazioni sulle best practice di Sicurezza, consulta Le migliori pratiche di sicurezza in AWS IoT Core.

Utilizzo AWS IoT Core con gli endpoint VPC dell'interfaccia

Con AWS IoT Core, puoi creare endpoint del piano di controllo IoT e endpoint di dati IoT all'interno del tuo cloud privato virtuale (VPC) utilizzando gli endpoint VPC di interfaccia. Gli endpoint VPC di interfaccia sono alimentati da AWS PrivateLink, una AWS tecnologia che è possibile utilizzare per accedere ai servizi in esecuzione AWS utilizzando indirizzi IP privati. Per ulteriori informazioni, consulta Amazon Virtual Private Cloud.

Per connettere dispositivi sul campo su reti remote, ad esempio una rete aziendale, al tuo Amazon VPC, fai riferimento alle opzioni elencate nella matrice di connettività Network-to-Amazon VPC.

Indice

- Creazione di endpoint VPC per il piano di controllo AWS IoT Core
- Creazione di endpoint VPC per il piano dati AWS IoT Core
- Creazione di endpoint VPC per provider di credenziali AWS IoT Core
- Creazione di un endpoint di interfaccia Amazon VPC
- Configura una zona ospitata privata
- Controllo dell'accesso agli AWS IoT Core endpoint tramite VPC
- Limitazioni
- Scalabilità degli endpoint VPC con AWS IoT Core
- Utilizzo di domini personalizzati con endpoint VPC
- Disponibilità di endpoint VPC per AWS IoT Core
- Utilizzo del tunneling AWS IoT Device Management sicuro con endpoint VPC di interfaccia

Creazione di endpoint VPC per il piano di controllo AWS IoT Core

Puoi creare un endpoint VPC per l'API del piano di AWS IoT Core controllo per connettere i tuoi dispositivi a AWS IoT servizi e altri servizi. AWS Per iniziare a usare gli endpoint VPC, crea un endpoint VPC di interfaccia e selezionalo come servizio. AWS IoT Core AWS Se utilizzi la CLI, chiama innanzitutto describe-vpc-endpoint-servicesper assicurarti di scegliere una zona di disponibilità in cui AWS IoT Core è presente nel tuo particolare. Regione AWS Ad esempio, in us-east-1, questo comando ha il seguente aspetto:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.api
```

Consulta le istruzioni dettagliate riportate di seguito per creare un endpoint di interfaccia Amazon VPC per AWS IoT Core il piano di controllo.

Creazione di endpoint VPC per il piano dati AWS IoT Core

Puoi creare un endpoint VPC per l'API del piano AWS IoT Core dati per connettere i tuoi dispositivi a AWS IoT servizi e altri servizi. AWS Per iniziare a usare gli endpoint VPC, crea un endpoint VPC di interfaccia e selezionalo come servizio. AWS IoT Core AWS Se utilizzi la CLI, chiama innanzitutto

describe-vpc-endpoint-servicesper assicurarti di scegliere una zona di disponibilità in cui AWS IoT Core è presente nel tuo particolare. Regione AWS Ad esempio, in us-east-1, questo comando ha il seguente aspetto:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.data
```

> **Note**
>
> La funzione VPC per la creazione automatica di un record DNS è disattivata. Per unirsi a questi endpoint, è necessario creare manualmente un registro DNS privato. Per ulteriori informazioni sui registri DNS VPC privati, consulta DNS privato per gli endpoint di interfaccia. Per ulteriori informazioni sulle limitazioni del AWS IoT Core VPC, consulta. Limitazioni

Per connettere i client MQTT alle interfacce degli endpoint VPC:

- È necessario creare manualmente i record DNS in una zona ospitata privata collegata al VPC. Per iniziare, consulta Creazione di una zona ospitata privata.

- All'interno della tua zona ospitata privata, crea un record di alias per ogni IP di interfaccia di rete elastica per l'endpoint VPC. Se disponi di più interfacce di rete IPs per più endpoint VPC, crea record DNS ponderati con pesi uguali su tutti i record ponderati. Questi indirizzi IP sono disponibili dalla chiamata DescribeNetworkInterfacesAPI se filtrati in base all'ID dell'endpoint VPC nel campo della descrizione.

Consulta le istruzioni dettagliate riportate di seguito per creare un endpoint di interfaccia Amazon VPC e configurare una zona ospitata privata per AWS IoT Core il piano dati.

Creazione di endpoint VPC per provider di credenziali AWS IoT Core

È possibile creare un endpoint VPC per consentire al provider di AWS IoT Core credenziali di connettere i dispositivi utilizzando l'autenticazione basata su certificati client e ottenere credenziali temporanee AWS in formato Signature Version 4.AWS Per iniziare a usare gli endpoint VPC per il provider di AWS IoT Core credenziali, esegui il comando create-vpc-endpointCLI per creare un endpoint VPC di interfaccia e seleziona il provider di credenziali come servizio. AWS IoT Core AWS Per assicurarti di scegliere una zona di disponibilità in cui AWS IoT Core è presente il tuo particolare

Regione AWS, esegui prima il comando. describe-vpc-endpoint-services Ad esempio, in us-east-1, questo comando ha il seguente aspetto:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-
east-1.iot.credentials
```

> **Note**
>
> La funzione VPC per la creazione automatica di un record DNS è disattivata. Per unirsi a questi endpoint, è necessario creare manualmente un registro DNS privato. Per ulteriori informazioni sui registri DNS VPC privati, consulta DNS privato per gli endpoint di interfaccia. Per ulteriori informazioni sulle limitazioni del AWS IoT Core VPC, consulta. Limitazioni

Per connettere i client HTTP alle interfacce degli endpoint VPC:

- È necessario creare manualmente i record DNS in una zona ospitata privata collegata al VPC. Per iniziare, consulta Creazione di una zona ospitata privata.

- All'interno della tua zona ospitata privata, crea un record di alias per ogni IP di interfaccia di rete elastica per l'endpoint VPC. Se disponi di più interfacce di rete IPs per più endpoint VPC, crea record DNS ponderati con pesi uguali su tutti i record ponderati. Questi indirizzi IP sono disponibili dalla chiamata DescribeNetworkInterfacesAPI se filtrati in base all'ID dell'endpoint VPC nel campo della descrizione.

Consulta le istruzioni dettagliate riportate di seguito per creare un endpoint di interfaccia Amazon VPC e configurare una zona ospitata privata per AWS IoT Core il fornitore di credenziali.

Creazione di un endpoint di interfaccia Amazon VPC

È possibile creare un endpoint VPC di interfaccia per connettersi ai AWS servizi forniti da. AWS PrivateLink Utilizzare la procedura seguente per creare un endpoint VPC di interfaccia che si connette al piano AWS IoT Core dati o AWS IoT Core al provider di credenziali. Per ulteriori informazioni, consulta Accedere a un AWS servizio utilizzando un endpoint VPC di interfaccia.

> **Note**
>
> I processi per creare un endpoint di interfaccia Amazon VPC per il piano AWS IoT Core dati e il provider di AWS IoT Core credenziali sono simili, ma è necessario apportare modifiche specifiche all'endpoint per far funzionare la connessione.

Per creare un endpoint VPC di interfaccia utilizzando la console VPC Endpoints

1. Vai alla console VPC Endpoints, sotto Virtual private cloud nel menu a sinistra, scegli Endpoints quindi Crea endpoint.
2. Nella pagina Crea endpoint, specifica le seguenti informazioni.
   - Scegli Servizio AWS s per Categoria di servizio.
   - Per Service Name (Nome servizio), esegui la ricerca inserendo la parola chiave `iot`. Nell'elenco dei `iot` servizi visualizzati, scegli l'endpoint.

     Se crei un endpoint VPC per il piano di AWS IoT Core controllo, scegli l'endpoint API del piano AWS IoT Core di controllo adatto al tuo. Regione AWS L'endpoint sarà del formato `com.amazonaws.`*`region`*`.iot.api`.

     Se crei un endpoint VPC per il piano AWS IoT Core dati, scegli l'endpoint API del piano AWS IoT Core dati per la tua regione. L'endpoint sarà del formato `com.amazonaws.`*`region`*`.iot.data`.

     Se crei un endpoint VPC per un provider di AWS IoT Core credenziali, scegli l'endpoint del provider di AWS IoT Core credenziali per la tua regione. L'endpoint sarà del formato `com.amazonaws.`*`region`*`.iot.credentials`.

     Se crei un endpoint VPC per le regioni FIPS (Federal Information Processing Standard), scegli l'endpoint API FIPS adatto alle tue esigenze. Regione AWS L'endpoint sarà del formato `com.amazonaws.`*`region`*`.iot-fips.api`. Questo è solo per il piano di controllo.

     > **Note**
     >
     > Il nome del servizio per il piano AWS IoT Core dati nella regione della Cina sarà nel formato`cn.com.amazonaws.`*`region`*`.iot.data`. Il nome del

servizio per l'aereo di AWS IoT Core controllo nella regione della Cina sarà nel formato`com.amazonaws.region.iot.api`.

- Per VPC e sottoreti, scegli il VPC in cui desideri creare l'endpoint e le zone di disponibilità (AZs) in cui desideri creare la rete di endpoint.
- Per Abilita il nome DNS, assicurati che Enable for this endpoint non sia selezionato per il piano dati e il provider di credenziali. AWS IoT Core AWS IoT Core Né il piano AWS IoT Core dati né il provider di AWS IoT Core credenziali supportano ancora i nomi DNS privati.

  Per impostazione AWS IoT Core predefinita, per il piano di controllo, è selezionata l'opzione Abilita nome DNS. Ciò garantisce che tutte le richieste agli endpoint pubblici del piano di AWS IoT Core controllo vengano indirizzate invece attraverso gli endpoint VPC. Quando questa opzione è abilitata, non è necessario configurare una zona ospitata privata.
- In Security group (Gruppo di sicurezza), scegli i gruppi di sicurezza da associare alle interfacce di rete dell'endpoint.
- Facoltativamente, puoi aggiungere o rimuovere i tag. I tag sono coppie nome-valore utilizzate per l'associazione al tuo endpoint.

3. Per creare l'endpoint VPC, scegli Create endpoint (Crea endpoint).

Dopo aver creato l' AWS PrivateLink endpoint, nella scheda Dettagli dell'endpoint vedrai un elenco di nomi DNS. Puoi utilizzare uno di questi nomi DNS che hai creato in questa sezione per configurare la tua zona ospitata privata. Se si utilizza AWS IoT Core il piano di controllo, non è necessario configurare una zona ospitata privata.

Configura una zona ospitata privata

> **Note**
>
> Se si utilizza AWS IoT Core il piano di controllo e l'opzione Abilita nome DNS è selezionata, non è necessario configurare una zona ospitata privata. Se la disabiliti, devi seguire questa procedura per configurare una zona ospitata privata.

Puoi utilizzare uno di questi nomi DNS creati nella sezione precedente per configurare la tua zona ospitata privata.

Per il piano AWS IoT Core dati

Il nome DNS deve essere il nome di configurazione del dominio o l'`IoT:Data-ATS`endpoint. Un nome DNS di esempio può essere:. *`xxx`*`-ats.data.iot.`*`region`*`.amazonaws.com`

Per fornitore di AWS IoT Core credenziali

Il nome DNS deve essere il tuo `iot:CredentialProvider` endpoint. Un nome DNS di esempio può essere:. *`xxxx`*`.credentials.iot.`*`region`*`.amazonaws.com`

Per il piano AWS IoT Core di controllo

Il nome DNS deve essere l'endpoint AWS IoT Core del piano di controllo. Un esempio di nome DNS per il piano di AWS IoT Core controllo è. *`xxxx`*`.api.iot.`*`region`*`.amazonaws.com`

> **Note**
>
> I processi per configurare la zona ospitata privata per il piano AWS IoT Core dati e il provider di AWS IoT Core credenziali sono simili, ma è necessario apportare modifiche specifiche all'endpoint per far funzionare la connessione.

Crea una zona ospitata privata

Per creare una zona ospitata privata utilizzando la console Route 53

1. Passa alla console Route 53 Hosted zone (Zone ospitate) e scegli Create hosted zone (Crea una zona ospitata).
2. Nella pagina Create hosted zone (Crea una zona ospitata), specifica le informazioni riportate di seguito.
   - Per Nome di dominio, inserisci l'indirizzo dell'endpoint del tuo `iot:Data-ATS` o dell'`iot:CredentialProvider`endpoint. Il seguente comando AWS CLI mostra come ottenere l'endpoint tramite una rete pubblica:`aws iot describe-endpoint --endpoint-type iot:Data-ATS`, o. `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`

     > **Note**
     >
     > Se utilizzi domini personalizzati, consulta Utilizzo di domini personalizzati con endpoint VPC. I domini personalizzati non sono supportati per AWS IoT Core il provider di credenziali.

- Nell'elenco Type (Tipo), scegli Private Hosted Zone (Zona ospitata privata).
- Facoltativamente, puoi aggiungere o rimuovere tag da associare alla tua zona ospitata.

3. Per creare la tua zona privata ospitata, scegli Create hosted zone (Crea una zona ospitata).

Per ulteriori informazioni consulta Creating a private hosted zone (Creazione di una zona ospitata privata).

Creazione di un record

Dopo aver creato una zona ospitata privata, è possibile creare un registro che indica al DNS come si desidera che il traffico venga instradato a quel dominio.

Per creare un record

1. Nell'elenco delle zone ospitate visualizzate, scegli la zona ospitata privata creata in precedenza e scegli Create record (Crea un record).
2. Utilizza il metodo della procedura guidata per creare il record. Se la console presenta il metodo Quick create (Creazione rapida), scegli Switch to wizard (Passa alla procedura guidata).
3. Scegli Simple Routing (Instradamento semplice) per Routing policy (Policy di instradamento) e poi Next (Successivo).
4. Nella pagina Configure records (Configura record), scegli Define simple record (Definisci record semplice).
5. Nella pagina Define simple record (Definisci record semplice):
   - Per Nome del record, inserisci `iot:Data-ATS` endpoint o `iot:CredentialProvider` endpoint. Deve essere lo stesso nome della zona ospitata privata.
   - Per Tipo di record, se desideri solo il IPv4 supporto, mantieni il valore uguale. `A - Routes traffic to an IPv4 address and some AWS resources` Se desideri solo IPv6 supporto, mantieni il valore ugualeAAAA `- Routes traffic to an IPv6 address and some AWS resources`. Se desideri il supporto dual-stack (entrambi IPv4 e IPv6), crea due record (e AAAA nella zona ospitata) con lo stesso nome del record **A** e lo stesso valore/indirizzamento del traffico verso.
   - In Value/Route traffic to (Valore/Instradamento del traffico a), seleziona Alias to VPC endpoint (Alias all'endpoint VPC). Scegli la tua Regione quindi scegli l'endpoint creato in precedenza, come descritto in ??? dall'elenco degli endpoint visualizzati.
6. Scegli Define simple record (Definizione del record semplice) per creare il record.

Controllo dell'accesso agli AWS IoT Core endpoint tramite VPC

Puoi limitare l'accesso AWS IoT Core ai dispositivi in modo che sia consentito solo tramite l'endpoint VPC utilizzando le chiavi contestuali delle condizioni VPC. AWS IoT Core supporta le seguenti chiavi di contesto relative al VPC:

- SourceVpc
- SourceVpce
- VPCSourceIp

> **Note**
>
> AWS IoT Core non supporta le policy degli endpoint per gli endpoint VPC.

Ad esempio, la seguente politica concede l'autorizzazione a connettersi AWS IoT Core utilizzando un ID client che corrisponde al nome dell'oggetto e a pubblicare su qualsiasi argomento preceduto dal nome dell'oggetto, a condizione che il dispositivo si connetta a un endpoint VPC con un ID endpoint VPC particolare. Questa policy negherebbe i tentativi di connessione all'endpoint dati IoT pubblico.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
            }
        }

        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}/*"
            ]
        }
    ]
}
```

Limitazioni

Gli endpoint VPC sono attualmente supportati per gli endpoint del piano di AWS IoT Core controllo, gli endpoint di AWS IoT Core dati e gli endpointdei fornitori di credenziali.AWS IoT Core Gli endpoint VPC sono supportati solo per gli endpoint FIPS (Federal Information Processing Standard) quando si utilizza il piano di controllo. AWS IoT Core

Limitazioni degli endpoint VPC del piano di controllo IoT

Questa sezione copre i limiti degli endpoint VPC del piano di controllo IoT.

- Gli endpoint VPC serviranno solo certificati ATS.
- I domini personalizzati non sono supportati per gli endpoint del piano di controllo.
- Per informazioni sulle politiche di sicurezza FIPS, vedere Politiche di sicurezza FIPS.

Limitazioni degli endpoint VPC per dati IoT

Questa sezione descrive i limiti degli endpoint VPC di dati IoT.

- I periodi keep-alive di MQTT i sono limitati a 230 secondi. I periodi di mantenimento in vita più lunghi verranno automaticamente ridotti a 230 secondi.
- Ogni endpoint VPC supporta un totale di 100.000 dispositivi collegati. Se hai bisogno di più connessioni vediScalabilità degli endpoint VPC con AWS IoT Core.
- Gli endpoint VPC serviranno solo Certificati ATS, ad eccezione dei domini personalizzati.
- Le policy degli endpoint VPC non sono supportate.

- Per gli endpoint VPC creati per il piano AWS IoT Core dati, AWS IoT Core non supporta l'utilizzo di record DNS pubblici zonali o regionali.

Limitazioni degli endpoint dei provider di credenziali

Questa sezione descrive i limiti degli endpoint VPC del provider di credenziali.

- Gli endpoint VPC serviranno solo certificati ATS.
- Le policy degli endpoint VPC non sono supportate.
- I domini personalizzati non sono supportati per gli endpoint dei provider di credenziali.
- Per gli endpoint VPC creati per il provider di AWS IoT Core credenziali, AWS IoT Core non supporta l'utilizzo di record DNS pubblici zonali o regionali.

Scalabilità degli endpoint VPC con AWS IoT Core

AWS IoT Core Gli endpoint VPC di interfaccia sono limitati a 100.000 dispositivi connessi su un singolo endpoint di interfaccia. Se il tuo caso d'uso richiede più connessioni simultanee al broker, ti consigliamo di utilizzare più endpoint VPC e di instradare manualmente i tuoi dispositivi attraverso gli endpoint dell'interfaccia. Quando crei registri DNS privati per instradare il traffico verso gli endpoint VPC, assicurati di creare tutti i registri ponderati quanti sono gli endpoint VPC per distribuire il traffico su più endpoint.

Utilizzo di domini personalizzati con endpoint VPC

Se desideri utilizzare domini personalizzati con endpoint VPC, devi creare i record dei nomi di dominio personalizzati in una zona ospitata privata e creare record di routing in Route53. Per ulteriori informazioni, consulta Creazione di una zona ospitata privata.

> **Note**
>
> I domini personalizzati sono supportati solo per gli endpoint di AWS IoT Core dati.

Disponibilità di endpoint VPC per AWS IoT Core

AWS IoT Core Gli endpoint VPC dell'interfaccia sono disponibili in tutte AWS IoT Core le regioni supportate. AWS IoT Core Gli endpoint VPC dell'interfaccia per il provider di AWS IoT Core credenziali non sono supportati nella regione Cina e. AWS GovCloud (US) Regions

Utilizzo del tunneling AWS IoT Device Management sicuro con endpoint VPC di interfaccia

AWS IoT Device Management il tunneling sicuro supporta gli endpoint VPC di interfaccia. Puoi utilizzare gli endpoint VPC per mantenere il traffico tra il tuo VPC e AWS IoT Secure Tunneling all'interno della AWS rete, senza richiedere un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione. AWS Direct Connect

Gli endpoint VPC di interfaccia sono alimentati da AWS PrivateLink, una tecnologia che consente di accedere in modo privato ai servizi utilizzando indirizzi IP privati. Per ulteriori informazioni, consulta Accedere a un AWS servizio utilizzando un endpoint VPC di interfaccia nella Guida. AWS PrivateLink

Indice

- Prerequisiti
- Ricezione di notifiche del tunnel tramite endpoint VPC
- Creazione di endpoint VPC per un tunneling sicuro
- Configurazione delle policy degli endpoint VPC sul server proxy
- Fasi successive

Prerequisiti

Prima di creare endpoint VPC per AWS IoT Secure Tunneling, verifica di disporre di quanto segue:

- Un AWS account con le autorizzazioni necessarie per creare endpoint VPC.
- Un VPC nel tuo AWS account.
- Comprensione dei concetti di tunneling AWS IoT Device Management sicuro.
- Familiarità con le policy AWS Identity and Access Management degli endpoint VPC e (IAM)

Ricezione di notifiche del tunnel tramite endpoint VPC

Per ricevere le notifiche del tunnel tramite un endpoint VPC, i dispositivi possono connettersi al piano AWS IoT Core dati tramite un endpoint VPC e sottoscrivere l'argomento MQTT riservato al tunneling sicuro.

Per istruzioni su come creare e configurare un endpoint VPC nel piano AWS IoT Core dati, consulta Using with AWS IoT Core interface VPC endpoint nella Developer Guide. AWS IoT

Creazione di endpoint VPC per un tunneling sicuro

È possibile creare endpoint VPC sia per il piano di controllo del tunneling sicuro che per il server proxy.

Per creare un endpoint VPC per un tunneling sicuro

1. Segui i passaggi descritti nella sezione Creazione di un endpoint di interfaccia nella Amazon VPC Developer Guide
2. Per Service name, scegli una delle seguenti opzioni in base al tipo di endpoint:

   Piano di controllo

   - Standard: `com.amazonaws.<region>.iot.tunneling.api`
   - FIPS (disponibile nelle regioni FIPS): `com.amazonaws.<region>.iot-fips.tunneling.api`

   Server proxy

   - Standard: `com.amazonaws.<region>.iot.tunneling.data`
   - FIPS (disponibile nelle regioni FIPS): `com.amazonaws.<region>.iot-fips.tunneling.data`

   Sostituisci *`<region>`* con il tuo. Regione AWS Ad esempio, `us-east-1`.
3. Completa i passaggi rimanenti del processo di creazione degli endpoint VPC in base ai requisiti di rete.

Configurazione delle policy degli endpoint VPC sul server proxy

Oltre all'autorizzazione basata su token di accesso client utilizzata per autorizzare le connessioni ai tunnel, puoi utilizzare le policy degli endpoint VPC per limitare ulteriormente il modo in cui i dispositivi possono utilizzare un endpoint VPC per connettersi al Secure Tunneling Proxy Server. Le policy degli endpoint VPC seguono una sintassi simile a quella di IAM e sono configurate sull'endpoint VPC stesso.

Tieni presente che l'unica azione IAM supportata per le policy degli endpoint VPC del server proxy è. `iot:ConnectToTunnel`

Di seguito sono riportati alcuni esempi di diverse policy per gli endpoint VPC.

Esempi di policy relative agli endpoint VPC del server proxy

Gli esempi seguenti mostrano le configurazioni delle policy degli endpoint VPC di Proxy Server per casi d'uso comuni.

Example- Politica predefinita

Questa policy consente ai dispositivi all'interno del tuo VPC di connettersi a qualsiasi tunnel nello stesso punto in Regione AWS cui viene creato l'endpoint, su qualsiasi account. AWS

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

Example- Limita l'accesso a account specifici AWS

Questa policy consente all'endpoint VPC di connettersi solo ai tunnel di account specifici. AWS

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
```

```
            "Action": "iot:ConnectToTunnel",
            "Resource": [
                "arn:aws:iot:us-east-1:111122223333:tunnel/*",
                "arn:aws:iot:us-east-1:444455556666:tunnel/*"
            ]
        }
    ]
}
```

Example- Limita le connessioni in base all'endpoint del tunnel

Puoi limitare l'accesso agli endpoint VPC per consentire solo ai dispositivi di connettersi all'estremità di origine o destinazione di un tunnel.

Solo origine:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iot:ClientMode": "source"
                }
            }
        }
    ]
}
```

Solo destinazione:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "iot:ClientMode": "destination"
                }
            }
        }
    ]
}
```

Example- Limita l'accesso in base ai tag delle risorse

Questa policy consente all'endpoint VPC di connettersi solo ai tunnel etichettati con una coppia chiave-valore specifica.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Environment": "Production"
                }
            }
        }
    ]
}
```

Example- Condizioni politiche combinate

Questa politica dimostra la combinazione di più elementi politici. Consente le connessioni a qualsiasi tunnel in un AWS account specifico, ma solo se il tunnel è etichettato con `AllowConnectionsThroughPrivateLink` set to `true` e il client non si connette all'estremità di destinazione del tunnel.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:111122223333:tunnel/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AllowConnectionsThroughPrivateLink": "true"
                }
            }
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": [
                "arn:aws:iot:us-east-1:111122223333:tunnel/*"
            ],
            "Condition": {
                "StringEquals": {
                    "iot:ClientMode": "destination"
                }
            }
        }
    ]
}
```

Fasi successive

Dopo aver creato e configurato gli endpoint VPC per AWS IoT Secure Tunneling, considera quanto segue:

- Verifica la configurazione dell'endpoint VPC collegando i dispositivi tramite l'endpoint.
- Monitora l'utilizzo degli endpoint VPC tramite metriche. Amazon CloudWatch
- Rivedi e aggiorna le policy degli endpoint VPC in base alle tue esigenze di sicurezza.

Per ulteriori informazioni sul tunneling AWS IoT Device Management sicuro, consulta. AWS IoT Secure Tunneling

Sicurezza dell'infrastruttura in AWS IoT

Essendo una raccolta di servizi gestiti, AWS IoT è protetto dalle procedure di sicurezza di rete AWS globali descritte nel white paper Amazon Web Services: Overview of Security Processes.

Utilizzi chiamate API AWS pubblicate per accedere AWS IoT tramite la rete. I client devono supportare Transport Layer Security (TLS) 1.2 o versioni successive. I client devono, inoltre, supportare le suite di crittografia con PFS (Perfect Forward Secrecy), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni come Java 7 e versioni successive, supporta tali modalità. Per ulteriori informazioni, consulta Sicurezza dei trasporti in AWS IoT Core.

Le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta che è associata a un IAM principale. In alternativa è possibile utilizzare AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

Monitoraggio della sicurezza delle flotte o dei dispositivi di produzione con Core AWS IoT

I parchi istanze IoT possono essere costituiti da un numero elevato di dispositivi con funzionalità diverse, usati per lunghi periodi di tempo e distribuiti in varie aree geografiche. Queste caratteristiche rendono la configurazione di un parco istanze complessa e soggetta a errori. E poiché i dispositivi presentano spesso vincoli di potenza di elaborazione, memoria e capacità di storage, ciò limita l'uso della crittografia e di altre forme di sicurezza nei dispositivi stessi. I dispositivi, inoltre, usano spesso software con vulnerabilità note. La combinazione di questi fattori rende i parchi istanze IoT un facile bersaglio per gli hacker e rende difficile la protezione continuativa di un parco istanze di dispositivi.

AWS IoT Device Defender affronta queste sfide fornendo strumenti per identificare i problemi di sicurezza e le deviazioni dalle migliori pratiche. È possibile utilizzare AWS IoT Device Defender per analizzare, controllare e monitorare i dispositivi connessi per rilevare comportamenti anomali e mitigare i rischi per la sicurezza. AWS IoT Device Defender può controllare le flotte di dispositivi per garantire che aderiscano alle migliori pratiche di sicurezza e rilevare comportamenti anomali sui dispositivi. In questo modo è possibile applicare politiche di sicurezza coerenti in tutto il parco AWS IoT dispositivi e rispondere rapidamente quando i dispositivi vengono compromessi. Per ulteriori informazioni, consulta AWS IoT Device Defender.

AWS IoT Device Advisor aggiorna e corregge il parco macchine in base alle esigenze. AWS IoT Device Advisor aggiorna automaticamente i casi di test. I test case che selezioni sono sempre con versione più recente. Per ulteriori informazioni, consulta Device Advisor.

Le migliori pratiche di sicurezza in AWS IoT Core

Questa sezione contiene informazioni sulle migliori pratiche di sicurezza per AWS IoT Core. Per informazioni sulle regole di sicurezza per le soluzioni IoT industriale, consultare Ten security golden rules for Industrial IoT solutions.

Protezione delle connessioni MQTT in AWS IoT

AWS IoT Coreè un servizio cloud gestito che consente ai dispositivi connessi di interagire con le applicazioni cloud e altri dispositivi in modo semplice e sicuro. AWS IoT Core supporta HTTP e MQTT WebSocket, un protocollo di comunicazione leggero progettato specificamente per tollerare connessioni intermittenti. Se ci si connette AWS IoT tramite MQTT, ciascuna connessione deve essere associata a un identificatore noto come ID client. Il client MQTT identifica in IDs modo univoco le connessioni MQTT. Se viene stabilita una nuova connessione utilizzando un ID client già richiesto per un'altra connessione, il broker di AWS IoT messaggi interrompe la vecchia connessione per consentire la nuova connessione. Il client IDs deve essere unico all'interno di ciascuno Account AWS di essi Regione AWS. Ciò significa che non è necessario imporre l'unicità globale del cliente al di IDs fuori della vostra regione Account AWS o in tutte le regioni all'interno della vostra. Account AWS

L'impatto e la gravità della chiusura delle connessioni MQTT nel parco istanze di dispositivi dipende da molti fattori. Ciò include:

- Il tuo caso d'uso (ad esempio, i dati a cui vengono inviati i dispositivi AWS IoT, la quantità di dati e la frequenza di invio dei dati).
- La configurazione client MQTT, ad esempio le impostazioni di riconnessione automatica, le tempistiche di back-off associate e l'utilizzo di sessioni persistenti MQTT).
- Vincoli delle risorse dispositivo.
- La causa radice delle disconnessioni, la sua aggressività e la sua persistenza.

Per evitare conflitti tra gli ID client e i loro potenziali impatti negativi, assicurati che ogni dispositivo o applicazione mobile disponga di una politica AWS IoT o IAM che limiti il client che IDs può essere utilizzato per le connessioni MQTT al broker di AWS IoT messaggi. Ad esempio, puoi utilizzare una policy IAM per impedire a un dispositivo di chiudere involontariamente la connessione di un altro dispositivo utilizzando un ID client già in uso. Per ulteriori informazioni, consulta Autorizzazione.

Tutti i dispositivi del parco dispositivi devono disporre di credenziali con privilegi che autorizzano solo le azioni previste, che includono (ma non solo) azioni AWS IoT MQTT come la pubblicazione

di messaggi o l'iscrizione ad argomenti con ambito e contesto specifici. Le policy di autorizzazione specifiche possono variare a seconda dei casi d'uso. Identifica le policy di autorizzazione che meglio soddisfano i requisiti aziendali e di sicurezza.

Per creare e gestire più facilmente le policy di autorizzazione, puoi utilizzare AWS IoT Core variabili politiche e le variabili delle policy IAM. Le variabili di policy possono essere inserite in una policy e, nel momento in cui questa viene valutata, vengono sostituite con i valori provenienti dalla richiesta del dispositivo. Con le variabili di policy è possibile creare una singola policy per concedere le autorizzazioni a più dispositivi. È possibile identificare le variabili di policy pertinenti per il proprio caso d'uso in base alla configurazione AWS IoT dell'account, al meccanismo di autenticazione e al protocollo di rete utilizzato per la connessione al broker di messaggi. AWS IoT Per scrivere policy di autorizzazione ottimali, considera gli aspetti specifici del tuo caso d'uso e il tuo modello di gestione delle minacce.

Ad esempio, se i dispositivi sono stati registrati nel AWS IoT registro, è possibile utilizzare le variabili Thing Policy nelle AWS IoT policy per concedere o negare le autorizzazioni in base alle proprietà degli oggetti come i nomi degli oggetti, i tipi di oggetto e i valori degli attributi degli oggetti. Il nome dell'oggetto viene ottenuto dall'ID client nel messaggio di connessione MQTT inviato quando un oggetto si connette a. AWS IoT Le variabili Thing Policy vengono sostituite quando un oggetto si connette AWS IoT tramite MQTT utilizzando l'autenticazione reciproca TLS o MQTT tramite il WebSocket protocollo utilizzando identità Amazon Cognito autenticate. Puoi utilizzare l'AttachThingPrincipalAPI per allegare certificati e identità Amazon Cognito autenticate a un oggetto. `iot:Connection.Thing.ThingName`è una variabile di policy utile per applicare le restrizioni relative agli ID client. La seguente AWS IoT politica di esempio richiede che il nome di un oggetto registrato venga utilizzato come ID client per le connessioni MQTT al broker di AWS IoT messaggi:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "iot:Connect",
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ]
  }
 ]
}
```

Se si desidera identificare i conflitti di ID client in corso, è possibile abilitare e utilizzare CloudWatch Logs for. AWS IoT Per ogni connessione MQTT che il broker di AWS IoT messaggi disconnette a causa di conflitti tra ID client, viene generato un record di registro simile al seguente:

```
{
    "timestamp": "2019-04-28 22:05:30.105",
    "logLevel": "ERROR",
    "traceId": "02a04a93-0b3a-b608-a27c-1ae8ebdb032a",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "clientId01",
    "principalId": "1670fcf6de55adc1930169142405c4a2493d9eb5487127cd0091ca0193a3d3f6",
    "sourceIp": "203.0.113.1",
    "sourcePort": 21335,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID"
}
```

È possibile utilizzare un filtro CloudWatch Logs, ad esempio per `{$.reason= "DUPLICATE_CLIENT_ID" }` cercare istanze di conflitti tra ID client o per impostare filtri CloudWatch metrici e CloudWatch allarmi corrispondenti per il monitoraggio e la segnalazione continui.

Puoi utilizzare AWS IoT Device Defender per identificare politiche IAM e eccessivamente permissive. AWS IoT AWS IoT Device Defender fornisce anche un controllo di controllo che ti avvisa se più dispositivi del tuo parco dispositivi si connettono al broker di AWS IoT messaggi utilizzando lo stesso ID cliente.

Puoi utilizzare AWS IoT Device Advisor per verificare che i tuoi dispositivi possano connettersi in modo affidabile AWS IoT Core e seguire le migliori pratiche di sicurezza.

Consulta anche

- AWS IoT Core
- Caratteristiche di sicurezza di AWS IoT
- AWS IoT Core variabili politiche
- Variabili delle policy IAM

- Amazon Cognito Identity
- AWS IoT Device Defender
- CloudWatch Registri per AWS IoT

Tieni sincronizzato l'orologio del dispositivo

È importante avere un orario preciso sul dispositivo. I certificati X.509 hanno data e ora di scadenza. L'orologio sul dispositivo viene utilizzato per verificare che un certificato server sia ancora valido. Se costruisci dispositivi IoT commerciali, ricorda che i tuoi prodotti potrebbero essere conservati per periodi prolungati prima di essere venduti. Gli orologi in tempo reale possono fermarsi nel corso del tempo e le batterie possono scaricarsi, quindi impostare il tempo in fabbrica non è sufficiente.

Per la maggior parte dei sistemi, ciò significa che il software del dispositivo deve includere un client NTP (Network Time Protocol). Il dispositivo deve attendere fino a quando si sincronizza con un server NTP prima di tentare di connettersi a AWS IoT Core. Se ciò non è possibile, il sistema dovrebbe fornire agli utenti la possibilità di impostare l'ora del dispositivo in modo che le connessioni successive abbiano esito positivo.

Dopo che il dispositivo è sincronizzato con un server NTP, può aprire una connessione con AWS IoT Core. L'inclinazione del clock consentita dipende dalla finalità della connessione.

Convalidare il certificato server

La prima cosa con cui un dispositivo interagisce AWS IoT è aprire una connessione sicura. Quando colleghi il dispositivo a AWS IoT, assicurati di parlare con un altro server AWS IoT e di non AWS IoT impersonare un altro server. Ciascun AWS IoT server è dotato di un certificato rilasciato per il dominio. `iot.amazonaws.com` Questo certificato è stato rilasciato AWS IoT da un'autorità di certificazione affidabile che ha verificato la nostra identità e la proprietà del dominio.

Una delle prime cose che AWS IoT Core fa quando un dispositivo si connette è inviare al dispositivo un certificato server. I dispositivi possono verificare che aspettino di connettersi a `iot.amazonaws.com` e che il server alla fine di tale connessione disponga di un certificato di un'autorità attendibile per tale dominio.

I certificati TLS sono in formato X.509 e includono una serie di informazioni, ad esempio il nome, la posizione, il nome di dominio e un periodo di validità dell'organizzazione. Il periodo di validità è specificato come una coppia di valori di tempo chiamati `notBefore` e `notAfter`. Servizi come ad

esempio AWS IoT Core utilizzano periodi di validità limitati (ad esempio, un anno) per i certificati server e iniziano a fornirne di nuovi prima della scadenza di quelli vecchi.

Utilizzare una singola identità per dispositivo

Utilizzare una singola identità per client. I dispositivi utilizzano generalmente certificati client X.509. Le applicazioni per il Web e i dispositivi mobili utilizzano Amazon Cognito Identity. In questo modo è possibile applicare le autorizzazioni a grana fine ai dispositivi.

Ad esempio, disponi di un'applicazione costituita da un dispositivo mobile che riceve aggiornamenti di stato da due diversi oggetti smart home, una lampadina e un termostato. La lampadina invia lo stato del suo livello della batteria e un termostato invia messaggi che segnalano la temperatura.

AWS IoT autentica i dispositivi singolarmente e tratta ogni connessione singolarmente. È possibile applicare controlli di accesso a grana fine utilizzando le policy di autorizzazione. È possibile definire una policy per il termostato che consenta la pubblicazione in uno spazio argomento. È possibile definire una policy separata per la lampadina che consenta la pubblicazione su uno spazio argomento diverso. Infine è possibile definire una policy per l'app mobile che consente solo di connettersi e sottoscrivere gli argomenti relativi al termostato e alla lampadina per ricevere messaggi da questi dispositivi.

Applicare il principio del minimo privilegio e definire l'ambito delle autorizzazioni per dispositivo il più possibile. Tutti i dispositivi o gli utenti devono disporre di una AWS IoT politica AWS IoT che consenta loro di connettersi solo con un ID client noto e di pubblicare e sottoscrivere una serie di argomenti identificati e fissi.

Usa un secondo Regione AWS come backup

Prendi in considerazione l'idea di archiviare una copia dei tuoi dati in un secondo Regione AWS come backup. Tieni presente che la AWS soluzione denominata Disaster Recovery for non AWS IoTè più disponibile. Sebbene la GitHublibreria associata rimanga accessibile, l' AWS ha resa obsoleta nel luglio 2023 e non fornisce più manutenzione o supporto. Per implementare le tue soluzioni o per esplorare opzioni di supporto aggiuntive, visita Contatti. AWS Se al tuo account è associato un AWS Technical Account Manager, contattalo per ricevere assistenza.

Utilizzare il provisioning Just In Time

La creazione e il provisioning manuali di ogni dispositivo possono richiedere molto tempo. AWS IoT fornisce un modo per definire un modello a cui effettuare il provisioning dei dispositivi

quando si connettono per AWS IoT la prima volta. Per ulteriori informazioni, consulta Just-in-time approvvigionamento.

Autorizzazioni per eseguire i test di AWS IoT Device Advisor

Il seguente modello di policy mostra le autorizzazioni minime e l'entità IAM necessarie per eseguire i casi di test di AWS IoT Device Advisor. Dovrai sostituirlo *your-device-role-arn* con il ruolo del dispositivo Amazon Resource Name (ARN) che hai creato in base ai prerequisiti.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thinggroup/your-thing-
group",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iotdeviceadvisor.amazonaws.com"
            }
        }
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke*",
                "iam:ListRoles",
                "iot:Connect",
                "iot:CreateJob",
                "iot:DeleteJob",
                "iot:DescribeCertificate",
                "iot:DescribeEndpoint",
                "iotjobsdata:DescribeJobExecution",
                "iot:DescribeJob",
                "iot:DescribeThing",
                "iotjobsdata:GetPendingJobExecutions",
                "iot:GetPolicy",
                "iot:ListAttachedPolicies",
                "iot:ListCertificates",
```

```
                "iot:ListPrincipalPolicies",
                "iot:ListThingPrincipals",
                "iot:ListThings",
                "iot:Publish",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:UpdateJobExecution",
                "iot:UpdateThingShadow",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "iotdeviceadvisor:*",
            "Resource": "*"
        }
    ]
}
```

Prevenzione del problema "confused deputy" tra servizi per Device Advisor

Il problema confused deputy è un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire un'azione può costringere un'entità maggiormente privilegiata a eseguire l'azione. Nel frattempo AWS, l'impersonificazione tra servizi può portare al confuso problema del vice. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare che ciò accada, AWS mette a disposizione strumenti che consentono di proteggere i dati relativi a tutti i servizi con responsabili del servizio a cui è stato concesso l'accesso alle risorse del vostro account.

Consigliamo di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy delle risorse per limitare le autorizzazioni con cui Device Advisor fornisce un altro servizio alla risorsa. Se si utilizzano entrambe le chiavi di contesto delle condizioni

globali, il valore `aws:SourceAccount` e l'account nel valore `aws:SourceArn` devono utilizzare lo stesso ID account nella stessa istruzione di policy.

Il valore di `aws:SourceArn` deve essere l'ARN della risorsa di definizione della suite. La risorsa di definizione della suite fa riferimento alla suite di test creata con Device Advisor.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. Se non si conosce l'ARN completo della risorsa o si scelgono più risorse, è necessario utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con caratteri jolly (*) per le parti sconosciute dell'ARN. Ad esempio, `arn:aws:iotdeviceadvisor:*:`*`account-id`*`:suitedefinition/*`.

L'esempio seguente mostra il modo in cui puoi utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` in Device Advisor per prevenire il problema "confused deputy".

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid": "ConfusedDeputyPreventionExamplePolicy",
        "Effect": "Allow",
        "Principal": {
            "Service": "iotdeviceadvisor.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": "arn:aws:iotdeviceadvisor:us-
east-1:123456789012:suitedefinition/ygp6rxa3tzvn"
        },
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
        }
        }
    }
}
```

AWS formazione e certificazione

Segui il seguente corso per conoscere i concetti chiave AWS IoT della sicurezza: AWS IoT Security Primer.

Monitoraggio AWS IoT

Il monitoraggio è un elemento importante per mantenere l'affidabilità, la disponibilità e le prestazioni delle AWS IoT vostre AWS soluzioni.

Consigliamo vivamente di raccogliere dati di monitoraggio da tutte le parti della AWS soluzione per semplificare il debug di un errore multipunto, se si verifica. Iniziare creando un piano di monitoraggio che risponda alle seguenti domande. Se non si è sicuri di come rispondere, è comunque possibile continuare ad abilitare la registrazione e stabilire le baseline delle prestazioni.

- Quali sono gli obiettivi del monitoraggio?
- Quali risorse verranno monitorate?
- Con quale frequenza eseguirai il monitoraggio di queste risorse?
- Quali strumenti di monitoraggio verranno usati?
- Chi eseguirà i processi di monitoraggio?
- Chi deve ricevere una notifica quando si verifica un problema?

Il passaggio successivo consiste nell'abilitare la registrazione e stabilire una linea di base delle AWS IoT prestazioni normali nel proprio ambiente misurando le prestazioni in diversi momenti e in diverse condizioni di carico. Durante il monitoraggio AWS IoT, conservate i dati di monitoraggio storici in modo da poterli confrontare con i dati sulle prestazioni correnti. Questo permette di identificare i normali modelli di prestazioni e le anomalie di prestazioni e definire metodi per risolvere i problemi.

Per stabilire le prestazioni di base per cui AWS IoT, dovresti monitorare queste metriche prima di tutto. È sempre possibile monitorare più parametri in un secondo momento.

- `PublishIn.Success`
- `PublishOut.Success`
- `Subscribe.Success`
- `Ping.Success`
- `Connect.Success`
- `GetThingShadow.Accepted`
- `UpdateThingShadow.Accepted`
- `DeleteThingShadow.Accepted`

- `RulesExecuted`

Gli argomenti di questa sezione consentono di avviare la registrazione e il monitoraggio di AWS IoT.

Argomenti

- Configurare la registrazione AWS IoT
- Monitora AWS IoT allarmi e metriche con Amazon CloudWatch
- Monitora AWS IoT utilizzando i log CloudWatch
- Carica i log lato dispositivo su Amazon CloudWatch
- Registrazione delle chiamate AWS IoT API utilizzando AWS CloudTrail

Configurare la registrazione AWS IoT

È necessario abilitare la registrazione utilizzando la AWS IoT console, la CLI o l'API prima di poter monitorare e AWS IoT registrare l'attività. È possibile configurare la registrazione AWS IoT a tre livelli: a livello di account, a livello di evento o a livello di risorsa specifica. La registrazione a livello di evento e specifica della risorsa è disponibile esclusivamente con la registrazione V2. I clienti che utilizzano la registrazione V1 devono eseguire una migrazione alla V2 per accedere a queste funzionalità. Vedi i dettagli.

Quando si valuta come configurare la AWS IoT registrazione, la configurazione di registrazione a livello di account determina come verranno registrate le AWS IoT attività, salvo diversa indicazione. All'inizio, potresti voler ottenere log dettagliati con un livello di registro predefinito di INFO o DEBUG. Dopo aver esaminato i log iniziali, è possibile modificare il livello di registro predefinito impostandolo su un livello meno dettagliato, ad esempio WARN o ERROR a livello di account o di evento, e impostare un livello di registro più dettagliato e specifico per le risorse che potrebbero richiedere maggiore attenzione. I livelli di log possono essere modificati ogni volta che vuoi.

Questo argomento riguarda l'accesso dal lato cloud. AWS IoTPer informazioni sulla registrazione e il monitoraggio lato dispositivo, consulta Caricare i log lato dispositivo su. CloudWatch

Per informazioni sulla registrazione e il monitoraggio, consulta Registrazione e monitoraggio. AWS IoT GreengrassAWS IoT Greengrass

Configurazione dell'accesso V2 AWS IoT

Determinazione della versione di registrazione

L'LoggingOptions API GetV2 restituisce un messaggio NotConfiguredException se la registrazione V2 non è abilitata. Questo errore si verifica quando la registrazione V1 è in uso o non è stata configurata alcuna registrazione.

Comprensione delle funzionalità di registrazione V2

La registrazione V2 offre due funzionalità chiave: registrazione a livello di evento e registrazione specifica delle risorse. La registrazione a livello di evento consente una configurazione di registrazione mirata con livelli di registro e destinazioni di gruppi di log personalizzabili. CloudWatch La registrazione specifica delle risorse consente di filtrare i log per gruppo di oggetti, IP di origine, ID client o ID principale. Insieme, queste funzionalità forniscono un controllo granulare e una visibilità completa sulle operazioni IoT, migliorando la ricercabilità dei log e riducendo i costi eliminando le attività di registrazione non necessarie.

Migrazione da V1 a V2

È possibile migrare alla registrazione V2 utilizzando l'API SetV2 LoggingOptions tramite la CLI o l'SDK. AWS Dopo la migrazione, indirizza AWS IoT automaticamente tutti i log al gruppo di log «LogSv2". CloudWatch AWSIot Importante: se le applicazioni o le risorse downstream consumano informazioni dai «AWSIotLog», aggiornale per utilizzare il percorso del gruppo di log corrispondente.

Configurare il ruolo e il criterio di registrazione

Prima di poter abilitare l'accesso AWS IoT, devi creare un ruolo IAM e una policy che AWS IoT autorizzi a scrivere le attività di AWS IoT registro nei gruppi di CloudWatch log per tuo conto. Puoi anche generare un ruolo IAM con le politiche necessarie nella sezione Logs della AWS IoT console.

> **Note**
>
> Prima di abilitare AWS IoT la registrazione, assicurati di aver compreso le autorizzazioni di CloudWatch accesso ai log. Gli utenti con accesso ai CloudWatch registri possono visualizzare le informazioni di debug dai tuoi dispositivi. Per ulteriori informazioni, consulta Authentication and Access Control for Amazon CloudWatch Logs.
> Se ti aspetti un traffico elevato a AWS IoT Core causa dei test di carico, prendi in considerazione la possibilità di disabilitare la registrazione IoT per evitare il throttling. Se

viene rilevato un traffico elevato, il nostro servizio potrebbe disabilitare l'accesso al tuo account.

Di seguito viene illustrato come creare un ruolo e una politica di registrazione per le risorse. AWS IoT Core

Creazione di un ruolo di logging

Per creare un ruolo di registrazione, apri la finestra di dialogo Roles hub of the IAM console (Hub ruoli della console IAM) e scegli Create role (Crea ruolo).

1. In Select trusted entity (Seleziona un'entità attendibile), scegli AWS Service. Quindi scegli IoT in Use case (Caso d'uso). Se IoT non viene visualizzata, entra e cerca IoT dal menu a discesa Casi d'uso per altri servizi AWS :. Seleziona Avanti.
2. Nella pagina Add permissions (Aggiungi autorizzazioni), vengono visualizzate le policy che sono automaticamente collegate al ruolo di servizio. Scegli Next (Successivo).
3. Nella pagina Name, review, and create (Nomina, verifica e crea), immetti Role name (Nome ruolo) e Role description (Descrizione ruolo) per il ruolo, quindi scegli Create role (Crea ruolo).

Policy del ruolo di logging

I seguenti documenti politici forniscono la politica relativa al ruolo e la politica di fiducia AWS IoT a cui è possibile inviare voci di registro per CloudWatch conto dell'utente. Se si configura la registrazione a livello di evento con un gruppo di CloudWatch log personalizzato, è necessario aggiornare la politica del ruolo per includere l'ARN della risorsa personalizzata.

Se hai anche autorizzato l'invio AWS IoT Core per LoRaWAN di voci di registro, vedrai un documento di policy creato appositamente per te che registra entrambe le attività.

> **Note**
>
> Questi documenti sono stati creati al momento della creazione del ruolo di logging. I documenti contengono variabili, e `${partition}`, `${region}` `${accountId}`, che devi sostituire con i tuoi valori.
>
> - Sostituisci la partizione con la partizione della regione.

- Sostituisci la regione con la AWS regione che usi. Assicurati di utilizzare la stessa AWS regione che hai usato per configurare la AWS CLI sul tuo dispositivo.
- Sostituisci account-id con l'ID del tuo AWS account.

Policy del ruolo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "iot:GetLoggingOptions",
                "iot:SetLoggingOptions",
                "iot:SetV2LoggingOptions",
                "iot:GetV2LoggingOptions",
                "iot:SetV2LoggingLevel",
                "iot:ListV2LoggingLevels",
                "iot:DeleteV2LoggingLevel"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:AWSIotLogsV2:*"
            ]
        }
    ]
}
```

Politica di fiducia per registrare solo AWS IoT Core le attività:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
        },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

AWS IoT La registrazione potrebbe non riuscire a pubblicare la registrazione su CloudWatch Logs a causa delle autorizzazioni dei ruoli IAM insufficienti. In questo caso, controlla le metriche di CloudWatch registrazione per esaminare e risolvere gli errori.

Configura la registrazione nella (console) AWS IoT

Questa sezione descrive come configurare la AWS IoT registrazione utilizzando la AWS IoT console. È possibile configurare la registrazione a livello di account, a livello di evento e specifica della risorsa.

Per AWS IoT configurare la registrazione:

1. Accedi alla AWS IoT console. Per ulteriori informazioni, consulta Apri la console AWS IoT.
2. Nel riquadro di navigazione a sinistra, scegli Registri (in precedenza una sezione in Impostazioni).
3. Configura la registrazione a livello di account: la registrazione a livello di account si applica a tutto il tuo AWS IoT parco dispositivi (dispositivi o endpoint), a meno che non venga sostituita da impostazioni a livello di evento o di risorsa.

    a. In Registrazione a livello di account, seleziona Gestisci la registrazione a livello di account per effettuare gli aggiornamenti.

    b. Seleziona la casella di controllo «Abilita registrazione» a cui iniziare a inviare i log. CloudWatch Quando «Abilita registrazione» è deselezionata, non AWS IoT invierà alcun registro ai gruppi di CloudWatch log, indipendentemente dalle configurazioni di registrazione a livello di evento o di risorsa.

    c. In IAM log role, seleziona un ruolo esistente dall'elenco a discesa. Puoi visualizzare i dettagli del ruolo per controllare le autorizzazioni del ruolo. In alternativa, seleziona Crea nuovo ruolo per configurare un nuovo ruolo IAM. Il ruolo di registrazione fornisce politiche che consentono di AWS IoT inviare voci di registro per tuo CloudWatch conto. Se si configura

la registrazione a livello di evento con un gruppo di CloudWatch log personalizzato, è necessario aggiornare la politica del ruolo per includere l'ARN per questo gruppo di log.

d. Scegliete il livello di registro predefinito che corrisponde al livello di dettaglio delle voci di registro che desiderate visualizzare nei registri. CloudWatch Nota: il livello di registro «DEBUG» fornisce il maggior numero di dettagli ma aumenta CloudWatch i costi. CloudWatch le destinazioni dei gruppi di log non possono essere configurate a livello di account. Tuttavia, è possibile specificare gruppi di log personalizzati per singoli tipi di eventi come descritto nella sezione seguente.

e. Scegliete Aggiorna registrazione per salvare le modifiche.

4. La registrazione a livello di evento consente di acquisire in modo selettivo i registri degli eventi pertinenti e di indirizzarli a gruppi di log dedicati. CloudWatch Ciò offre la flessibilità necessaria per organizzare i log in base ai casi d'uso per una migliore individuabilità, condividerli con diversi destinatari e ridurre i CloudWatch costi abilitando i log e impostando i livelli di registro in base alla criticità degli eventi.

   Configura la registrazione a livello di evento: la registrazione a livello di evento acquisisce eventi specifici, come i tentativi di autenticazione del client. AWS IoT Queste impostazioni hanno la precedenza sulla registrazione a livello di account.

   a. Nella sezione Registrazione a livello di evento, seleziona Gestisci registrazione a livello di evento per effettuare gli aggiornamenti.

   b. Per impostazione predefinita, i tipi di eventi ereditano la configurazione di registrazione a livello di account. Nota: quando è configurata la registrazione specifica per una risorsa, ha la precedenza sulle impostazioni a livello di account ed evento.

   c. Per modificare le impostazioni per i singoli eventi, fate clic sul valore nella riga dell'evento corrispondente. È possibile regolare sia il livello di registro che la destinazione del gruppo di CloudWatch log. Quando si specifica una destinazione personalizzata per un gruppo di CloudWatch log, è necessario verificare che la policy del ruolo IAM includa le autorizzazioni per il nuovo gruppo di log. Il mancato aggiornamento della politica del ruolo AWS IoT impedirà la scrittura di log nel gruppo di log personalizzato. Dopo aver effettuato la selezione, fai clic sul segno di spunta per confermare la scelta. La colonna «È modificato» mostrerà «Sì» per indicare le modifiche in sospeso.

   d. Fai clic su Update Logging per applicare le modifiche o scegli Annulla per eliminarle.

5. Configura le sostituzioni specifiche della risorsa: le sostituzioni specifiche della risorsa applicano un'impostazione di registrazione alle risorse selezionate. Una risorsa può essere un gruppo di oggetti, un IP di origine, un ID client o un ID principale. La configurazione di registrazione

specifica della risorsa ha la precedenza sulle impostazioni a livello di account e a livello di evento. Se abilitata, genera registri per tutti i tipi di eventi al livello di registrazione configurato per le risorse specificate. Ad esempio, è possibile impostare la registrazione a livello di debug per un oggetto specifico mantenendo la registrazione a livello di informazioni per tutti gli altri oggetti.

a. Seleziona Aggiungi sostituzioni specifiche della risorsa nella sezione Sostituzioni specifiche della risorsa.
b. Scegliete una destinazione del log: Thing group, Source IP, Client ID o Principal ID.
c. Immettete il valore di destinazione del log corrispondente per il tipo di destinazione selezionato.
d. Seleziona il livello di registro desiderato dal menu a discesa nella sezione Livello di registro specifico della risorsa.
e. Fai clic su Invia per aggiungere l'override o su Annulla per annullare le modifiche.
f. Per modificare una modifica esistente specifica della risorsa, seleziona la casella di controllo accanto alla risorsa e fai clic su «Rimuovi» per eliminare l'override o su «Modifica» per modificare.

Dopo aver abilitato la registrazione, visita Visualizzazione dei AWS IoT log nella console CloudWatch per ulteriori informazioni sulla visualizzazione delle voci di registro.

Configurazione dell'accesso a livello di account ed evento (AWS IoT CLI)

Questa sezione descrive come configurare la registrazione globale per AWS IoT utilizzando la CLI.

È possibile configurare facoltativamente la registrazione a livello di evento. La registrazione a livello di evento consente di acquisire informazioni di registrazione a livello di evento, ad esempio eventi di autenticazione e autorizzazione o di creazione di certificati. È possibile personalizzare sia il livello di registro che le destinazioni dei gruppi di CloudWatch log a livello di evento. La registrazione a livello di evento opera a un livello più mirato rispetto alla registrazione a livello di account e pertanto ha la precedenza sulle impostazioni di registrazione a livello di account. Questo approccio gerarchico consente di mantenere diverse strategie di registrazione per diversi tipi di eventi in base alla loro importanza operativa e alle considerazioni relative ai costi.

> **Note**
>
> Hai bisogno dell'ARN (Amazon Resource Name) del ruolo che desideri utilizzare. Se è necessario creare un ruolo da utilizzare per la registrazione, consulta Creazione di un ruolo di

logging prima di continuare. Quando specificate un gruppo di CloudWatch log personalizzato per qualsiasi tipo di evento, assicuratevi che il vostro ruolo di registrazione disponga delle autorizzazioni necessarie per il gruppo di log di destinazione.
L'entità principale utilizzata per effettuare la chiamata API deve avere Passaggio delle autorizzazioni ai ruoli per il ruolo di logging.

È inoltre possibile eseguire questa procedura con l'API utilizzando i metodi nell' AWS API che corrispondono ai comandi CLI mostrati qui.

Per utilizzare la CLI per configurare la registrazione predefinita per AWS IoT

1. Usa il comando set-v2-logging-options per impostare le opzioni di logging per l'account.

```
aws iot set-v2-logging-options \
    --event-configurations event-configuration-list \
    --role-arn logging-role-arn \
    --default-log-level log-level
```

dove:

--role-arn

Il ruolo ARN che concede AWS IoT il permesso di scrivere nei log in Logs. CloudWatch La configurazione Role-ARN è richiesta per la configurazione iniziale.

--default-log-level

Il livello di log da usare. I valori validi sono:`ERROR`,, `WARN`INFO, `DEBUG` o. `DISABLED` Default-log-levella configurazione è richiesta per la configurazione iniziale.

--no-disable-all-logs

Un parametro opzionale che abilita tutte le AWS IoT registrazioni. Utilizzare questo parametro per abilitare la registrazione quando è attualmente disattivata.

--disable-all-logs

Un parametro opzionale che disabilita tutte le AWS IoT registrazioni. Utilizzare questo parametro per disabilitare la registrazione quando è attualmente abilitata.

--event-configurations

Questo parametro è facoltativo e consente di personalizzare le impostazioni di registrazione per singoli tipi di eventi:

- eventType: il tipo di evento che sostituisce l'impostazione a livello di account
- LogLevel: sovrascrive l'impostazione a livello di account con DEBUG, INFO, ERROR, WARN o DISABLED
- LogDestination: specifica un gruppo di log personalizzato per la consegna dei log CloudWatch

È possibile configurare il livello di registrazione e la destinazione del registro in modo indipendente per ogni tipo di evento. Se non specificato, gli eventi erediteranno le impostazioni a livello di account

```
aws iot set-v2-logging-options \
    --event-configurations "[{\"eventType\":\"Publish-In\",\"logLevel\":\"INFO\",
\"logDestination\":\"examplePublishInLogGroup\"}]"
```

2. Utilizzare il comando get-v2-logging-options per ottenere le opzioni di registrazione correnti.

```
aws iot get-v2-logging-options \
    --verbose
```

dove:

--verbose

Un parametro opzionale che consente di recuperare tutti gli EventTypes e le relative configurazioni.

Dopo aver abilitato la registrazione, visita Visualizzazione dei AWS IoT log nella console CloudWatch per ulteriori informazioni sulla visualizzazione delle voci di registro.

> **Note**
>
> AWS IoT continua a supportare i comandi precedenti (set-logging-optionseget-logging-options) a impostare e ottenere la registrazione globale sul tuo account. Va tenuto presente che, utilizzando questi comandi, i log risultanti contengono testo normale anziché payload

JSON e la latenza di logging è generalmente più elevata. Non verranno apportati ulteriori miglioramenti all'implementazione dei comandi precedenti. Ti consigliamo di utilizzare le versioni «v2" per configurare le opzioni di registrazione e, quando possibile, modificare le applicazioni precedenti che utilizzano le versioni precedenti.

Configura le sostituzioni specifiche per le risorse in (CLI) AWS IoT

Questa sezione descrive come configurare le sostituzioni specifiche della risorsa utilizzando la CLI. AWS IoT Le sostituzioni specifiche della risorsa consentono di specificare un livello di registrazione per una risorsa specifica identificata da Thing Group, Client ID, Source IP o Principal ID. Quando la registrazione specifica della risorsa è abilitata, sostituisce sia le impostazioni a livello di account che a livello di evento. Tutti i tipi di eventi genereranno registri per la risorsa specificata al livello di registrazione configurato, anche se tali eventi sono disabilitati nelle configurazioni a livello di evento.

I gruppi di oggetti possono contenere altri gruppi di oggetti per creare una relazione gerarchica. In questa procedura viene descritto come configurare la registrazione di un singolo gruppo di oggetti. È possibile applicare questa procedura al gruppo di elementi padre in una gerarchia per configurare la registrazione per tutti i gruppi di elementi nella gerarchia. È inoltre possibile applicare questa procedura a un gruppo di elementi figlio per ignorare la configurazione di registrazione del relativo elemento padre.

Un oggetto può essere membro di un gruppo di oggetti. Questa appartenenza consente all'oggetto di ereditare configurazioni, politiche e impostazioni applicate al gruppo di oggetti. I gruppi di oggetti vengono utilizzati per gestire e applicare le impostazioni a più elementi collettivamente, anziché gestire ogni elemento singolarmente. Quando l'ID client corrisponde al nome dell'oggetto, AWS IoT Core assocerà automaticamente la sessione client alla risorsa oggetto corrispondente. Ciò consente alla sessione client di ereditare le configurazioni e le impostazioni applicate ai gruppi di oggetti a cui appartiene l'oggetto, inclusi i livelli di registrazione. Se l'ID cliente non corrisponde al nome dell'oggetto, è possibile abilitare l'allegato esclusivo dell'oggetto per stabilire l'associazione. Per ulteriori informazioni, consulta ???.

Oltre ai gruppi di oggetti, puoi anche registrare destinazioni come l'ID client di un dispositivo, l'IP di origine e l'ID principale.

> **Note**
>
> Hai bisogno dell'Amazon Resource Name (ARN) del ruolo che desideri utilizzare. Se è necessario creare un ruolo da utilizzare per la registrazione, consulta Creazione di un ruolo di logging prima di continuare.
> L'entità principale utilizzata per effettuare la chiamata API deve avere Passaggio delle autorizzazioni ai ruoli per il ruolo di logging.

È inoltre possibile eseguire questa procedura con l'API utilizzando i metodi nell' AWS API che corrispondono ai comandi CLI mostrati qui.

Per utilizzare la CLI per configurare le sostituzioni specifiche della risorsa per AWS IoT

1. Abilita la registrazione a livello di account prima di configurare la registrazione specifica della risorsa utilizzando il seguente comando: comando aws iot set-v2-logging-options
2. Usa set-v2-logging-levelil comando per configurare le sostituzioni specifiche della risorsa. Vedi il seguente esempio per la configurazione del gruppo di oggetti:

   ```
   aws iot set-v2-logging-level \
                --log-target targetType=THING_GROUP,targetName=thing_group_name \
                --log-level log_level
   ```

   --log-target

   Il tipo e il nome della risorsa per cui stai configurando il logging. Il valore `targetType` deve essere uno dei seguenti: `THING_GROUP` | `CLIENT_ID` | `SOURCE_IP` | `PRINCIPAL_ID`. Il valore del parametro log-target può essere testo, come illustrato nell'esempio di comando precedente, o una stringa JSON, ad esempio nell'esempio seguente.

   ```
   aws iot set-v2-logging-level \
                --log-target '{"targetType": "THING_GROUP","targetName":
    "thing_group_name"}' \
                --log-level log_level
   ```

   --log-level

   Livello di logging usato durante la generazione di log per la risorsa specificata. I valori validi sono: DEBUG, INFO, ERROR, WARN e DISABLED.

3. Utilizzare il comando list-v2-logging-levels per elencare i livelli di registrazione attualmente configurati.

```
aws iot list-v2-logging-levels
```

4. Utilizza il comando delete-v2-logging-level per eliminare un livello di registrazione specifico delle risorse, come negli esempi seguenti.

```
aws iot delete-v2-logging-level \
              --target-type "THING_GROUP" \
              --target-name "thing_group_name"
```

```
aws iot delete-v2-logging-level \
              --target-type=CLIENT_ID
              --target-name=ClientId1
```

--target-type

Il valore `target-type` deve essere uno dei seguenti: `THING_GROUP` | `CLIENT_ID` | `SOURCE_IP` | `PRINCIPAL_ID`.

--target-name

Nome del gruppo di oggetti per il quale rimuovere il livello di registrazione.

Livelli di log

Questi livelli di log determinano gli eventi registrati e si applicano ai livelli di log predefiniti e specifici delle risorse.

ERRORE

Qualsiasi errore che provoca la mancata riuscita di un'operazione.

Esempio: impossibile autenticare il dispositivo a causa del certificato scaduto.

I log includono solo informazioni per il livello ERROR.

WARN

Qualsiasi evento che può potenzialmente causare incoerenze nel sistema, ma potrebbe non provocare la mancata riuscita dell'operazione.

Esempio: avvicinamento al limite di frequenza dei messaggi.

I log includono informazioni per i livelli ERROR e WARN.

INFO

Informazioni generali sul flusso di oggetti.

Esempio: un cliente si è iscritto con successo a un argomento MQTT.

I log includono informazioni per i livelli INFO, ERROR e WARN.

DEBUG

Informazioni che possono essere utili quando si esegue il debug di un problema.

Esempio: IoT Rules Engine ha rilevato un messaggio pubblicato nell'argomento delle regole «rule/test» e ha avviato correttamente l'esecuzione. La regola è configurata con. RepublishAction

I log includono informazioni per i livelli DEBUG, INFO, ERROR e WARN.

DISABILITATO

Tutte le attività di logging sino disabilitate.

Monitora AWS IoT allarmi e metriche con Amazon CloudWatch

Puoi monitorare AWS IoT l'utilizzo CloudWatch, che raccoglie ed elabora i dati grezzi trasformandoli in metriche leggibili e quasi AWS IoT in tempo reale. Queste statistiche vengono registrate per un periodo di due settimane, per permettere l'accesso alle informazioni storiche e offrire una prospettiva migliore sulle prestazioni del servizio o dell'applicazione Web. Per impostazione predefinita, i dati AWS IoT metrici vengono inviati automaticamente a CloudWatch intervalli di un minuto. Per ulteriori informazioni, consulta What Are Amazon CloudWatch, Amazon CloudWatch Events e Amazon CloudWatch Logs? nella Amazon CloudWatch User Guide.

Utilizzo delle AWS IoT metriche

Le metriche riportate da AWS IoT forniscono informazioni che è possibile analizzare in diversi modi. I casi d'uso seguenti sono basati su uno scenario in cui sono presenti dieci oggetti che si connettono a Internet una volta al giorno. Ogni giorno:

- Dieci cose si connettono AWS IoT all'incirca nello stesso momento.

- Ogni oggetto sottoscrive un filtro di argomenti e attende un'ora prima della disconnessione. Durante questo periodo, gli oggetti comunicano tra loro e apprendono ulteriori informazioni sullo stato del mondo.
- Ogni oggetto pubblica percezioni basate sui dati appena rilevati usando `UpdateThingShadow`.
- Ogni cosa si disconnette da AWS IoT.

Per aiutarti a iniziare, questi argomenti esplorano alcune delle domande che potresti avere.

- Come è possibile ricevere una notifica se gli oggetti non si connettono correttamente ogni giorno?
- Come è possibile ricevere una notifica se gli oggetti non pubblicano dati ogni giorno?
- Come è possibile ricevere una notifica se gli aggiornamenti alle copie shadow degli oggetti vengono rifiutati ogni giorno?
- Come posso creare un CloudWatch allarme per Jobs?

Ulteriori informazioni su CloudWatch allarmi e metriche

- Creazione di CloudWatch allarmi da monitorare AWS IoT
- AWS IoT metriche e dimensioni

Creazione di CloudWatch allarmi da monitorare AWS IoT

Puoi creare un CloudWatch allarme che invia un messaggio Amazon SNS quando l'allarme cambia stato. Un allarme monitora un singolo parametro per un periodo di tempo specificato. Quando il valore del parametro supera una determinata soglia in una serie di periodi di tempo, vengono eseguite una o più azioni. L'operazione corrisponde all'invio di una notifica a un argomento di Amazon SNS o a una policy di Auto Scaling. Gli allarmi attivano azioni solo per cambiamenti di stato sostenuti. CloudWatch gli allarmi non attivano azioni semplicemente perché si trovano in uno stato particolare; lo stato deve essere cambiato e mantenuto per un determinato numero di periodi.

Negli argomenti seguenti vengono descritti alcuni esempi di utilizzo degli allarmi CloudWatch.

- Come è possibile ricevere una notifica se gli oggetti non si connettono correttamente ogni giorno?
- Come è possibile ricevere una notifica se gli oggetti non pubblicano dati ogni giorno?
- Come è possibile ricevere una notifica se gli aggiornamenti alle copie shadow degli oggetti vengono rifiutati ogni giorno?
- Come posso creare un CloudWatch allarme per i lavori?

Puoi vedere tutte le metriche che gli CloudWatch allarmi possono monitorare. AWS IoT metriche e dimensioni

Come è possibile ricevere una notifica se gli oggetti non si connettono correttamente ogni giorno?

1. Crea un argomento Amazon SNS denominato `things-not-connecting-successfully` e registra il suo nome Amazon Resource Name (ARN). Questa procedura farà riferimento all'ARN dell'argomento come *`sns-topic-arn`* .

   Per maggiori informazioni su come creare una notifica Amazon SNS, consulta Nozioni di base su Amazon SNS.

2. Crea l'allarme.

```
aws cloudwatch put-metric-alarm \
    --alarm-name ConnectSuccessAlarm \
    --alarm-description "Alarm when my Things don't connect successfully" \
    --namespace AWS/IoT \
    --metric-name Connect.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Testa l'allarme.

```
aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

4. Verificare che l'allarme venga visualizzato nella console CloudWatch .

Come è possibile ricevere una notifica se gli oggetti non pubblicano dati ogni giorno?

1. Crea un argomento Amazon SNS denominato `things-not-publishing-data` e registra il suo nome Amazon Resource Name (ARN). Questa procedura farà riferimento all'ARN dell'argomento come *`sns-topic-arn`* .

   Per maggiori informazioni su come creare una notifica Amazon SNS, consulta Nozioni di base su Amazon SNS.

2. Crea l'allarme.

```
aws cloudwatch put-metric-alarm \
    --alarm-name PublishInSuccessAlarm\
    --alarm-description "Alarm when my Things don't publish their data \
    --namespace AWS/IoT \
    --metric-name PublishIn.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Testa l'allarme.

```
aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

4. Verificare che l'allarme venga visualizzato nella console CloudWatch .

Come è possibile ricevere una notifica se gli aggiornamenti alle copie shadow degli oggetti vengono rifiutati ogni giorno?

1. Crea un argomento Amazon SNS denominato `things-shadow-updates-rejected` e registra il suo nome Amazon Resource Name (ARN). Questa procedura farà riferimento all'ARN dell'argomento come *`sns-topic-arn`* .

Per maggiori informazioni su come creare una notifica Amazon SNS, consulta Nozioni di base su Amazon SNS.

2. Crea l'allarme.

```
aws cloudwatch put-metric-alarm \
    --alarm-name UpdateThingShadowSuccessAlarm \
    --alarm-description "Alarm when my Things Shadow updates are getting rejected"
 \
    --namespace AWS/IoT \
    --metric-name UpdateThingShadow.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Testa l'allarme.

```
aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --state-
reason "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --state-
reason "initializing" --state-value ALARM
```

4. Verificare che l'allarme venga visualizzato nella console CloudWatch .

Come posso creare un CloudWatch allarme per i lavori?

Il servizio Jobs fornisce CloudWatch metriche per monitorare i lavori. Puoi creare CloudWatch allarmi per monitorarne uno qualsiasi. Parametri processi

Il comando seguente crea un CloudWatch allarme per monitorare il numero totale di esecuzioni di job non riuscite per Job *SampleOTAJob* e avvisa l'utente quando più di 20 esecuzioni di job non sono riuscite. L'allarme monitora il parametro `FailedJobExecutionTotalCount` dei processi controllando il valore riportato ogni 300 secondi. Viene attivato quando un singolo valore segnalato è maggiore di 20, il che significa che ci sono state più di 20 esecuzioni di processi non riuscite

dall'avvio del processo. Quando l'allarme si disattiva, invia una notifica all'argomento Amazon SNS fornito.

```
aws cloudwatch put-metric-alarm \
    --alarm-name TotalFailedJobExecution-SampleOTAJob \
    --alarm-description "Alarm when total number of failed job execution exceeds the
 threshold for SampleOTAJob" \
    --namespace AWS/IoT \
    --metric-name FailedJobExecutionTotalCount \
    --dimensions Name=JobId,Value=SampleOTAJob \
    --statistic Sum \
    --threshold 20 \
    --comparison-operator GreaterThanThreshold \
    --period 300 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:SampleOTAJob-has-too-
many-failed-job-ececutions
```

Il comando seguente crea un CloudWatch allarme per monitorare il numero di esecuzioni di job non riuscite per Job *`SampleOTAJob`* in un determinato periodo. Viene quindi avvisato quando più di cinque esecuzioni di processi non sono riuscite durante tale periodo. L'allarme monitora il parametro `FailedJobExecutionCount` dei processi controllando il valore riportato ogni 3600 secondi. Viene attivato quando un singolo valore segnalato è maggiore di 5, il che significa che ci sono state più di 5 esecuzioni di processi non riuscite nell'ultima ora. Quando l'allarme si disattiva, invia una notifica all'argomento Amazon SNS fornito.

```
aws cloudwatch put-metric-alarm \
    --alarm-name FailedJobExecution-SampleOTAJob \
    --alarm-description "Alarm when number of failed job execution per hour exceeds the
 threshold for SampleOTAJob" \
    --namespace AWS/IoT \
    --metric-name FailedJobExecutionCount \
    --dimensions Name=JobId,Value=SampleOTAJob \
    --statistic Sum \
    --threshold 5 \
    --comparison-operator GreaterThanThreshold \
    --period 3600 \
    --unit Count \
    --evaluation-periods 1 \
```

```
    --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:SampleOTAJob-has-too-
many-failed-job-ececutions-per-hour
```

AWS IoT metriche e dimensioni

Quando interagisci con AWS IoT, il servizio invia metriche e dimensioni a CloudWatch ogni minuto. Puoi utilizzare AWS IoT, utilizzare la CloudWatch console o AWS CLI visualizzare queste metriche.

Per visualizzare le metriche utilizzando la CloudWatch console, apri la CloudWatch console. Nel pannello di navigazione, scegli Parametri quindi scegli Tutti i parametri. Nella scheda Sfoglia, cerca per AWS IoT visualizzare l'elenco delle metriche. I parametri vengono raggruppati prima in base allo spazio dei nomi del servizio e successivamente in base alle diverse combinazioni di dimensioni all'interno di ogni spazio dei nomi.

Per visualizzare le metriche utilizzando AWS CLI, esegui il comando seguente.

```
aws cloudwatch list-metrics --namespace "AWS/IoT"
```

CloudWatch visualizza i seguenti gruppi di metriche per: AWS IoT

- AWS IoT metriche
- AWS IoT Core metriche del fornitore di credenziali
- Parametri di autenticazione
- Certificato del server (OCSP), metriche di fissaggio.
- Parametri delle regole
- Parametri delle operazioni sulle regole
- Parametri specifici per operazioni HTTP
- Parametri del broker di messaggi
- Parametri Device Shadow
- Metriche di registrazione
- Parametri processi
- Parametri di Device Defender Audit
- Parametri di Device Defender Detect
- Parametri di provisioning dei dispositivi

- LoRametriche WAN
- Parametri di indicizzazione del parco istanze
- Dimensioni per i parametri

AWS IoT metriche

Metrica	Description
`AddThingToDynamicThingGroupsFailed`	Il numero di eventi di errore associati all'aggiunta di un oggetto a un gruppo di oggetti dinamici. La dimensione `DynamicThingGroupName` contiene il nome dei gruppi dinamici che non sono riusciti ad aggiungere elementi.
`NumLogBatchesFailedToPublishThrottled`	Il batch singolo di eventi di log la cui pubblicazione non è riuscita a causa di errori di throttling.
`NumLogEventsFailedToPublishThrottled`	Il numero di eventi di log nel batch la cui pubblicazione non è riuscita a causa di errori di throttling.

AWS IoT Core metriche del fornitore di credenziali

Metrica	Description
`CredentialExchangeSuccess`	Il numero di richieste con esito positivo `AssumeRoleWithCertificate` a provider di credenziali AWS IoT Core .

Parametri di autenticazione

> **Note**
>
> Le metriche di autenticazione vengono visualizzate nella CloudWatch console in Protocol Metrics.

Metrica	Description
`Connection.AuthNError`	Il numero di tentativi di connessione AWS IoT Core rifiutati a causa di errori di autenticazione. Questa metrica considera solo le connessioni che inviano una stringa SNI (Server Name Indication) corrispon dente a un endpoint dell'utente. Account AWS Questa metrica include i tentativi di connessione da fonti esterne come strumenti di scansione Internet o attività di sondaggio. La `Protocol` dimensione contiene il protocollo utilizzato per inviare il tentativo di connessione.

Certificato del server (OCSP), metriche di fissaggio.

Metrica	Description
OCSPStapleRecupera data.Success	La risposta OCSP è stata ricevuta ed elaborata correttamente. Questa risposta verrà inclusa durante l'handshake TLS per il dominio configurato. La `DomainConfigurationName` dimensione contiene il nome del dominio configurato con la legatura OCSP del certificato del server abilitata.

Parametri delle regole

Metrica	Description
`ParseError`	Il numero di errori di analisi JSON che si sono verificati in messaggi pubblicati su un argomento che la regola sta ascoltando. La dimensione `RuleName` contiene il nome della regola.
`RuleExecutionThrottled`	Il numero di messaggi sottoposto a throttling dal motore di regole a causa di comportamento dannoso

Metrica	Description
	o perché il numero di messaggi supera il limite di throttling del motore di regole. La dimensione `RuleName` contiene il nome della regola da attivare.
`RuleNotFound`	La regola da attivare non è stata trovata. La dimensione `RuleName` contiene il nome della regola.
`RulesExecuted`	Il numero di AWS IoT regole eseguite.
`TopicMatch`	Il numero di messaggi in arrivo pubblicati su un argomento che la regola sta ascoltando. La dimensione `RuleName` contiene il nome della regola.

Parametri delle operazioni sulle regole

Metrica	Description
`Failure`	Il numero di invocazioni di operazioni sulle regole non riuscite. La dimensione `RuleName` contiene il nome della regola che specifica l'operazione. La dimensione `ActionType` contiene il tipo di azione che è stato invocato.
`Success`	Il numero di invocazioni di operazioni sulle regole riuscite. La dimensione `RuleName` contiene il nome della regola che specifica l'operazione. La dimension e `ActionType` contiene il tipo di azione che è stato invocato.
`ErrorActionFailure`	Il numero di azioni di errore non riuscite. La dimensione `RuleName` contiene il nome della regola che specifica l'operazione. La dimensione `ActionType` contiene il tipo di azione che è stato invocato.

Metrica	Description
`ErrorActionSuccess`	Il numero di azioni di errore riuscite correttamente. La dimensione `RuleName` contiene il nome della regola che specifica l'operazione. La dimensione `ActionType` contiene il tipo di azione che è stato invocato.

Parametri specifici per operazioni HTTP

Metrica	Description
`HttpCode_Other`	Generato se il codice di stato della risposta dal web downstream non service/application è 2xx, 4xx o 5xx.
`HttpCode_4XX`	Generato se il codice di stato della risposta dal Web a valle service/application è compreso tra 400 e 499.
`HttpCode_5XX`	Generato se il codice di stato della risposta dal web downstream service/application è compreso tra 500 e 599.
`HttpInvalidUrl`	Generato se un URL endpoint, dopo la sostituzione dei modelli di sostituzione, non inizia con `https://`.
`HttpRequestTimeout`	Generato se il web downstream service/application non restituisce una risposta entro il limite di timeout della richiesta. Per ulteriori informazioni, consulta Service Quotas.
`HttpUnknownHost`	Generato se l'URL è valido, ma il servizio non esiste o non è raggiungibile.

Parametri del broker di messaggi

> **Note**
>
> Le metriche del broker di messaggi vengono visualizzate nella CloudWatch console in Metriche del protocollo.

Metrica	Description
`Connect.AuthError`	Il numero di richieste di connessione che il broker di messaggi non ha potuto autorizzare. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `CONNECT`.
`Connect.ClientError`	Il numero di richieste di connessione rifiutate perché il messaggio MQTT non soddisfaceva i requisiti definiti in. AWS IoT quote La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `CONNECT`.
`Connect.ClientIDThrottle`	Il numero di richieste di connessione per cui è stato eseguito il throttling perché il client ha superato la velocità di richiesta di connessione consentita per un ID specifico. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `CONNECT`.
`Connect.ServerError`	Il numero di richieste di connessione non riuscite a causa di un errore interno. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `CONNECT`.
`Connect.Success`	Il numero di connessioni con esito positivo al broker di messaggi. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `CONNECT`.

Metrica	Description
`Connect.Throttle`	Il numero di richieste di connessione per cui è stato eseguito il throttling perché l'account ha superato la velocità di richiesta di connessione consentita. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `CONNECT`.
`Ping.Success`	Il numero di messaggi ping ricevuti dal broker di messaggi. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio ping.
`PublishIn.AuthError`	Il numero di richieste di pubblicazione che il broker di messaggi non ha potuto autorizzare. La dimensione `Protocol` contiene il protocollo utilizzato per pubblicare il messaggio. HTTP Publish non supporta questa metrica.
`PublishIn.ClientError`	Il numero di richieste di pubblicazione rifiutate dal broker di messaggi perché il messaggio non soddisfaceva i requisiti definiti in AWS IoT quote. La dimensione `Protocol` contiene il protocollo utilizzato per pubblicare il messaggio.
`PublishIn.ServerError`	Il numero di richieste di pubblicazione che il broker di messaggi non è riuscito a elaborare a causa di un errore interno. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`.
`PublishIn.Success`	Il numero di richieste di pubblicazione correttamente elaborate dal broker di messaggi. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`.

Metrica	Description
`PublishIn.Throttle`	Il numero di richieste di pubblicazione per cui è stato eseguito il throttling perché il client ha superato la velocità di messaggi in arrivo consentita. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`. HTTP Publish non supporta questa metrica.
`PublishOut.AuthError`	Il numero di richieste di pubblicazione effettuate dal broker di messaggi che AWS IoT non ha potuto autorizzare. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`.
`PublishOut.ClientError`	Il numero di richieste di pubblicazione effettuate dal broker di messaggi che sono state rifiutate perché il messaggio non soddisfaceva i requisiti definiti in AWS IoT quote. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`.
`PublishOut.Success`	Il numero di richieste di pubblicazione correttamente effettuate dal broker di messaggi. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`.
`PublishOut.Throttle`	Il numero di richieste di pubblicazione limitate perché il client ha superato la velocità di messaggi in arrivo consentita. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`.
`PublishRetained.AuthError`	Il numero di richieste di pubblicazione con il flag `RETAIN` segnala che il broker di messaggi non ha potuto fornire l'autorizzazione. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`.

Metrica	Description
`PublishRetained.ServerError`	Il numero di richieste di pubblicazione memorizzate che il broker di messaggi non è riuscito a elaborare a causa di un errore interno. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`.
`PublishRetained.Success`	Il numero di richieste di pubblicazione con il flag `RETAIN` segnala che il broker di messaggi le ha elaborate correttamente. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`.
`PublishRetained.Throttle`	Il numero di richieste di pubblicazione con il flag `RETAIN` segnala che hanno subito una limitazione, perché il client ha superato la velocità di messaggi in arrivo consentita. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `PUBLISH`.
`Queued.Success`	Il numero di messaggi archiviati che sono stati correttamente elaborati dal broker di messaggi per i client scollegati dalle sessioni persistenti. I messaggi con un QoS pari a 1 vengono archiviati mentre un client con una sessione persistente viene scollegato.
`Queued.Throttle`	Il numero di messaggi che non è stato possibile archiviare e che sono stati limitati mentre i client con sessioni persistenti venivano scollegati. Ciò si verifica quando i client superano il limite di messaggi in coda al secondo per account. I messaggi con un QoS pari a 1 vengono archiviati mentre un client con una sessione persistente viene scollegato.

Metrica	Description
`Queued.ServerError`	Il numero di messaggi che non sono stati archiviati per una sessione persistente a causa di un errore interno. Quando i client con una sessione persistente vengono scollegati, i messaggi con una qualità del servizio (QoS) pari a 1 vengono archiviati.
`Subscribe.AuthError`	Il numero di richieste di sottoscrizione effettuate da un client, che non è stato possibile autorizzare. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `SUBSCRIBE` .
`Subscribe.ClientError`	Il numero di richieste di sottoscrizione che sono state rifiutate perché il `SUBSCRIBE` messaggio non soddisfaceva i requisiti definiti in AWS IoT quote. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `SUBSCRIBE` .
`Subscribe.ServerError`	Il numero di richieste di sottoscrizione respinte a causa di un errore interno. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `SUBSCRIBE` .
`Subscribe.Success`	Il numero di richieste di sottoscrizione correttamente elaborate dal broker di messaggi. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `SUBSCRIBE` .

Metrica	Description
`Subscribe.Throttle`	Il numero di richieste di iscrizione che sono state limitate perché i limiti di tariffa consentiti per le richieste di iscrizione sono stati superati per le tue. Account AWS Questi limiti includono gli abbonamenti al secondo per account, gli abbonamenti per account e gli abbonamenti per connessione descritti nel broker di AWS IoT Core messaggi e nei limiti e nelle quote del protocollo. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `SUBSCRIBE` .
`Throttle.Exceeded`	Questa metrica viene visualizzata CloudWatch quando un client MQTT è limitato ai pacchetti al secondo per i limiti del livello di connessione. Questo parametro non si applica alle connessioni HTTP.
`Unsubscribe.ClientError`	Il numero di richieste di annullamento dell'iscrizione che sono state rifiutate perché il `UNSUBSCRIBE` messaggio non soddisfaceva i requisiti definiti in. AWS IoT quote La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `UNSUBSCRIBE` .
`Unsubscribe.ServerError`	Il numero di richieste di annullamento della sottoscrizione respinte a causa di un errore interno. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `UNSUBSCRIBE` .
`Unsubscribe.Success`	Il numero di richieste di annullamento della sottoscrizione correttamente elaborate dal broker di messaggi. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `UNSUBSCRIBE` .

Metrica	Description
`Unsubscribe.Throttle`	Il numero di richieste di annullamento della sottoscrizione respinte perché il client ha superato la velocità di richiesta di annullamento della sottoscrizione consentita. La dimensione `Protocol` contiene il protocollo utilizzato per inviare il messaggio `UNSUBSCRIBE` .

Parametri Device Shadow

> **Note**
>
> Le metriche relative all'ombra del dispositivo vengono visualizzate nella CloudWatch console sotto Protocol Metrics.

Metrica	Description
`DeleteThingShadow.Accepted`	Numero di richieste `DeleteThingShadow` elaborate correttamente. La dimensione `Protocol` contiene il protocollo utilizzato per effettuare la richiesta.
`GetThingShadow.Accepted`	Numero di richieste `GetThingShadow` elaborate correttamente. La dimensione `Protocol` contiene il protocollo utilizzato per effettuare la richiesta.
`ListThingShadow.Accepted`	Numero di richieste `ListThingShadow` elaborate correttamente. La dimensione `Protocol` contiene il protocollo utilizzato per effettuare la richiesta.
`UpdateThingShadow.Accepted`	Numero di richieste `UpdateThingShadow` elaborate correttamente. La dimensione `Protocol` contiene il protocollo utilizzato per effettuare la richiesta.

Metriche di registrazione

Metrica	Description
`CloudwatchLogs:LogGroupCreationFailed`	Il numero di `CreateLogGroup` richieste non riuscite. AWS IoT Logging crea un gruppo di CloudWatch log se non esiste. La `LogGroup` dimensione contiene il gruppo di log utilizzato per effettuare la richiesta.
`CloudwatchLogs:LogStreamCreationFailed`	Il numero di `CreateLogStream` richieste non riuscite. AWS IoT Logging crea flussi di CloudWatch log per inviare eventi di log. La `LogGroup` dimension e contiene il gruppo di log utilizzato per effettuare la richiesta.
`CloudwatchLogs:PutRetentionPolicyFailed`	Il numero di `PutRetentionPolicy` richieste non riuscite. AWS IoT Logging crea un gruppo di CloudWatch log se non esiste e la policy di conservazione è impostata su 30 giorni. La `LogGroup` dimensione contiene il gruppo di log utilizzato per effettuare la richiesta.
`CloudwatchLogs:PutLogEventsFailed`	Il numero di `PutLogEvents` richieste non riuscite. La `LogGroup` dimensione contiene il gruppo di CloudWatch log utilizzato per effettuare la richiesta.

Parametri processi

Metrica	Description
`CanceledJobExecutionCount`	Il numero di esecuzioni di job il cui stato è cambiato `CANCELED` entro un periodo di tempo determina to da CloudWatch. (Per ulteriori informazioni sui CloudWatch parametri, consulta Amazon CloudWatc

Metrica	Description
	h Metrics.) La dimensione `JobId` contiene l'ID del processo.
`CanceledJobExecutionTotalCount`	Il numero totale di esecuzioni di processi il cui stato è `CANCELED` per il dato processo. La dimensione `JobId` contiene l'ID del processo.
`ClientErrorCount`	Numero di errori client generati durante l'esecuzione del processo. La dimensione `JobId` contiene l'ID del processo.
`FailedJobExecutionCount`	Il numero di esecuzioni di job il cui stato è cambiato `FAILED` entro un periodo di tempo determinato da. CloudWatch (Per ulteriori informazioni sui CloudWatch parametri, consulta Amazon CloudWatch Metrics.) La dimensione `JobId` contiene l'ID del processo.
`FailedJobExecutionTotalCount`	Il numero totale di esecuzioni di processi il cui stato è `FAILED` per il dato processo. La dimensione `JobId` contiene l'ID del processo.
`InProgressJobExecutionCount`	Il numero di esecuzioni di job il cui stato è cambiato `IN_PROGRESS` entro un periodo di tempo determinato da. CloudWatch (Per ulteriori informazioni sui CloudWatch parametri, consulta Amazon CloudWatch Metrics.) La dimensione `JobId` contiene l'ID del processo.
`InProgressJobExecutionTotalCount`	Il numero totale di esecuzioni di processi il cui stato è `IN_PROGRESS` per il dato processo. La dimensione `JobId` contiene l'ID del processo.
`RejectedJobExecutionTotalCount`	Il numero totale di esecuzioni di processi il cui stato è `REJECTED` per il dato processo. La dimensione `JobId` contiene l'ID del processo.

Metrica	Description
`RemovedJobExecutionTotalCount`	Il numero totale di esecuzioni di processi il cui stato è `REMOVED` per il dato processo. La dimensione `JobId` contiene l'ID del processo.
`QueuedJobExecutionCount`	Il numero di esecuzioni di job il cui stato è cambiato `QUEUED` entro un periodo di tempo determinato da. CloudWatch (Per ulteriori informazioni sui CloudWatch parametri, consulta Amazon CloudWatch Metrics.) La dimensione `JobId` contiene l'ID del processo.
`QueuedJobExecutionTotalCount`	Il numero totale di esecuzioni di processi il cui stato è `QUEUED` per il dato processo. La dimensione `JobId` contiene l'ID del processo.
`RejectedJobExecutionCount`	Il numero di esecuzioni di job il cui stato è cambiato `REJECTED` entro un periodo di tempo determinato da. CloudWatch (Per ulteriori informazioni sui CloudWatch parametri, consulta Amazon CloudWatch Metrics.) La dimensione `JobId` contiene l'ID del processo.
`RemovedJobExecutionCount`	Il numero di esecuzioni di job il cui stato è cambiato `REMOVED` entro un periodo di tempo determinato da. CloudWatch (Per ulteriori informazioni sui CloudWatch parametri, consulta Amazon CloudWatch Metrics.) La dimensione `JobId` contiene l'ID del processo.
`ServerErrorCount`	Numero di errori server generati durante l'esecuzione del processo. La dimensione `JobId` contiene l'ID del processo.
`SuccededJobExecutionCount`	Il numero di esecuzioni di job il cui stato è cambiato `SUCCESS` entro un periodo di tempo determinato da. CloudWatch (Per ulteriori informazioni sui CloudWatch parametri, consulta Amazon CloudWatch Metrics.) La dimensione `JobId` contiene l'ID del processo.

Metrica	Description
`SucceededJobExecutionTotalCount`	Il numero totale di esecuzioni di processi il cui stato è `SUCCESS` per il dato processo. La dimensione `JobId` contiene l'ID del processo.

Parametri di Device Defender Audit

Metrica	Description
`NonCompliantResources`	Il numero di risorse risultate non conformi a un controllo. Il sistema rileva il numero di risorse risultate non conformi per ogni controllo di ogni audit eseguito.
`ResourcesEvaluated`	Il numero di risorse valutate per la conformità. Il sistema rileva il numero di risorse valutate per ogni controllo di ogni audit eseguito.
`MisconfiguredDeviceDefenderNotification`	Ti avvisa quando la configurazione SNS di non è configurata correttamente. AWS IoT Device Defender Dimensioni

Parametri di Device Defender Detect

Metrica	Description
`NumOfMetricsExported`	Il numero di metriche esportate per una metrica lato cloud, lato dispositivo o personalizzata. Il sistema riporta il numero di metriche esportate per l'account, per una metrica specifica. Questa metrica è disponibi le solo per i clienti che utilizzano l'esportazione delle metriche.

Metrica	Description
NumOfMetricsSkipped	Il numero di metriche ignorate per una metrica lato cloud, lato dispositivo o personalizzata. Il sistema riporta il numero di metriche ignorate per l'account, per una metrica specifica a causa delle autorizzazioni insufficienti fornite a Device Defender Detect per la pubblicazione sull'argomento mqtt. Questa metrica è disponibile solo per i clienti che utilizzano l'esporta zione delle metriche.
NumOfMetricsExceedingSizeLi mit	Il numero di metriche ignorate per l'esportazione per una metrica lato cloud, lato dispositivo o personali zzata a causa della dimensione che supera i limiti di dimensione dei messaggi MQTT. Il sistema riporta il numero di metriche ignorate per l'esportazione per l'account, per una metrica specifica a causa della dimensione che supera i limiti di dimensione dei messaggi MQTT. Questa metrica è disponibile solo per i clienti che utilizzano l'esportazione delle metriche.
Violations	Il numero di nuove violazioni di comportamenti per il profilo di sicurezza rilevate dall'ultima valutazio ne completata. Il sistema rileva il numero di nuove violazioni per l'account per un profilo di sicurezza specifico e per un comportamento specifico di un profilo di sicurezza specifico.
ViolationsCleared	Il numero di violazioni di comportamenti per il profilo di sicurezza risolte dall'ultima valutazione completat a. Il sistema rileva il numero di violazioni risolte per l'account per un profilo di sicurezza specifico e per un comportamento specifico di un profilo di sicurezza specifico.

Metrica	Description
`ViolationsInvalidated`	Il numero di violazioni di comportamenti per il profilo di sicurezza per cui le informazioni non sono più disponibili dall'ultima valutazione completata (perché il dispositivo ha interrotto la reportistica o perché, per qualsiasi motivo, non viene più effettuato il monitoraggio). Il sistema rileva il numero di violazioni non convalidate per l'intero account per un profilo di sicurezza specifico e per un comportamento specifico di un profilo di sicurezza specifico.
`MisconfiguredDeviceDefenderNotification`	Ti avvisa quando la configurazione SNS di non è configurata correttamente. AWS IoT Device Defender Dimensioni

Parametri di provisioning dei dispositivi

AWS IoT Metriche di approvvigionamento della flotta

Metrica	Description
`ApproximateNumberOfThingsRegistered`	Il numero di oggetti registrati tramite provisioning del parco istanze. Mentre il conteggio è generalmente accurato, l'architettura distribuita di AWS IoT Core rende difficile mantenere un conteggio preciso degli oggetti registrati. La statistica da utilizzare per questo parametro è: • Max per segnalare il numero totale di elementi registrati. Per un numero di elementi registrati durante la finestra di CloudWatch aggregazione, consulta la metrica. `RegisterThingFailed`

Metrica	Description
	Dimensioni: ClaimCertificateId
`CreateKeysAndCertificateFailed`	Il numero di errori che si sono verificati chiamando l'API MQTT `CreateKeysAndCertificate` . La metrica viene emessa in entrambi i casi Success (valore = 0) e Failure (valore = 1). Questa metrica può essere utilizzata per tenere traccia del numero di certificati creati e registrati durante le finestre di aggregazione CloudWatch supportate, ad esempio 5 minuti o 1 ora. Le statistiche disponibili per questo parametro sono: • Sum per segnalare il numero di chiamate non riuscite. • SampleCountper segnalare il numero totale di chiamate riuscite e non riuscite.
`CreateCertificateFromCsrFailed`	Il numero di errori che si sono verificati chiamando l'API MQTT `CreateCertificateFromCsr` . La metrica viene emessa in entrambi i casi Success (valore = 0) e Failure (valore = 1). Questa metrica può essere utilizzata per tenere traccia del numero di elementi registrati durante le finestre CloudWatch di aggregazione supportate, ad esempio 5 minuti o 1 ora. Le statistiche disponibili per questo parametro sono: • Sum per segnalare il numero di chiamate non riuscite. • SampleCountper segnalare il numero totale di chiamate riuscite e non riuscite.

Metrica	Description
`RegisterThingFailed`	Il numero di errori che si sono verificati chiamando l'API MQTT `RegisterThing` . La metrica viene emessa in entrambi i casi Success (valore = 0) e Failure (valore = 1). Questa metrica può essere utilizzata per tenere traccia del numero di elementi registrati durante le finestre CloudWatch di aggregazione supportate, ad esempio 5 minuti o 1 ora. Per il numero totale di elementi registrati, consulta il parametro `ApproximateNumberOfThingsRegistered` . Le statistiche disponibili per questo parametro sono: • Sum per segnalare il numero di chiamate non riuscite. • SampleCountper segnalare il numero totale di chiamate riuscite e non riuscite. Dimensioni: TemplateName

Just-in-time metriche di approvvigionamento

Metrica	Description
`ProvisionThing.ClientError`	Il numero di provisioning non riusciti di un dispositivo a causa di un errore del client. Ad esempio, la policy specificata nel modello non esisteva.
`ProvisionThing.ServerError`	Il numero di provisioning non riusciti di un dispositivo a causa di un errore del server. I clienti possono riprovare a eseguire il provisioning del dispositivo dopo l'attesa e possono contattare AWS IoT se il problema rimane lo stesso.

Metrica	Description
`ProvisionThing.Success`	Il numero di provisioning eseguiti correttamente per un dispositivo.

LoRametriche WAN

La tabella seguente mostra le metriche per. AWS IoT Core per LoRaWAN Per ulteriori informazioni, consulta le AWS IoT Core per LoRaWAN metriche.

AWS IoT Core per LoRaWAN metriche

Metrica	Description
Dispositivi/gateway attivi	Il numero di dispositivi e gateway LoRa WAN attivi nel tuo account.
Numero di messaggi in uplink	Il numero di messaggi uplink inviati entro un periodo di tempo specificato per tutti i gateway e i dispositivi attivi del tuo. Account AWS I messaggi uplink sono messaggi inviati dal dispositivo a. AWS IoT Core per LoRaWAN
Conteggio di messaggi di downlink	Il numero di messaggi di downlink inviati entro un periodo di tempo specificato per tutti i gateway e i dispositivi attivi nel tuo. Account AWS I messaggi di downlink sono messaggi inviati dal tuo dispositivo AWS IoT Core per LoRaWAN .
Tasso di perdita di messaggi	Dopo aver aggiunto il dispositivo e aver effettuato la connessione AWS IoT Core per LoRaWAN, il dispositivo può avviare un messaggio di uplink per iniziare a scambiare messaggi con il cloud. Puoi utilizzare questa metrica per monitorare quindi la frequenza di messaggi in uplink che vengono persi.
Unisci le metriche	Dopo aver aggiunto il dispositivo e il gateway, esegui una procedura di unione in modo che il dispositi

Metrica	Description
	vo possa inviare dati in uplink e comunicare con. AWS IoT Core per LoRaWAN Puoi utilizzare questa metrica per ottenere informazioni sulle metriche di accesso per tutti i dispositivi attivi nel tuo Account AWS.
Indicatore della potenza media del segnale ricevuto (RSSI)	È possibile utilizzare questa metrica per monitorar e l'RSSI (indicatore di potenza del segnale ricevuto) medio entro il periodo di tempo specificato. L'RSSI è una misura che indica se il segnale è sufficien temente forte per una buona connessione wireless. Questo valore è negativo e deve essere più vicino allo zero per una connessione stabile.
Rapporto medio segnale/rumore (SNR)	È possibile utilizzare questa metrica per monitorar e l'SNR (Signal-to-noise rapporto) medio entro la durata di tempo specificata. L'SNR è una misura che indica se il segnale ricevuto è sufficientemente forte rispetto al livello di rumore per una buona connessio ne wireless. Il valore SNR è positivo e deve essere maggiore di zero per indicare che la potenza del segnale è più forte della potenza del rumore.
Disponibilità del gateway	È possibile utilizzare questa metrica per ottenere informazioni sulla disponibilità di questo gateway entro un periodo di tempo specificato. Questa metrica mostra il tempo di connessione websocket di questo gateway per un periodo di tempo specificato.

Just-in-time metriche di approvvigionamento

Metrica	Description
`ProvisionThing.ClientError`	Il numero di provisioning non riusciti di un dispositivo a causa di un errore del client. Ad esempio, la policy specificata nel modello non esisteva.
`ProvisionThing.ServerError`	Il numero di provisioning non riusciti di un dispositivo a causa di un errore del server. I clienti possono riprovare a eseguire il provisioning del dispositivo dopo l'attesa e possono contattare AWS IoT se il problema rimane lo stesso.
`ProvisionThing.Success`	Il numero di provisioning eseguiti correttamente per un dispositivo.

Parametri di indicizzazione del parco istanze

AWS IoT metriche di indicizzazione della flotta

Metrica	Description
`NamedShadowCountForDynamicGroupQueryLimitExceeded`	Viene elaborato un massimo di 25 copie shadow denominate per oggetto per termini di query che non sono specifici dell'origine dei dati nei gruppi di oggetti dinamici. Quando questo limite viene violato, il tipo di evento `NamedShadowCountForDynamicGroupQueryLimitExceeded` verrà emesso.

Dimensioni per i parametri

I parametri utilizzano lo spazio dei nomi forniscono i parametri per le seguenti dimensioni:

Dimensione	Description
`ActionType`	Il tipo di operazione specificato dalla regola attivata dalla richiesta.

Dimensione	Description
`BehaviorName`	Il nome del comportamento del profilo di sicurezza Device Defender Detect che viene monitorato.
`ClaimCertificateId`	Il `certificateId` del reclamo utilizzato per il provisioning dei dispositivi.
`CheckName`	Il nome del controllo Device Defender Audit i cui risultati vengono monitorati.
`JobId`	L'ID del lavoro di cui viene monitorato l'avanzamento o la connessione success/failure ai messaggi.
`Protocol`	Il protocollo usato per effettuare la richiesta. I valori validi sono: MQTT o HTTP
`RuleName`	Il nome della regola attivata dalla richiesta.
`ScheduledAuditName`	Il nome del Device Defender Audit i cui risultati di controllo vengono monitorati. Ha il valore `0nDemand` se i risultati riportati si riferiscono a un audit effettuato on demand.
`SecurityProfileName`	Il nome del profilo di sicurezza Device Defender Detect i cui comportamenti vengono monitorati.
`TemplateName`	Il nome del modello dell'artefatto di provisioning.
`SourceArn`	Fa riferimento al profilo di sicurezza per il rilevamento o all'account arn per il controllo.
`RoleArn`	Si riferisce al ruolo che Device Defender ha cercato di assumere.
`TopicArn`	Fa riferimento all'argomento SNS su cui Device Defender ha tentato di pubblicare.

Dimensione	Description
`Error`	Fornisce una breve descrizione dell'errore ricevuto durante il tentativo di pubblicazione sull'argomento SNS. I valori possibili sono: • "KMSKeyNotFound«: indica che la chiave KMS non esiste per l'argomento. • "InvalidTopicName«: indica che l'argomento SNS non è valido. • «KMSAccessNegato»: indica che il ruolo non dispone delle autorizzazioni per la chiave KMS per l'argomento. • "AuthorizationError«: indica che il ruolo fornito non autorizza Device Defender a pubblicare sull'argo mento SNS. • "SNSTopicNotFound«: indica che l'argomento SNS fornito non esiste. • "FailureToAssumeRole«: indica che il ruolo fornito non autorizza Device Defender ad assumere il ruolo. • "CrossRegionSNSTopic«: indica che l'argomento SNS esiste in una regione diversa.

Monitora AWS IoT utilizzando i log CloudWatch

Quando AWS IoT la registrazione è abilitata, AWS IoT invia gli eventi di avanzamento relativi a ciascun messaggio mentre passa dai tuoi dispositivi tramite il broker di messaggi e il motore delle regole. Nella CloudWatch console, CloudWatch i log vengono visualizzati in un gruppo di log denominato AWSIot Logs.

Per ulteriori informazioni sui CloudWatch registri, vedere Registri. CloudWatch Per informazioni sui AWS IoT CloudWatch log supportati, vedere. CloudWatch Registra le voci di registro AWS IoT

Visualizzazione dei AWS IoT log nella console CloudWatch

> **Note**
>
> Il gruppo di `AWSIotLogsV2` log non è visibile nella CloudWatch console fino a quando:
>
> - Hai abilitato l'accesso. AWS IoT Per maggiori informazioni su come abilitare l'accesso, vedi AWS IoTConfigurare la registrazione AWS IoT
> - Alcune voci di registro sono state scritte dalle AWS IoT operazioni.

Per visualizzare AWS IoT i log nella console CloudWatch

1. Vai a https://console.aws.amazon.com/cloudwatch/. Nel pannello di navigazione, selezionare Log groups (Gruppi di log).
2. Nella casella di testo Filter (Filtra), inserisci **`AWSIotLogsV2`** e premi Invio.
3. Fai doppio clic sul gruppo di log `AWSIotLogsV2`.
4. Seleziona Cerca tutto. Viene visualizzato un elenco completo dei AWS IoT log generati per l'account.
5. Scegli l'icona di espansione per esaminare un singolo flusso.

È possibile anche inserire una query nella casella di testo Filter events (Filtra eventi). Di seguito sono illustrate alcune query interessanti da provare:

- `{ $.logLevel = "INFO" }`

  Trova tutti i log con livello `INFO`.
- `{ $.status = "Success" }`

  Trova tutti i log con stato `Success`.
- `{ $.status = "Success" && $.eventType = "GetThingShadow" }`

  Trova tutti i log con stato `Success` e un tipo di evento `GetThingShadow`.

Per ulteriori informazioni sulla creazione di espressioni di filtro, vedere CloudWatch Logs Queries.

CloudWatch Registra le voci di registro AWS IoT

Ogni componente di AWS IoT genera le proprie voci di registro. Ogni voce di registro ha un `eventType` che specifica l'operazione che ha causato la generazione della voce di registro. In questa sezione vengono descritti le voci di registro generate dai seguenti componenti di AWS IoT .

Argomenti

- Voci di registro del broker messaggi
- Voci di registro OCSP del certificato del server
- Voci del registro Device Shadow
- Voci di registro del motore delle regole
- Voci del registro processi
- Voci del registro di provisioning dei dispositivi
- Voci del registro del gruppo di oggetti dinamici
- Voci di registro di indicizzazione del parco istanze
- attributi Common CloudWatch Logs

Voci di registro del broker messaggi

Il broker di AWS IoT messaggi genera voci di registro per i seguenti eventi:

Argomenti

- Connetti voce di registro
- Disconnetti voce registro
- DeleteConnection voce di registro
- GetRetainedMessage voce di registro
- ListRetainedMessage registrazione del registro
- Voce di registro di pubblicazione
- Voce di registro di pubblicazione
- Voce di log Queued
- Sottoscrivi voce di registro
- Annulla l'iscrizione al registro

Connetti voce di registro

Il broker di AWS IoT messaggi genera una voce di registro con un `eventType` di `Connect` quando un client MQTT si connette.

Esempio di voce di registro di connessione

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Connect",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Oltre a[attributi Common CloudWatch Logs](), le voci di `Connect` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

principalId

ID dell'entità principale da cui proviene la richiesta.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

sourceIp

Indirizzo IP da cui ha avuto origine la richiesta.

sourcePort

Porta da cui ha avuto origine la richiesta.

Disconnetti voce registro

Il broker di AWS IoT messaggi genera una voce di registro con un `eventType` di `Disconnect` quando un client MQTT si disconnette.

Esempio di voce di registro di disconnessione

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID",
    "disconnectReason": "CLIENT_INITIATED_DISCONNECT"
}
```

Oltre a[attributi Common CloudWatch Logs](), le voci di `Disconnect` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

principalId

ID dell'entità principale da cui proviene la richiesta.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

sourceIp

Indirizzo IP da cui ha avuto origine la richiesta.

sourcePort

Porta da cui ha avuto origine la richiesta.

motivo

Il motivo per cui il client viene disconnesso.

details

Una breve spiegazione dell'errore.

disconnectReason

Il motivo per cui il client viene disconnesso.

DeleteConnection voce di registro

Il broker di AWS IoT messaggi genera una voce di registro con un `eventType` di `DeleteConnection` quando viene eliminata una connessione client MQTT.

DeleteConnection esempio di immissione di registro

```
{
    "timestamp": "2025-08-09 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DeleteConnection",
    "protocol": "HTTP",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `DeleteConnection` registro contengono i seguenti attributi:

clientId

L'ID del client che verrà disconnesso.

principalId

ID dell'entità principale da cui proviene la richiesta. Per informazioni su come identificare il principale utilizzando l'ID principale, consulta Confrontare le identità e le credenziali IAM nella Guida per l'utente IAM.

protocol

Il protocollo usato per effettuare la richiesta. Il valore valido è HTTP.

sourceIp

Indirizzo IP da cui ha avuto origine la richiesta.

sourcePort

Porta da cui ha avuto origine la richiesta.

GetRetainedMessage voce di registro

Il broker di AWS IoT messaggi genera una voce di registro con una chiamata `eventType` `GetRetainedMessage`di `GetRetainedMessage` quando.

GetRetainedMessage esempio di immissione di registro

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetRetainedMessage",
    "protocol": "HTTP",
    "topicName": "a/b/c",
    "qos": "1",
    "lastModifiedDate": "2017-08-07 18:47:56.664"
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `GetRetainedMessage` registro contengono i seguenti attributi:

lastModifiedDate

Data e ora dell'epoca, in millisecondi, in cui il messaggio conservato è stato archiviato da. AWS IoT

protocol

Il protocollo usato per effettuare la richiesta. Valore valido: `HTTP`.

qos

Il livello di qualità del servizio (QoS) utilizzato nella richiesta di pubblicazione. I valori validi sono `0` e `1`.

topicName

Nome dell'argomento sottoscritto.

ListRetainedMessage registrazione del registro

Il broker di AWS IoT messaggi genera una voce di registro con una chiamata `eventType` [`ListRetainedMessages`](#)di `ListRetainedMessage` quando.

ListRetainedMessage esempio di immissione di registro

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "ListRetainedMessage",
    "protocol": "HTTP"
}
```

Oltre a[attributi Common CloudWatch Logs](#), le voci di `ListRetainedMessage` registro contengono il seguente attributo:

protocol

Il protocollo usato per effettuare la richiesta. Valore valido: `HTTP`.

Voce di registro di pubblicazione

Quando il broker di AWS IoT messaggi riceve un messaggio MQTT, genera una voce di registro con un `eventType` of`Publish-In`.

Esempio di voce di registro di pubblicazione

```
{
        "timestamp": "2017-08-10 15:39:30.961",
        "logLevel": "INFO",
        "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
        "accountId": "123456789012",
        "status": "Success",
        "eventType": "Publish-In",
        "protocol": "MQTT",
        "topicName": "$aws/things/MyThing/shadow/get",
        "clientId": "abf27092886e49a8a5c1922749736453",
        "principalId":
 "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
        "sourceIp": "205.251.233.181",
        "sourcePort": 13490,
        "retain": "True"
    }
```

Oltre a[attributi Common CloudWatch Logs](), le voci di `Publish-In` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

principalId

ID dell'entità principale da cui proviene la richiesta.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

retain

L'attributo utilizzato quando un messaggio ha il flag RETAIN impostato con un valore di `True`. Se nel messaggio non è impostato il flag RETAIN, questo attributo non viene visualizzato nella voce di registro. Per ulteriori informazioni, consulta Messaggi conservati da MQTT.

sourceIp

Indirizzo IP da cui ha avuto origine la richiesta.

sourcePort

Porta da cui ha avuto origine la richiesta.

topicName

Nome dell'argomento sottoscritto.

Voce di registro di pubblicazione

Quando il broker di messaggi pubblica un messaggio MQTT, genera un log `eventType` di `Publish-Out`.

Esempio di voce di registro di pubblicazione

```
{
    "timestamp": "2017-08-10 15:39:30.961",
    "logLevel": "INFO",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Publish-Out",
    "protocol": "MQTT",
    "topicName": "$aws/things/MyThing/shadow/get",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Oltre a[attributi Common CloudWatch Logs](), le voci di `Publish-Out` registro contengono i seguenti attributi:

clientId

L'ID del client sottoscritto che riceve messaggi su quell'argomento MQTT.

principalId

ID dell'entità principale da cui proviene la richiesta.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

sourceIp

Indirizzo IP da cui ha avuto origine la richiesta.

sourcePort

Porta da cui ha avuto origine la richiesta.

topicName

Nome dell'argomento sottoscritto.

Voce di log Queued

Quando un dispositivo con una sessione persistente viene disconnesso, il broker di messaggi MQTT archivia i messaggi del dispositivo e AWS IoT genera voci di registro con un EventType di. `Queued` Per ulteriori informazioni sulle sessioni persistenti MQTT, vedere. Sessioni persistenti MQTT

Esempio di voce del log errori del server Queued

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "ERROR",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
    "protocol": "MQTT",
    "eventType": "Queued",
    "status": "Failure",
    "details": "Server Error"
}
```

Oltre aattributi Common CloudWatch Logs, le voci del registro degli errori del `Queued` server contengono i seguenti attributi:

clientId

L'ID del client in cui il messaggio viene accodato.

details

`Server Error`

Un errore del server ha impedito l'archiviazione del messaggio.

protocol

Il protocollo usato per effettuare la richiesta. Il valore sarà sempre `MQTT`.

qos

Il livello di qualità del servizio (QoS) della richiesta. Il valore sarà sempre 1 perché i messaggi con QoS pari a 0 non vengono archiviati.

topicName

Nome dell'argomento sottoscritto.

Esempio di voce di log delle operazioni riuscite Queued

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "INFO",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
    "protocol": "MQTT",
    "eventType": "Queued",
    "status": "Success"
}
```

Oltre a[attributi Common CloudWatch Logs](), le voci del registro di `Queued` successo contengono i seguenti attributi:

clientId

L'ID del client in cui il messaggio viene accodato.

protocol

Il protocollo usato per effettuare la richiesta. Il valore sarà sempre `MQTT`.

qos

Il livello di qualità del servizio (QoS) della richiesta. Il valore sarà sempre 1 perché i messaggi con QoS pari a 0 non vengono archiviati.

topicName

Nome dell'argomento sottoscritto.

Esempio di voce di log con limitazione (della larghezza di banda della rete) Queued

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "ERROR",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
    "protocol": "MQTT",
    "eventType": "Queued",
    "status": "Failure",
    "details": "Throttled while queueing offline message"
}
```

Oltre aattributi Common CloudWatch Logs, le voci `Queued` di registro limitate contengono i seguenti attributi:

clientId

L'ID del client in cui il messaggio viene accodato.

details

`Throttled while queueing offline message`

Il client ha superato il limite `Queued messages per second per account` , pertanto il messaggio non è stato archiviato.

protocol

Il protocollo usato per effettuare la richiesta. Il valore sarà sempre `MQTT`.

qos

Il livello di qualità del servizio (QoS) della richiesta. Il valore sarà sempre 1 perché i messaggi con QoS pari a 0 non vengono archiviati.

topicName

Nome dell'argomento sottoscritto.

Sottoscrivi voce di registro

Il broker di AWS IoT messaggi genera una voce di registro con un `eventType` di `Subscribe` quando un client MQTT sottoscrive un argomento.

Esempio di voce di log MQTT 3 Subscribe

```
{
    "timestamp": "2017-08-10 15:39:04.413",
    "logLevel": "INFO",
    "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Subscribe",
    "protocol": "MQTT",
    "topicName": "$aws/things/MyThing/shadow/#",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `Subscribe` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

principalId

ID dell'entità principale da cui proviene la richiesta.

protocol

Il protocollo usato per effettuare la richiesta. Il valore sarà sempre `MQTT`.

sourceIp

Indirizzo IP da cui ha avuto origine la richiesta.

sourcePort

Porta da cui ha avuto origine la richiesta.

topicName

Nome dell'argomento sottoscritto.

Esempio di voce di log MQTT 5 Subscribe

```
{
 "timestamp": "2022-11-30 16:24:15.628",
 "logLevel": "INFO",
 "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "Subscribe",
 "protocol": "MQTT",
 "topicName": "test/topic1,$invalid/reserved/topic",
 "subscriptions": [
  {
   "topicName": "test/topic1",
   "reasonCode": 1
  },
  {
   "topicName": "$invalid/reserved/topic",
   "reasonCode": 143
  }
 ],
 "clientId": "abf27092886e49a8a5c1922749736453",
 "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
 "sourceIp": "205.251.233.181",
 "sourcePort": 13490
}
```

Per le operazioni MQTT 5 Subscribe, oltre a attributi Common CloudWatch Logs e Attributi voce di log MQTT 3 Subscribe, le voci di log MQTT 5 `Subscribe` contengono il seguente attributo:

sottoscrizioni

Un elenco di mappature tra gli argomenti richiesti nella richiesta Subscribe e il singolo codice motivo MQTT 5. Per ulteriori informazioni, consulta Codici motivo MQTT.

Annulla l'iscrizione al registro

Il broker di AWS IoT messaggi genera una voce di registro con un `eventType` di `Unsubscribe` quando un client MQTT annulla l'iscrizione a un argomento MQTT.

Esempio di inserimento nel registro di annullamento dell'iscrizione MQTT

```
{
    "timestamp": "2024-08-20 22:53:32.844",
    "logLevel": "INFO",
    "traceId": "db6bd09a-2c3f-1cd2-27cc-fd6b1ce03b58",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Unsubscribe",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `Unsubscribe` registro contengono i seguenti attributi:

protocol

Il protocollo usato per effettuare la richiesta. Il valore sarà sempre `MQTT`.

clientId

ID del client da cui proviene la richiesta.

principalId

ID dell'entità principale da cui proviene la richiesta.

sourceIp

Indirizzo IP da cui ha avuto origine la richiesta.

sourcePort

Porta da cui ha avuto origine la richiesta.

Voci di registro OCSP del certificato del server

AWS IoT Core genera voci di registro per il seguente evento:

Argomenti

- Recupera la voce del registro OCSPStaple dei dati
- Recupera la voce del registro OCSPStaple dei dati per gli endpoint privati

Recupera la voce del registro OCSPStaple dei dati

AWS IoT Core genera una voce di registro con un `eventType` di `RetrieveOCSPStapleData` quando il server recupera i dati di base OCSP.

Recupera esempi OCSPStaple di immissione nel registro dei dati

Di seguito è riportato un esempio di immissione di registro di`Success`.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "INFO",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
 "connectionDetails": {
  "httpStatusCode": "200",
  "ocspResponderUri": "http://ocsp.example.com",
  "sourceIp": "205.251.233.181",
  "targetIp": "250.15.5.3"
 },
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 },
 "ocspResponseDetails": {
```

```
  "responseCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
  "ocspResponseStatus": "successful",
  "certStatus": "good",
  "signature":
 "4C:6F:63:61:6C:20:52:65:73:70:6F:6E:64:65:72:20:53:69:67:6E:61:74:75:72:65",
  "thisUpdateTime": "Jan 31 01:21:02 2024 UTC",
  "nextUpdateTime": "Feb 02 00:21:02 2024 UTC",
  "producedAtTime": "Jan 31 01:37:03 2024 UTC",
  "stapledDataPayloadSize": "XXX"
 }
}
```

Di seguito è riportato un esempio di immissione di registro di`Failure`.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "ERROR",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Failure",
 "reason": "A non 2xx HTTP response was received from the OCSP responder.",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
 "connectionDetails": {
  "httpStatusCode": "444",
  "ocspResponderUri": "http://ocsp.example.com",
  "sourceIp": "205.251.233.181",
  "targetIp": "250.15.5.3"
 },
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 }
}
```

Per l'`RetrieveOCSPStaple`operazione, oltre aattributi Common CloudWatch Logs, le voci di registro contengono i seguenti attributi:

motivo

Il motivo per cui l'operazione non riesce.

domainConfigName

Il nome della configurazione del dominio.

Dettagli della connessione

Una breve spiegazione dei dettagli della connessione.

- httpStatusCode

  Codici di stato HTTP restituiti dal risponditore OCSP in risposta alla richiesta del client effettuata al server.

- ocspResponderUri

  L'URI del risponditore OCSP che viene AWS IoT Core recuperato dal certificato del server.

- sourceIp

  L'indirizzo IP di origine del server. AWS IoT Core

- TargetIP

  L'indirizzo IP di destinazione del risponditore OCSP.

ocspRequestDetails

Dettagli della richiesta OCSP.

- Nome del richiedente

  L'identificatore del AWS IoT Core server che invia una richiesta al risponditore OCSP.

- requestCertId

  L'ID del certificato della richiesta. Questo è l'ID del certificato per il quale viene richiesta la risposta OCSP.

ocspResponseDetails

Dettagli della risposta OCSP.

- responseCertId

  L'ID del certificato della risposta OCSP.

- ocspResponseStatus

  Lo stato della risposta OCSP.

- CertStatus

  Stato del certificato.
- signature

  La firma applicata alla risposta da un'entità attendibile.
- thisUpdateTime

  È noto che l'ora in cui lo stato indicato è corretto.
- nextUpdateTime

  L'ora o prima della quale saranno disponibili informazioni più recenti sullo stato del certificato.
- producedAtTime

  L'ora in cui il risponditore OCSP ha firmato questa risposta.
- stapledDataPayloadDimensioni

  La dimensione del payload dei dati graffati.

Recupera la voce del registro OCSPStaple dei dati per gli endpoint privati

AWS IoT Core genera una voce di registro con un `eventType` di `RetrieveOCSPStapleData` quando il server recupera i dati di base OCSP.

Recupera esempi di immissione nel registro OCSPStaple dei dati per endpoint privati

Di seguito è riportato un esempio di immissione di registro di. `Success`

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "INFO",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
        "lambdaDetails": {
        "lambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
        "sourceArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
testDomainConfigure/6bzfg"
      },
```

```
        "authorizedResponderArn": "arn:aws:acm:us-west-2:123456789012:certificate/
certificate_ID",
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 },
 "ocspResponseDetails": {
  "responderId": "04:C1:3F:8F:27:D6:49:13:F8:DE:B2:36:9D:85:8E:F8:31:3B:A6:D0"
              "responseCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
  "ocspResponseStatus": "successful",
  "certStatus": "good",
  "signature":
 "4C:6F:63:61:6C:20:52:65:73:70:6F:6E:64:65:72:20:53:69:67:6E:61:74:75:72:65",
  "thisUpdateTime": "Jan 31 01:21:02 2024 UTC",
  "nextUpdateTime": "Feb 02 00:21:02 2024 UTC",
  "producedAtTime": "Jan 31 01:37:03 2024 UTC",
  "stapledDataPayloadSize": "XXX"
 }
}
```

Di seguito è riportato un esempio di immissione di registro di`Failure`.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "ERROR",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Failure",
 "reason": "The payload returned by the Lambda function exceeds the maximum response
 size of 7 kilobytes.",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
        "lambdaDetails": {
        "lambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
        "sourceArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
testDomainConfigure/6bzfg"
      },
        "authorizedResponderArn": "arn:aws:acm:us-west-2:123456789012:certificate/
certificate_ID",
  "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
```

```
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 }
}
```

Per l'`RetrieveOCSPStaple`operazione, oltre agli attributi nella voce di registro Retrieve OCSPStaple Data, le voci di registro per gli endpoint privati contengono i seguenti attributi: attributi Common CloudWatch Logs

Dettagli Lambda

Dettagli della funzione Lambda.

- LambdaARN

  ARN della funzione Lambda.

- Fonte ARN

  L'ARN della configurazione di dominio.

authorizedResponderArn

L'ARN del risponditore di autorizzazione, se ce n'è uno configurato nella configurazione del dominio.

Voci del registro Device Shadow

Il servizio AWS IoT Device Shadow genera voci di registro per i seguenti eventi:

Argomenti

- DeleteThingShadow voce di registro
- GetThingShadow voce di registro
- UpdateThingShadow voce di registro

DeleteThingShadow voce di registro

Il servizio Device Shadow genera una voce log con un `eventType` di `DeleteThingShadow` quando viene ricevuta una richiesta di eliminazione di una copia shadow di un dispositivo.

DeleteThingShadow esempio di immissione nel registro

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DeleteThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "Jack",
    "topicName": "$aws/things/Jack/shadow/delete"
}
```

Oltre a[attributi Common CloudWatch Logs](), le voci di `DeleteThingShadow` registro contengono i seguenti attributi:

deviceShadowName

Nome della copia shadow da aggiornare.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

topicName

Nome dell'argomento in cui la richiesta è stata pubblicata.

GetThingShadow voce di registro

Il servizio Device Shadow genera una voce log `eventType` di `GetThingShadow` quando viene ricevuta una richiesta per una copia shadow.

GetThingShadow esempio di immissione nel registro

```
{
    "timestamp": "2017-08-09 17:56:30.941",
    "logLevel": "INFO",
    "traceId": "b575f19a-97a2-cf72-0ed0-c64a783a2504",
    "accountId": "123456789012",
    "status": "Success",
```

```
    "eventType": "GetThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "MyThing",
    "topicName": "$aws/things/MyThing/shadow/get"
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `GetThingShadow` registro contengono i seguenti attributi:

deviceShadowName

Nome della copia shadow richiesta.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

topicName

Nome dell'argomento in cui la richiesta è stata pubblicata.

UpdateThingShadow voce di registro

Il servizio Device Shadow genera una voce log con un `eventType` di `UpdateThingShadow` quando viene ricevuta una richiesta di aggiornamento di una copia shadow di un dispositivo.

UpdateThingShadow esempio di immissione nel registro

```
{
    "timestamp": "2017-08-07 18:43:59.436",
    "logLevel": "INFO",
    "traceId": "d0074ba8-0c4b-a400-69df-76326d414c28",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "UpdateThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "Jack",
    "topicName": "$aws/things/Jack/shadow/update"
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `UpdateThingShadow` registro contengono i seguenti attributi:

deviceShadowName

Nome della copia shadow da aggiornare.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

topicName

Nome dell'argomento in cui la richiesta è stata pubblicata.

Voci di registro del motore delle regole

Il motore AWS IoT delle regole genera i log per i seguenti eventi:

Argomenti

- FunctionExecution voce di registro
- RuleExecution voce di registro
- RuleMatch voce di registro
- RuleExecutionThrottled voce di registro
- RuleNotFound voce di registro
- StartingRuleExecution voce di registro

FunctionExecution voce di registro

Il motore di regole genera una voce log `eventType` di `FunctionExecution` quando una query SQL di una regola chiama una funzione esterna. Una funzione esterna viene chiamata quando l'azione di una regola effettua una richiesta HTTP a AWS IoT o a un altro servizio Web (ad esempio, chiamando `get_thing_shadow` o`machinelearning_predict`).

FunctionExecution esempio di immissione nel registro

```
{
    "timestamp": "2017-07-13 18:33:51.903",
    "logLevel": "DEBUG",
    "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
    "status": "Success",
    "eventType": "FunctionExecution",
    "clientId": "N/A",
```

```
    "topicName":"rules/test",
    "ruleName": "ruleTestPredict",
    "ruleAction": "MachinelearningPredict",
    "resources": {
        "ModelId": "predict-model"
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `FunctionExecution` registro contengono i seguenti attributi:

clientId

N/A per i log `FunctionExecution`.

principalId

ID dell'entità principale da cui proviene la richiesta.

risorse

Raccolta di risorse usate dalle operazioni della regola.

ruleName

Nome della regola corrispondente.

topicName

Nome dell'argomento sottoscritto.

RuleExecution voce di registro

Quando il motore AWS IoT delle regole attiva l'azione di una regola, genera una voce di `RuleExecution` registro.

RuleExecution esempio di immissione di registro

```
{
    "timestamp": "2017-08-10 16:32:46.070",
    "logLevel": "INFO",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
```

```
    "eventType": "RuleExecution",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "ruleAction": "RepublishAction",
    "resources": {
        "RepublishTopic": "rules/republish"
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

RuleExecution con esempio di inserimento di log in batch

```
{
    "logLevel": "INFO",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleExecution",
    "ruleName": "rule_test",
    "ruleAction": "HttpAction",
    "resources": {
        "Url": "https://example.com",
        "ConfirmationUrl": "https://example.com"
    },
    "details": "HttpAction made a request to the specified endpoint",
    "batchDetails": {
        "timestamps": [
            "1234567890123",
            "1234567890123",
            "1234567890123"
        ],
        "traceIds": [
            "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
            "30aa7ccc-1d23-0b97-aa7b-76196d83537c",
            "30aa7ccc-1d23-0b97-aa7b-76196d83537d"
        ],
        "clientIds": [
            "N/A",
            "N/A",
            "N/A"
        ],
        "topicNames": [
            "topic/ruletest",
```

```
            "topic/ruletest",
            "topic/ruletest"
        ],
        "principalIds": [
            "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
            "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
            "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
        ],
        "batchSize": 3,
        "batchSizeInBytes": 114
    }
}
```

Oltre a attributi Common CloudWatch Logs, le voci di registro `RuleExecution` contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

principalId

ID dell'entità principale da cui proviene la richiesta.

risorse

Raccolta di risorse usate dalle operazioni della regola.

ruleAction

Il nome dell'operazione attivata.

ruleName

Nome della regola corrispondente.

topicName

Nome dell'argomento sottoscritto.

RuleMatch voce di registro

Il motore AWS IoT delle regole genera una voce di registro con un `eventType` di `RuleMatch` quando il broker di messaggi riceve un messaggio che corrisponde a una regola.

RuleMatch esempio di immissione di log

```
{
    "timestamp": "2017-08-10 16:32:46.002",
    "logLevel": "INFO",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleMatch",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

Oltre a[attributi Common CloudWatch Logs](), le voci di `RuleMatch` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

principalId

ID dell'entità principale da cui proviene la richiesta.

ruleName

Nome della regola corrispondente.

topicName

Nome dell'argomento sottoscritto.

RuleExecutionThrottled voce di registro

Quando un'esecuzione viene limitata, il motore delle AWS IoT regole genera una voce di registro con un `eventType` of. `RuleExecutionThrottled`

RuleExecutionThrottled esempio di immissione di registro

```
{
    "timestamp": "2017-10-04 19:25:46.070",
    "logLevel": "ERROR",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
```

```
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleExecutionThrottled",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleExecutionThrottled",
    "details": "Exection of Rule example_rule throttled"
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `RuleExecutionThrottled` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

details

Una breve spiegazione dell'errore.

principalId

ID dell'entità principale da cui proviene la richiesta.

motivo

La stringa "RuleExecutionThrottled».

ruleName

Nome della regola da attivare.

topicName

Il nome dell'argomento che è stato pubblicato.

RuleNotFound voce di registro

Quando il motore AWS IoT delle regole non riesce a trovare una regola con un determinato nome, genera una voce di registro con un `eventType` of`RuleNotFound`.

RuleNotFound esempio di immissione di registro

```
{
```

```
    "timestamp": "2017-10-04 19:25:46.070",
    "logLevel": "ERROR",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleNotFound",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleNotFound",
    "details": "Rule example_rule not found"
}
```

Oltre a[attributi Common CloudWatch Logs](), le voci di `RuleNotFound` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

details

Una breve spiegazione dell'errore.

principalId

ID dell'entità principale da cui proviene la richiesta.

motivo

La stringa "RuleNotFound».

ruleName

Il nome della regola che non è stato possibile trovare.

topicName

Il nome dell'argomento che è stato pubblicato.

StartingRuleExecution voce di registro

Quando il motore AWS IoT delle regole inizia ad attivare l'azione di una regola, genera una voce di registro con un `eventType` of`StartingRuleExecution`.

StartingRuleExecution esempio di immissione di registro

```
{
    "timestamp": "2017-08-10 16:32:46.002",
    "logLevel": "DEBUG",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "StartingRuleExecution",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "ruleAction": "RepublishAction",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `rule-` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

principalId

ID dell'entità principale da cui proviene la richiesta.

ruleAction

Il nome dell'operazione attivata.

ruleName

Nome della regola corrispondente.

topicName

Nome dell'argomento sottoscritto.

Voci del registro processi

Il servizio AWS IoT Job genera voci di registro per i seguenti eventi. I log vengono generati quando viene ricevuta una richiesta MQTT o HTTP dal dispositivo.

Argomenti

- DescribeJobExecution voce di registro

- GetPendingJobExecution voce di registro
- ReportFinalJobExecutionCount voce di registro
- StartNextPendingJobExecution voce di registro
- UpdateJobExecution voce di registro

DescribeJobExecution voce di registro

Il servizio AWS IoT Jobs genera una voce di registro con un `eventType` di `DescribeJobExecution` quando il servizio riceve una richiesta per descrivere l'esecuzione di un processo.

DescribeJobExecution esempio di immissione di registro

```
{
    "timestamp": "2017-08-10 19:13:22.841",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DescribeJobExecution",
    "protocol": "MQTT",
    "clientId": "thingOne",
    "jobId": "002",
    "topicName": "$aws/things/thingOne/jobs/002/get",
    "clientToken": "myToken",
    "details": "The request status is SUCCESS."
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `GetJobExecution` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

clientToken

Identificatore univoco con distinzione tra maiuscole e minuscole per assicurare l'idempotenza della richiesta. Per ulteriori informazioni, consulta Come assicurare l'idempotenza.

details

Ulteriori informazioni dal servizio Jobs.

jobId

Job ID per l'esecuzione del processo.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

topicName

Argomento usato per effettuare la richiesta.

GetPendingJobExecution voce di registro

Il servizio AWS IoT Jobs genera una voce di registro con un `eventType` di `GetPendingJobExecution` quando il servizio riceve una richiesta di esecuzione del lavoro.

GetPendingJobExecution esempio di immissione di registro

```
{
    "timestamp": "2018-06-13 17:45:17.197",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetPendingJobExecution",
    "protocol": "MQTT",
    "clientId": "299966ad-54de-40b4-99d3-4fc8b52da0c5",
    "topicName": "$aws/things/299966ad-54de-40b4-99d3-4fc8b52da0c5/jobs/get",
    "clientToken": "24b9a741-15a7-44fc-bd3c-1ff2e34e5e82",
    "details": "The request status is SUCCESS."
}
```

Oltre a[attributi Common CloudWatch Logs](), le voci di `GetPendingJobExecution` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

clientToken

Identificatore univoco con distinzione tra maiuscole e minuscole per assicurare l'idempotenza della richiesta. Per ulteriori informazioni, consulta Come assicurare l'idempotenza.

details

Ulteriori informazioni dal servizio Jobs.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

topicName

Nome dell'argomento sottoscritto.

ReportFinalJobExecutionCount voce di registro

Il servizio AWS IoT Jobs genera una voce di registro con `entryType` `ReportFinalJobExecutionCount` l'indicazione del completamento di un lavoro.

ReportFinalJobExecutionCount esempio di immissione di registro

```
{
    "timestamp": "2017-08-10 19:44:16.776",
    "logLevel": "INFO",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "ReportFinalJobExecutionCount",
    "jobId": "002",
    "details": "Job 002 completed. QUEUED job execution count: 0 IN_PROGRESS job
 execution count: 0 FAILED job execution count: 0 SUCCEEDED job execution count: 1
 CANCELED job execution count: 0 REJECTED job execution count: 0 REMOVED job execution
 count: 0"
}
```

Oltre a[attributi Common CloudWatch Logs](), le voci di `ReportFinalJobExecutionCount` registro contengono i seguenti attributi:

details

Ulteriori informazioni dal servizio Jobs.

jobId

Job ID per l'esecuzione del processo.

StartNextPendingJobExecution voce di registro

Quando riceve una richiesta per avviare la successiva esecuzione del processo in sospeso, il servizio AWS IoT Jobs genera una voce di registro con un `eventType` of`StartNextPendingJobExecution`.

StartNextPendingJobExecution esempio di immissione di registro

```
{
    "timestamp": "2018-06-13 17:49:51.036",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "StartNextPendingJobExecution",
    "protocol": "MQTT",
    "clientId": "95c47808-b1ca-4794-bc68-a588d6d9216c",
    "topicName": "$aws/things/95c47808-b1ca-4794-bc68-a588d6d9216c/jobs/start-next",
    "clientToken": "bd7447c4-3a05-49f4-8517-dd89b2c68d94",
    "details": "The request status is SUCCESS."
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `StartNextPendingJobExecution` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

clientToken

Identificatore univoco con distinzione tra maiuscole e minuscole per assicurare l'idempotenza della richiesta. Per ulteriori informazioni, consulta Come assicurare l'idempotenza.

details

Ulteriori informazioni dal servizio Jobs.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

topicName

Argomento usato per effettuare la richiesta.

UpdateJobExecution voce di registro

Il servizio AWS IoT Jobs genera una voce di registro con un `eventType` di `UpdateJobExecution` quando il servizio riceve una richiesta di aggiornamento dell'esecuzione di un lavoro.

UpdateJobExecution esempio di immissione di registro

```
{
    "timestamp": "2017-08-10 19:25:14.758",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "UpdateJobExecution",
    "protocol": "MQTT",
    "clientId": "thingOne",
    "jobId": "002",
    "topicName": "$aws/things/thingOne/jobs/002/update",
    "clientToken": "myClientToken",
    "versionNumber": "1",
    "details": "The destination status is IN_PROGRESS. The request status is SUCCESS."
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `UpdateJobExecution` registro contengono i seguenti attributi:

clientId

ID del client da cui proviene la richiesta.

clientToken

Identificatore univoco con distinzione tra maiuscole e minuscole per assicurare l'idempotenza della richiesta. Per ulteriori informazioni, consulta Come assicurare l'idempotenza.

details

Ulteriori informazioni dal servizio Jobs.

jobId

Job ID per l'esecuzione del processo.

protocol

Il protocollo usato per effettuare la richiesta. I valori validi sono `MQTT` e `HTTP`.

topicName

Argomento usato per effettuare la richiesta.

versionNumber

Versione dell'esecuzione del processo.

Voci del registro di provisioning dei dispositivi

Il servizio AWS IoT Device Provisioning genera registri per i seguenti eventi.

Argomenti

- GetDeviceCredentials voce di registro
- ProvisionDevice voce di registro

GetDeviceCredentials voce di registro

Il servizio AWS IoT Device Provisioning genera una voce di registro con l'indicazione `eventType` di `GetDeviceCredential` quando un client chiama`GetDeviceCredential`.

GetDeviceCredentials esempio di immissione di registro

```
{
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "GetDeviceCredentials",
  "deviceCertificateId" :
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `GetDeviceCredentials` registro contengono i seguenti attributi:

details

Una breve spiegazione dell'errore.

deviceCertificateId

L'ID del certificato del dispositivo.

ProvisionDevice voce di registro

Il servizio AWS IoT Device Provisioning genera una voce di registro con l'indicazione `eventType` di `ProvisionDevice` quando un client chiama`ProvisionDevice`.

ProvisionDevice esempio di immissione di registro

```
{
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "ProvisionDevice",
  "provisioningTemplateName" : "myTemplate",
  "deviceCertificateId" :
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
 }
```

Oltre a[attributi Common CloudWatch Logs](), le voci di `ProvisionDevice` registro contengono i seguenti attributi:

details

Una breve spiegazione dell'errore.

deviceCertificateId

L'ID del certificato del dispositivo.

provisioningTemplateName

Il nome del modello dell'artefatto di provisioning.

Voci del registro del gruppo di oggetti dinamici

AWS IoT I Dynamic Thing Groups generano log per il seguente evento.

Argomenti

- AddThingToDynamicThingGroupsFailed voce di registro

AddThingToDynamicThingGroupsFailed voce di registro

Quando non AWS IoT è stato possibile aggiungere un elemento ai gruppi dinamici specificati, genera una voce di registro con un `eventType` of`AddThingToDynamicThingGroupsFailed`. Questo accade se un oggetto soddisfaceva i criteri per essere inserito nel gruppo di oggetti dinamico; tuttavia, non è stato possibile aggiungerlo al gruppo dinamico o è stato rimosso dal gruppo dinamico. Questo può accadere perché:

- L'oggetto appartiene già al numero massimo di gruppi.
- L'opzione --override-dynamic-groups è stata utilizzata per aggiungere l'oggetto a un gruppo di oggetti statico. È stato rimosso da un gruppo di oggetti dinamico per rendere ciò possibile.

Per ulteriori informazioni, consulta Limitazioni e conflitti del gruppo di oggetti dinamico.

AddThingToDynamicThingGroupsFailed esempio di immissione di registro

Questo esempio mostra la voce di log di un errore `AddThingToDynamicThingGroupsFailed`. In questo esempio, TestThingsoddisfaceva i criteri per rientrare nei gruppi di oggetti dinamici elencati in`dynamicThingGroupNames`, ma non poteva essere aggiunto a tali gruppi dinamici, come descritto in`reason`.

```
{
 "timestamp": "2020-03-16 22:24:43.804",
 "logLevel": "ERROR",
 "traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",
 "accountId": "57EXAMPLE833",
 "status": "Failure",
 "eventType": "AddThingToDynamicThingGroupsFailed",
 "thingName": "TestThing",
 "dynamicThingGroupNames": [
  "DynamicThingGroup11",
  "DynamicThingGroup12",
  "DynamicThingGroup13",
  "DynamicThingGroup14"
 ],
 "reason": "The thing failed to be added to the given dynamic thing group(s) because
 the thing already belongs to the maximum allowed number of groups."
```

```
}
```

Oltre aattributi Common CloudWatch Logs, le voci di `AddThingToDynamicThingGroupsFailed` registro contengono i seguenti attributi:

dynamicThingGroupNomi

Un array dei gruppi di oggetti dinamici a cui non è stato possibile aggiungere l'oggetto.

motivo

Il motivo per il quale non è stato possibile aggiungere l'oggetto ai gruppi di oggetti dinamici.

thingName

Il nome dell'oggetto che non è stato possibile aggiungere a un gruppo di oggetti dinamico.

Voci di registro di indicizzazione del parco istanze

AWS IoT l'indicizzazione della flotta genera voci di registro per i seguenti eventi.

Argomenti

- NamedShadowCountForDynamicGroupQueryLimitExceeded voce di registro

NamedShadowCountForDynamicGroupQueryLimitExceeded voce di registro

Viene elaborato un massimo di 25 shadow denominate per oggetto per termini di query che non sono specifici dell'origine dei dati nei gruppi dinamici. Quando questo limite viene violato, il tipo di evento `NamedShadowCountForDynamicGroupQueryLimitExceeded` verrà emesso.

NamedShadowCountForDynamicGroupQueryLimitExceeded esempio di immissione nel registro

Questo esempio mostra la voce di registro per un errore `NamedShadowCountForDynamicGroupQueryLimitExceeded`. In questo esempio i risultati di `DynamicGroup`, basati su tuti i valori, possono essere imprecisi come descritto nel campo `reason`.

```
{
"timestamp": "2020-03-16 22:24:43.804",
"logLevel": "ERROR",
"traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",
"accountId": "571032923833",
```

```
"status": "Failure",
"eventType": "NamedShadowCountForDynamicGroupQueryLimitExceeded",
"thingName": "TestThing",
"reason": "A maximum of 25 named shadows per thing are processed for non-data source
 specific query terms in dynamic groups."
}
```

attributi Common CloudWatch Logs

Tutte le voci di registro CloudWatch di Logs includono i seguenti attributi:

accountId

Il tuo Account AWS ID.

eventType

Tipo di evento per cui il log è stato generato. Il valore del tipo di evento dipende dall'evento che ha generato la voce di registro. Ogni descrizione della voce di registro include il valore di `eventType` per tale voce di registro.

logLevel

Livello di log usato. Per ulteriori informazioni, consulta the section called “Livelli di log”.

status

Stato della richiesta.

timestamp

Timestamp UTC leggibile relativo al momento in cui il client si è connesso al broker di messaggi AWS IoT .

traceId

Identificatore generato in modo casuale che può essere usato per correlare tutti i log per una richiesta specifica.

Carica i log lato dispositivo su Amazon CloudWatch

Puoi caricare log storici lato dispositivo su CloudWatch Amazon per monitorare e analizzare l'attività di un dispositivo sul campo. I log lato dispositivo possono includere file di log del sistema,

dell'applicazione e del dispositivo. Questo processo utilizza un parametro di azione CloudWatch Logs rules per pubblicare i log lato dispositivo in un gruppo di log definito dal cliente.

Come funziona

Il processo inizia quando un AWS IoT dispositivo invia messaggi MQTT contenenti file di registro formattati a un argomento. AWS IoT Una AWS IoT regola monitora l'argomento del messaggio e invia i file di registro a un gruppo di CloudWatch registri definito dall'utente. È quindi possibile rivedere e analizzare le informazioni.

Argomenti

- Argomenti MQTT
- Operazione delle regole

Argomenti MQTT

Scegli uno spazio dei nomi dell'argomento MQTT che utilizzerai per pubblicare i log. Ti consigliamo di utilizzare questo formato per lo spazio degli argomenti comune, `$aws/rules/things/thing_name/logs`, e questo formato per argomenti degli errori, `$aws/rules/things/thing_name/logs/errors`. La struttura di denominazione per i log e gli argomenti degli errori è consigliata, ma non obbligatoria. For ulteriori informazioni, consulta Progettazione di argomenti MQTT per AWS IoT Core.

Utilizzando lo spazio tematico comune consigliato, si utilizzano argomenti riservati AWS IoT di Basic Ingest. AWS IoT Basic Ingest invia in modo sicuro i dati del dispositivo ai AWS servizi supportati dalle azioni delle regole. AWS IoT Rimuove il broker di publish/subscribe messaggi dal percorso di inserimento, rendendolo più conveniente. Per ulteriori informazioni, consulta Riduzione dei costi di messaggistica con Basic Ingest.

Se si utilizza batchMode per caricare file di log, i messaggi devono seguire un formato specifico che include un timestamp UNIX e un messaggio. Per ulteriori informazioni, consultate l'argomento Requisiti del formato dei messaggi MQTT per BatchMode all'interno dell'azione della regola Logs. CloudWatch

Operazione delle regole

Quando AWS IoT riceve i messaggi MQTT dai dispositivi client, una AWS IoT regola monitora l'argomento definito dal cliente e pubblica i contenuti in un gruppo di log definito dall'utente.

CloudWatch Questo processo utilizza un'azione della regola CloudWatch Logs per monitorare MQTT per batch di file di registro. Per ulteriori informazioni, vedere l'azione della regola CloudWatch AWS IoT Logs.

Modalità batch

`batchMode`è un parametro booleano all'interno dell'azione della regola AWS IoT CloudWatch Logs. Questo parametro è facoltativo ed è disattivato (`false`) per impostazione predefinita. Per caricare i file di registro lato dispositivo in batch, è necessario attivare questo parametro () `true` quando si crea la regola. AWS IoT Per ulteriori informazioni, consulta CloudWatch Log nella sezione Azioni delle regole.AWS IoT

Caricamento dei log lato dispositivo utilizzando le regole AWS IoT

Puoi utilizzare il motore AWS IoT delle regole per caricare i record di registro dai file di registro esistenti sul dispositivo (log di sistema, applicazione e dispositivo-client) su Amazon. CloudWatch Quando i log lato dispositivo vengono pubblicati su un argomento MQTT, l'azione Logs rules trasferisce i messaggi in Logs. CloudWatch CloudWatch Questo processo descrive come caricare i log dei dispositivi in batch utilizzando il parametro `batchMode` di azione delle regole attivato (impostato su `true`).

Per iniziare a caricare i log lato dispositivo su, completa i seguenti prerequisiti. CloudWatch

Prerequisiti

Prima di iniziare, esegui queste attività:

- Crea almeno un dispositivo IoT di destinazione registrato AWS IoT Core come AWS IoT oggetto. Per ulteriori informazioni, consulta Crea un oggetto.
- Determina lo spazio degli argomenti MQTT per l'inserimento e gli errori. Per ulteriori informazioni sugli argomenti MQTT e sulle convenzioni di denominazione consigliate, consulta la sezione Argomenti Argomenti MQTT MQTT in Upload device-side logs to Amazon. CloudWatch

Per ulteriori informazioni su questi prerequisiti, consulta Caricare i log lato dispositivo su. CloudWatch

CloudWatch Creazione di un gruppo di log

Per creare un gruppo di CloudWatch log, completare i seguenti passaggi. Scegliete la scheda appropriata a seconda che preferiate eseguire i passaggi tramite AWS Command Line Interface (AWS CLI). Console di gestione AWS

Console di gestione AWS

Per creare un gruppo di CloudWatch log utilizzando Console di gestione AWS

1. Aprire Console di gestione AWS e accedere a CloudWatch.
2. Nel pannello di navigazione, scegli Logs (Log), quindi Log groups (Gruppi di log).
3. Scegliere Crea gruppo di log.
4. Aggiorna il campo Log group name (Nome del gruppo di log) e, facoltativamente, aggiorna i campi Retention setting (Impostazione conservazione).
5. Scegli Create (Crea).

AWS CLI

Per creare un gruppo di CloudWatch log utilizzando il AWS CLI

1. Per creare il gruppo di log, esegui il seguente comando. Per ulteriori informazioni, vedere `create-log-group` nella Guida ai comandi AWS CLI v2.

   Sostituisci il nome del gruppo di log nell'esempio (`uploadLogsGroup`) con il tuo nome preferito.

   ```
   aws logs create-log-group --log-group-name uploadLogsGroup
   ```

2. Per confermare che il gruppo di log è stato creato correttamente, esegui il seguente comando.

   ```
   aws logs describe-log-groups --log-group-name-prefix uploadLogsGroup
   ```

   Output di esempio:

   ```
   {
       "logGroups": [
           {
               "logGroupName": "uploadLogsGroup",
               "creationTime": 1674521804657,
               "metricFilterCount": 0,
               "arn": "arn:aws:logs:us-east-1:111122223333:log-
   group:uploadLogsGroup:*",
               "storedBytes": 0
   ```

```
        }
    ]
}
```

Creazione di una regola dell'argomento

Per creare una AWS IoT regola, completa i passaggi seguenti. Scegliete la scheda appropriata a seconda che preferiate eseguire i passaggi tramite AWS Command Line Interface (AWS CLI). Console di gestione AWS

Console di gestione AWS

Per creare una regola tematica utilizzando Console di gestione AWS

1. Apri l'hub delle regole.
   a. Apri la Console di gestione AWS e accedi a AWS IoT.
   b. Nella barra di navigazione, scegli Message routing (Instradamento dei messaggi) e quindi Rules (Regole).
   c. Scegli Crea regola.
2. Inserisci le proprietà della regola.
   a. Inserisci un nome della regola alfanumerico.
   b. (Facoltativo) Compila i campi Rule description (Descrizione regola) e Tags (Tag).
   c. Scegli Next (Successivo).
3. Inserisci un'istruzione SQL.
   a. Inserisci un'istruzione SQL utilizzando l'argomento MQTT che hai definito per l'acquisizione.

      Ad esempio, `SELECT * FROM '$aws/rules/things/thing_name/logs'`
   b. Scegli Next (Successivo).
4. Inserisci le azioni delle regole.
   a. Nel menu Azione 1, scegli CloudWatchregistri.
   b. Scegli l'opzione Log group name (Nome del gruppo di log) quindi seleziona il gruppo di log che hai creato.

c. Seleziona Use batch mode (Utilizza la modalità batch).

d. Specifica il ruolo IAM per la regola.

Se hai già un ruolo IAM per la regola, procedi nel seguente modo.

1. Nel menu IAM role (Ruolo IAM), scegli il tuo ruolo IAM.

Se non hai già un ruolo IAM per la regola, procedi nel seguente modo.

1. Scegli Create new role (Crea nuovo ruolo).
2. In Role name (Nome ruolo), inserisci un nome univoco e scegli Create (Crea).
3. Conferma che il nome del ruolo IAM sia corretto nel campo IAM role (Ruolo IAM).

e. Scegli Next (Successivo).

5. Rivedi la configurazione del modello.

a. Rivedi le impostazioni del modello di processo per verificare che siano corrette.

b. Al termine, selezionare Create (Crea).

AWS CLI

Per creare un ruolo IAM e una regola tematica utilizzando il AWS CLI

1. Crea un ruolo IAM che conceda i diritti sulla AWS IoT regola.

a. Creare una policy IAM

Per creare una policy IAM, esegui il comando seguente. Assicurati di aggiornare il valore del parametro `policy-name`. Per ulteriori informazioni, consulta la Guida `create-policy`ai comandi della AWS CLI versione v2.

> **Note**
>
> Se stai utilizzando un sistema operativo Microsoft Windows, potrebbe essere necessario sostituire l'indicatore di fine riga (\) con un segno di spunta (`) o un altro carattere.

```
aws iam create-policy \
    --policy-name uploadLogsPolicy \
    --policy-document \
'{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iot:CreateTopicRule",
            "iot:Publish",
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:GetLogEvents"
        ],
        "Resource": "*"
    }
}'
```

b. Copia l'ARN della policy dal tuo output in un editor di testo.

   Output di esempio:

```
{
    "Policy": {
        "PolicyName": "uploadLogsPolicy",
        "PermissionsBoundaryUsageCount": 0,
        "CreateDate": "2023-01-23T18:30:10Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "AAABBBCCCDDDEEEFFFGGG",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::111122223333:policy/uploadLogsPolicy",
        "UpdateDate": "2023-01-23T18:30:10Z"
    }
}
```

c. Crea un ruolo IAM e una policy di attendibilità.

Per creare una policy IAM, esegui il comando seguente. Assicurati di aggiornare il valore del parametro `role-name`. Per ulteriori informazioni, vedere `create-role`nella Guida ai comandi AWS CLI v2.

```
aws iam create-role \
--role-name uploadLogsRole \
--assume-role-policy-document \
'{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'
```

d. Collega la policy IAM al ruolo.

Per creare una policy IAM, esegui il comando seguente. Assicurati di aggiornare i valori dei parametri `role-name` e `policy-arn`. Per ulteriori informazioni, vedere `attach-role-policy`nella Guida ai comandi AWS CLI v2.

```
aws iam attach-role-policy \
--role-name uploadLogsRole \
--policy-arn arn:aws:iam::111122223333:policy/uploadLogsPolicy
```

e. Rivedi il ruolo.

Per confermare che ruolo IAM sia stato creato correttamente, esegui il seguente comando. Assicurati di aggiornare il valore del parametro `role-name`. Per ulteriori informazioni, vedere `get-role`nella Guida ai comandi AWS CLI v2.

```
aws iam get-role --role-name uploadLogsRole
```

Output di esempio:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "uploadLogsRole",
        "RoleId": "AAABBBCCCDDDEEEFFFGGG",
        "Arn": "arn:aws:iam::111122223333:role/uploadLogsRole",
        "CreateDate": "2023-01-23T19:17:15+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "iot.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Description": "",
        "MaxSessionDuration": 3600,
        "RoleLastUsed": {}
    }
}
```

2. Crea una regola AWS IoT tematica in. AWS CLI

   a. Per creare una regola AWS IoT tematica, esegui il comando seguente. Assicurati di aggiornare i valori dell'`sql`istruzione `--rule-name` `descriptionroleARN` , e `logGroupName` dei parametri. Per ulteriori informazioni, consultate la [create-topic-rule](#)Guida ai comandi della AWS CLI versione v2.

```
aws iot create-topic-rule \
--rule-name uploadLogsRule \
--topic-rule-payload \
 '{
  "sql":"SELECT * FROM 'rules/things/thing_name/logs'",
  "description":"Upload logs test rule",
  "ruleDisabled":false,
  "awsIotSqlVersion":"2016-03-23",
  "actions":[
```

```
   {"cloudwatchLogs":
    {"roleArn":"arn:aws:iam::111122223333:role/uploadLogsRole",
    "logGroupName":"uploadLogsGroup",
    "batchMode":true}
    }
   ]
}'
```

b. Per confermare che la regola è stata creata correttamente, esegui il seguente comando. Assicurati di aggiornare il valore del parametro `role-name`. Per ulteriori informazioni, vedere get-topic-rulenella Guida ai comandi AWS CLI v2.

```
aws iot get-topic-rule --rule-name uploadLogsRule
```

Output di esempio:

```
{
    "ruleArn": "arn:aws:iot:us-east-1:111122223333:rule/uploadLogsRule",
    "rule": {
        "ruleName": "uploadLogsRule",
        "sql": "SELECT * FROM rules/things/thing_name/logs",
        "description": "Upload logs test rule",
        "createdAt": "2023-01-24T16:28:15+00:00",
        "actions": [
            {
                "cloudwatchLogs": {
                    "roleArn": "arn:aws:iam::111122223333:role/
uploadLogsRole",
                    "logGroupName": "uploadLogsGroup",
                    "batchMode": true
                }
            }
        ],
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23"
    }
}
```

Invio di log lato dispositivo a AWS IoT

Per inviare registri lato dispositivo a AWS IoT

1. A cui inviare i log cronologici AWS IoT, comunica con i tuoi dispositivi per garantire quanto segue.

   - Le informazioni di log vengono inviate allo spazio dei nomi degli argomenti corretto come specificato nella sezione Prerequisiti di questa procedura.

     Ad esempio, `$aws/rules/things/thing_name/logs`

   - Il payload del messaggio MQTT è formattato correttamente. Per ulteriori informazioni sull'argomento MQTT e sulla convenzione di denominazione consigliata, consultate la sezione all'Argomenti MQTTinterno. Carica i log lato dispositivo su Amazon CloudWatch

2. Verificare che i messaggi MQTT vengano ricevuti all'interno del AWS IoT client MQTT.

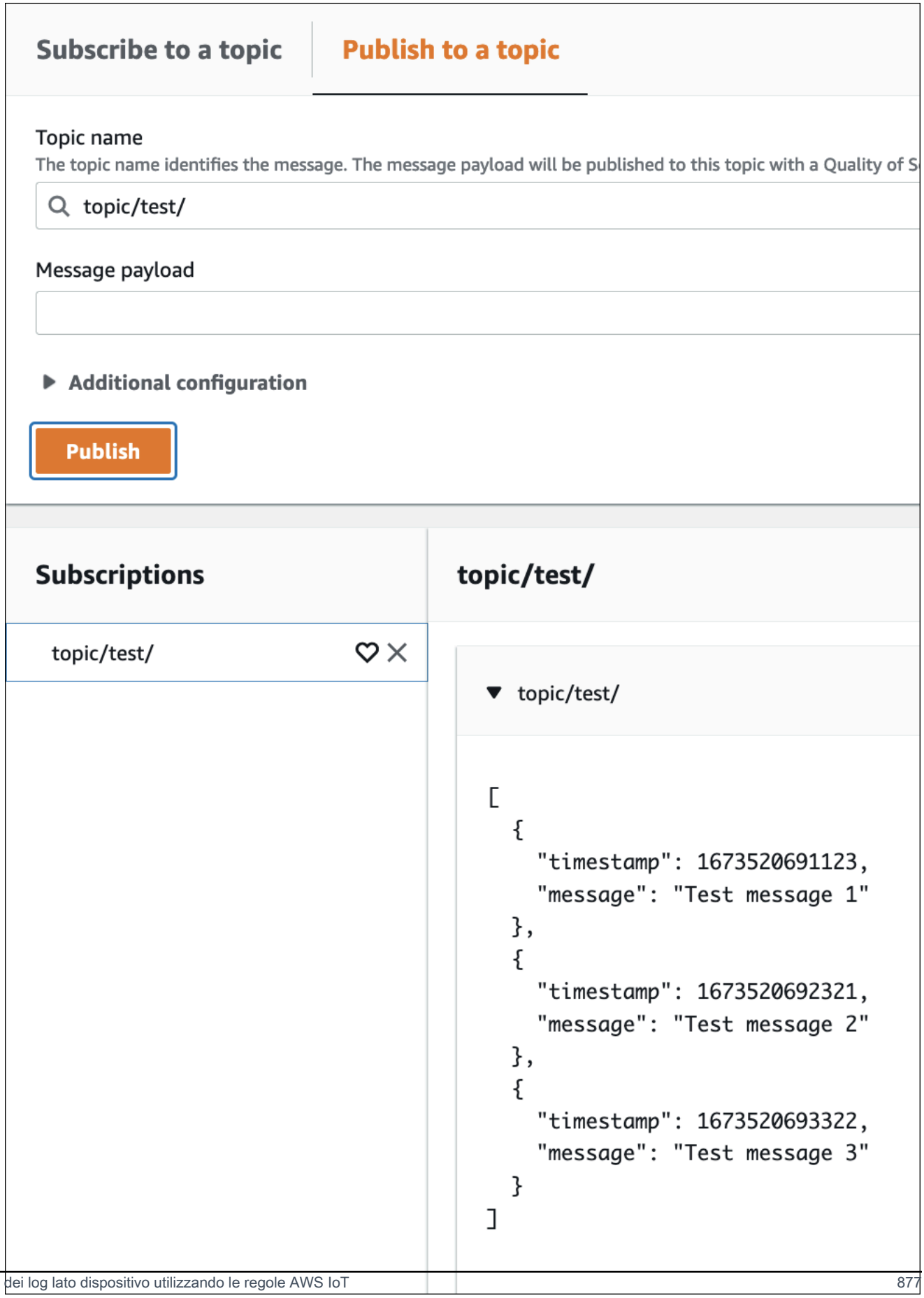
   a. Apri Console di gestione AWS e vai a. AWS IoT

   b. Per visualizzare il client di test MQTT, nella barra di navigazione, scegliete Test, MQTT test client.

   c. Per Sottoscrivi a un argomento, Filtro argomento, inserite lo spazio dei nomi dell'argomento.

   d. Scegli Abbonati.

      I messaggi MQTT vengono visualizzati nella tabella Subscriptions (Sottoscrizioni) e Topic (Argomento), come illustrato di seguito. La visualizzazione di questi messaggi può richiedere fino a cinque minuti.


Subscribe to a topic
Publish to a topic
Topic name
The topic name identifies the message. The message payload will be published to this topic with a Quality of S
topic/test/
Message payload
Additional configuration
Publish
Subscriptions
topic/test/
topic/test/
topic/test/
[
{
"timestamp": 1673520691123,
"message": "Test message 1"
},
{
"timestamp": 1673520692321,
"message": "Test message 2"
},
{
"timestamp": 1673520693322,
"message": "Test message 3"
}
]

Visualizzazione dei dati di log

Per esaminare i record di registro in Logs CloudWatch

1. Apri e Console di gestione AWS accedi a. CloudWatch
2. Nella barra di navigazione, scegli Logs, Logs Insights.
3. Nel menu Seleziona gruppi di log, scegli il gruppo di log specificato nella AWS IoT regola.
4. Nella pagina Logs insights (Informazioni dettagliate sui log), scegli Run query (Esegui query).

Registrazione delle chiamate AWS IoT API utilizzando AWS CloudTrail

AWS IoT è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o AWS servizio in AWS IoT. CloudTrail acquisisce tutte le chiamate API relative AWS IoT agli eventi, incluse le chiamate dalla AWS IoT console e le chiamate in codice a. AWS IoT APIs Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per. AWS IoT Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare a quale richiesta è stata inviata AWS IoT, l'indirizzo IP da cui è stata effettuata, chi ha effettuato la richiesta, quando è stata effettuata e altri dettagli.

Per ulteriori informazioni CloudTrail, consulta la Guida AWS CloudTrail per l'utente.

AWS IoT informazioni in CloudTrail

CloudTrail è abilitato sul tuo account al Account AWS momento della creazione dell'account. Quando si verifica un'attività in AWS IoT, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi AWS di servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare eventi recenti in Account AWS. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nell' Account AWS che includa gli eventi per AWS IoT, crea un trail. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando crei un percorso nella console, il percorso si applica a tutti i Regione AWS file. Il trail registra gli eventi di tutti Regione AWS i file della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Puoi configurare altri AWS servizi per

analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta:

- Panoramica della creazione di un trail
- CloudTrail Servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

> **Note**
>
> AWS IoT le azioni del piano dati (lato dispositivo) non vengono registrate da CloudTrail. Utilizzare CloudWatch per monitorare queste azioni.

In generale, le azioni AWS IoT del piano di controllo che apportano modifiche vengono registrate da CloudTrail. Le chiamate come CreateThinge UpdateCertificatelasciano CloudTrail immissioni, mentre le chiamate come ListThingse non ListTopicRuleslo fanno. CreateKeysAndCertificate

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali dell'utente IAM o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

AWS IoT le azioni sono documentate nell'AWS IoT API Reference. Wireless AWS IoT le azioni sono documentate nell'Wireless AWS IoT API Reference.

Comprendere le AWS IoT voci dei file di registro

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro.

Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'`AttachPolicy`azione.


```
{
    "timestamp":"1460159496",
    "AdditionalEventData":"",
    "Annotation":"",
    "ApiVersion":"",
    "ErrorCode":"",
    "ErrorMessage":"",
    "EventID":"8bff4fed-c229-4d2d-8264-4ab28a487505",
    "EventName":"AttachPolicy",
    "EventTime":"2016-04-08T23:51:36Z",
    "EventType":"AwsApiCall",
    "ReadOnly":"",
    "RecipientAccountList":"",
    "RequestID":"d4875df2-fde4-11e5-b829-23bf9b56cbcd",
    "RequestParamters":{
        "principal":"arn:aws:iot:us-
east-1:123456789012:cert/528ce36e8047f6a75ee51ab7beddb4eb268ad41d2ea881a10b67e8e76924d894",
        "policyName":"ExamplePolicyForIoT"
    },
    "Resources":"",
    "ResponseElements":"",
    "SourceIpAddress":"52.90.213.26",
    "UserAgent":"aws-internal/3",
    "UserIdentity":{
        "type":"AssumedRole",
        "principalId":"AKIAI44QH8DHBEXAMPLE",
        "arn":"arn:aws:sts::12345678912:assumed-role/iotmonitor-us-east-1-beta-
InstanceRole-1C5T1YCYMHPYT/i-35d0a4b6",
        "accountId":"222222222222",
        "accessKeyId":"access-key-id",
        "sessionContext":{
            "attributes":{
                "mfaAuthenticated":"false",
                "creationDate":"Fri Apr 08 23:51:10 UTC 2016"
            },
```

```
            "sessionIssuer":{
                "type":"Role",
                "principalId":"AKIAI44QH8DHBEXAMPLE",
                "arn":"arn:aws:iam::123456789012:role/executionServiceEC2Role/
iotmonitor-us-east-1-beta-InstanceRole-1C5T1YCYMHPYT",
                "accountId":"222222222222",
                "userName":"iotmonitor-us-east-1-InstanceRole-1C5T1YCYMHPYT"
            }
        },
        "invokedBy":{
            "serviceAccountId":"111111111111"
        }
    },
    "VpcEndpointId":""
}
```

Regole per AWS IoT

Le regole offrono ai tuoi dispositivi la possibilità di interagire con Servizi AWS. Le regole vengono analizzate e vengono eseguite operazioni in base al flusso di argomenti MQTT. È possibile usare le regole a supporto di attività come le seguenti:

- Aumentare o filtrare i dati ricevuti da un dispositivo.
- Scrivere i dati ricevuti da un dispositivo in un database Amazon DynamoDB.
- Salvare un file in Amazon S3.
- Inviare una notifica push a tutti gli utenti che utilizzano Amazon SNS.
- Pubblicare i dati in una coda Amazon SQS.
- Richiamare una funzione Lambda per estrarre i dati.
- Elaborare i messaggi da un numero elevato di dispositivi con Amazon Kinesis.
- Invia dati ad Amazon OpenSearch Service.
- Acquisisci una CloudWatch metrica.
- Cambia una CloudWatch sveglia.
- Invia i dati da un messaggio MQTT ad Amazon SageMaker AI per fare previsioni basate su un modello di machine learning (ML).
- Inviare un messaggio a un flusso di input Salesforce IoT.
- Avviare il processo di una macchina a stati Step Functions.
- Invia i dati del messaggio a un AWS IoT Events input.
- Inviare i dati del messaggio a una proprietà di asset in AWS IoT SiteWise.
- Inviare i dati dei messaggi a un'applicazione Web o a un servizio.

Le tue regole possono utilizzare messaggi MQTT che passano attraverso il publish/subscribe protocollo supportato da. the section called “Protocolli di dispositivo di comunicazione” È inoltre possibile utilizzare la funzione Basic Ingest per inviare in modo sicuro i dati del dispositivo a Servizi AWS quelli elencati in precedenza, senza incorrere in costi di messaggistica. La funzione Basic Ingest ottimizza il flusso di dati rimuovendo il broker di publish/subscribe messaggi dal percorso di ingestione. Ciò la rende conveniente pur mantenendo le funzionalità di sicurezza ed elaborazione dei dati di. AWS IoT

Prima di AWS IoT poter eseguire queste azioni, devi concedergli l'autorizzazione ad accedere alle tue AWS risorse per tuo conto. Quando le azioni vengono eseguite, ti vengono addebitati i costi standard per le azioni Servizi AWS che utilizzi.

Indice

- Concedere a qualsiasi AWS IoT regola l'accesso richiesto
- Passaggio delle autorizzazioni ai ruoli
- Creare una AWS IoT regola
- Gestione di una regola AWS IoT
- AWS IoT azioni relative alle regole
- Risoluzione dei problemi relativi a una regola
- Accesso alle risorse di più account utilizzando le regole AWS IoT
- Gestione degli errori (operazione in caso di errore)
- Riduzione dei costi di messaggistica con Basic Ingest
- AWS IoT Riferimento SQL

Concedere a qualsiasi AWS IoT regola l'accesso richiesto

Usa i ruoli IAM per controllare le AWS risorse a cui ogni regola ha accesso. Prima di creare una regola, devi creare un ruolo IAM con una policy che consenta l'accesso alle AWS risorse richieste. AWS IoT assume questo ruolo quando implementa una regola.

Completa i seguenti passaggi per creare il ruolo e la AWS IoT policy IAM che garantiscano a una AWS IoT regola l'accesso richiesto (AWS CLI).

1. Salva il seguente documento sulla politica di fiducia, che concede AWS IoT l'autorizzazione ad assumere il ruolo, in un file denominato`iot-role-trust.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
        },
```

```
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
            },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:iot:us-east-1:123456789012:rule/
rulename"
            }
        }
        }
    ]
}
```

Utilizzare il comando create-role per creare un ruolo IAM specificando il file `iot-role-trust.json`:

```
aws iam create-role --role-name my-iot-role --assume-role-policy-document
 file://iot-role-trust.json
```

L'output di questo comando è simile al seguente:

```
{
 "Role": {
  "AssumeRolePolicyDocument": "url-encoded-json",
  "RoleId": "AKIAIOSFODNN7EXAMPLE",
  "CreateDate": "2015-09-30T18:43:32.821Z",
  "RoleName": "my-iot-role",
  "Path": "/",
  "Arn": "arn:aws:iam::123456789012:role/my-iot-role"
 }
}
```

2. Salvare il seguente JSON in un file denominato `my-iot-policy.json`.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
```

```
    "Action": "dynamodb:*",
    "Resource": "*"
   }
  ]
}
```

Questo JSON è un esempio di documento di policy che concede l'accesso AWS IoT dell'amministratore a DynamoDB.

Utilizzate il comando create-policy per concedere AWS IoT l'accesso alle vostre AWS risorse dopo aver assunto il ruolo, inserendo nel file: `my-iot-policy.json`

```
aws iam create-policy --policy-name my-iot-policy --policy-document file://my-iot-policy.json
```

Per ulteriori informazioni su come concedere l'accesso a Servizi AWS nelle politiche per AWS IoT, consulta. Creare una AWS IoT regola

L'output del comando create-policy contiene l'ARN della policy. Collegamento di una policy al ruolo.

```
{
 "Policy": {
  "PolicyName": "my-iot-policy",
  "CreateDate": "2015-09-30T19:31:18.620Z",
  "AttachmentCount": 0,
  "IsAttachable": true,
  "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
  "DefaultVersionId": "v1",
  "Path": "/",
  "Arn": "arn:aws:iam::123456789012:policy/my-iot-policy",
  "UpdateDate": "2015-09-30T19:31:18.620Z"
 }
}
```

3. Usa il attach-role-policycomando per associare la tua politica al tuo ruolo:

```
aws iam attach-role-policy --role-name my-iot-role --policy-arn
 "arn:aws:iam::123456789012:policy/my-iot-policy"
```

Revoca l'accesso al motore delle regole

Per revocare immediatamente l'accesso al Rule Engine, procedi come segue

1. Rimuovi iot.amazonaws.com dalla politica di fiducia
2. Segui i passaggi per revocare le sessioni di ruolo iot

Passaggio delle autorizzazioni ai ruoli

Parte di una definizione di regola è costituita da un ruolo IAM che concede l'autorizzazione di accesso alle risorse specificate nell'operazione della regola. Il motore di regole presuppone l'uso del ruolo quando l'operazione della regola viene richiamata. Il ruolo deve essere definito nello stesso modo della Account AWS regola.

Quando crei o sostituisci una regola, in realtà passi un ruolo al motore di regole. È necessaria l'autorizzazione `iam:PassRole` per eseguire questa operazione. Per verificare che disponi di questa autorizzazione, crea una policy che conceda l'autorizzazione `iam:PassRole` e collegala al tuo utente IAM. La policy seguente mostra come concedere l'autorizzazione `iam:PassRole` per un ruolo.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "Stmt1",
   "Effect": "Allow",
   "Action": [
    "iam:PassRole"
   ],
   "Resource": [
    "arn:aws:iam::123456789012:role/myRole"
   ]
  }
 ]
}
```

In questa policy di esempio viene concessa l'autorizzazione `iam:PassRole` al ruolo `myRole`. Il ruolo viene specificato usando l'ARN del ruolo. Collega questa policy all'utente IAM o al ruolo a cui l'utente appartiene. Per ulteriori informazioni, consulta la sezione relativa all'utilizzo di policy gestite.

> **Note**
>
> Le funzioni Lambda usano una policy basata sulle risorse collegata direttamente alla funzione Lambda stessa. Quando crei una regola che richiama una funzione Lambda, non trasferisci un ruolo; pertanto, l'utente che crea la regola non ha bisogno dell'autorizzazione `iam:PassRole`. Per ulteriori informazioni sull'autorizzazione per la funzione Lambda, consulta la pagina relativa alla Concessione di autorizzazioni usando una policy per le risorse.

Creare una AWS IoT regola

Puoi creare AWS IoT regole per indirizzare i dati dagli oggetti connessi all'interazione con altri AWS servizi. Una AWS IoT regola è composta dai seguenti componenti:

Componenti di una regola

Componente	Description	Obbligatorio o facoltativo
Nome regola	Nome della regola . Tieni presente che non consigliamo l'uso di informazioni di identificazione personale nei nomi delle regole.	Obbligatorio.
Descrizione della regola	Una descrizione testuale della regola. Tieni presente che non consigliamo l'uso di informazioni di identificazione personale nelle descrizioni delle regole.	Opzionale.
Istruzione SQL	Sintassi SQL semplificata per filtrare i messaggi ricevuti in un argomento MQTT ed effettuare il push dei dati in un'altra posizione. Per ulteriori informazioni, consulta AWS IoT Riferimento SQL.	Obbligatorio.

Componente	Description	Obbligatorio o facoltativo
Versione SQL	Versione del motore di regole SQL da usare durante la valutazione della regola. Sebbene questa proprietà sia facoltativa, si consiglia vivamente di specificare la versione SQL. La AWS IoT Core console imposta questa proprietà come `2016-03-23` predefinita. Se questa proprietà non è impostata, ad esempio in un AWS CLI comando o in un CloudFormation modello, `2015-10-08` viene utilizzata. Per ulteriori informazioni, consulta Versioni SQL.	Obbligatorio.
Una o più azioni	Le azioni AWS IoT eseguite quando si applica la regola. Ad esempio, è possibile inserire i dati in una tabella Dynamo DB, scrivere i dati in un bucket Amazon S3, eseguire la pubblicazione in un argomento Amazon SNS o invocare una funzione Lambda.	Obbligatorio.
Un'operazione in caso di errore	L'azione viene AWS IoT eseguita quando non è in grado di eseguire l'azione di una regola.	Opzionale.

Prima di creare una AWS IoT regola, devi creare un ruolo IAM con una policy che consenta l'accesso alle AWS risorse richieste. AWS IoT assume questo ruolo quando implementa una regola. Per ulteriori informazioni, vedere Concessione a una AWS IoT regola dell'accesso richiesto e Passaggio delle autorizzazioni di ruolo.

Quando crei una regola, fai attenzione alla quantità di dati che pubblichi negli argomenti. Se crei regole che includono uno schema di argomento con caratteri jolly, potrebbero corrispondere a una percentuale elevata dei tuoi messaggi. In tal caso, potrebbe essere necessario aumentare la capacità delle risorse AWS utilizzate dalle operazioni di destinazione. Consigliamo di evitare schemi di argomento con caratteri jolly nelle regole di ripubblicazione per evitare elaborazioni duplicate e ridurre i costi.

> **Note**
>
> La creazione e l'aggiornamento di regole sono operazioni a livello di amministratore. Qualsiasi utente che abbia l'autorizzazione per creare o aggiornare regole potrà accedere ai dati elaborati dalle regole.

Crea una regola (Console)

Per creare una regola (Console di gestione AWS)

Usa il Console di gestione AWScomando per creare una regola:

1. Apri la AWS IoT console.
2. Nella barra di navigazione a sinistra, scegli Instradamento dei messaggi dalla sezione Gestisci. Quindi scegli Regole.
3. Nella pagina Regole, scegli Crea regola.
4. Nella pagina Specificare le proprietà della regola, inserisci un nome per la regola. La descrizione e i tag della regola sono facoltativi. Scegli Next (Successivo).
5. Nella pagina Configura l'istruzione SQL, scegli una versione SQL e inserisci un'istruzione SQL. Un esempio di istruzione SQL può essere`SELECT temperature FROM 'iot/topic' WHERE temperature > 50`. Per ulteriori informazioni, vedere Versioni SQL e riferimenti AWS IoT SQL.
6. Nella pagina Allega azioni alle regole, aggiungi le azioni delle regole per indirizzare i dati ad altri AWS servizi.

    1. In Azioni relative alle regole, seleziona un'azione della regola dall'elenco a discesa. Ad esempio, puoi scegliere Kinesis Stream. Per ulteriori informazioni sulle azioni delle regole, consulta le azioni delle AWS IoT regole.
    2. A seconda dell'azione della regola scelta, inserisci i dettagli di configurazione correlati. Ad esempio, se scegli Kinesis Stream, dovrai scegliere o creare una risorsa di flusso di dati e, facoltativamente, inserire dettagli di configurazione come la chiave di partizione, che viene utilizzata per raggruppare i dati per shard in uno stream.
    3. Nel ruolo IAM, scegli o crea un ruolo per concedere AWS IoT l'accesso al tuo endpoint. Tieni presente che AWS IoT creerà automaticamente una policy con un prefisso corrispondente al ruolo IAM selezionato. `aws-iot-rule` Puoi scegliere Visualizza per visualizzare il tuo ruolo

IAM e la policy dalla console IAM. L'azione di errore è facoltativa. Ulteriori informazioni sono disponibili in Gestione degli errori (azione di errore). Per ulteriori informazioni sulla creazione di un ruolo IAM per la regola, consulta Concedere a una regola l'accesso richiesto. Scegli Next (Successivo).

7. Nella pagina Rivedi e crea, esamina tutta la configurazione e apporta le modifiche necessarie. Scegli Create (Crea).

Dopo aver creato correttamente una regola, la regola verrà visualizzata nella pagina Regole. È possibile selezionare una regola per aprire la pagina Dettagli in cui è possibile visualizzare una regola, modificare una regola, disattivarla ed eliminarla.

Creare una regola (CLI)

Per creare una regola (AWS CLI)

Usa il create-topic-rulecomando per creare una regola:

```
aws iot create-topic-rule --rule-name myrule --topic-rule-payload file://myrule.json
```

Di seguito è riportato un esempio di file di payload con una regola che inserisce tutti i messaggi inviati all'argomento `iot/test` nella tabella Dynamo DB specificata. L'istruzione SQL filtra i messaggi e il ruolo ARN concede il AWS IoT permesso di scrivere nella tabella DynamoDB.

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "dynamoDB": {
    "tableName": "my-dynamodb-table",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
    "hashKeyField": "topic",
    "hashKeyValue": "${topic(2)}",
    "rangeKeyField": "timestamp",
    "rangeKeyValue": "${timestamp()}"
   }
  }
 ]
}
```

Di seguito è riportato un esempio di file di payload con una regola che inserisce tutti i messaggi inviati all'argomento `iot/test` nel bucket S3 specificato. L'istruzione SQL filtra i messaggi e il ruolo ARN concede l' AWS IoT autorizzazione alla scrittura nel bucket Amazon S3.

```
{
 "awsIotSqlVersion": "2016-03-23",
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "actions": [
  {
   "s3": {
    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
    "bucketName": "amzn-s3-demo-bucket",
    "key": "myS3Key"
   }
  }
 ]
}
```

Di seguito è riportato un esempio di file di payload con una regola che invia i dati ad Amazon OpenSearch Service:

```
{
 "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "OpenSearch": {
    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es",
    "endpoint": "https://my-endpoint",
    "index": "my-index",
    "type": "my-type",
    "id": "${newuuid()}"
   }
  }
 ]
}
```

Di seguito è riportato un esempio di file di payload con una regola che richiama una funzione Lambda:

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "lambda": {
    "functionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-function"
   }
  }
 ]
}
```

Di seguito è riportato un esempio di file di payload con una regola che esegue la pubblicazione in un argomento Amazon SNS:

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "sns": {
    "targetArn": "arn:aws:sns:us-west-2:123456789012:my-sns-topic",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
   }
  }
 ]
}
```

Di seguito è riportato un esempio di file di payload con una regola che esegue la ripubblicazione in un argomento MQTT diverso:

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "republish": {
    "topic": "my-mqtt-topic",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
```

```
   }
  }
 ]
}
```

Di seguito è riportato un esempio di file di payload con una regola che invia i dati a un flusso Amazon Data Firehose:

```
{
 "sql": "SELECT * FROM 'my-topic'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "firehose": {
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
    "deliveryStreamName": "my-stream-name"
   }
  }
 ]
}
```

Di seguito è riportato un esempio di file di payload con una regola che utilizza la `machinelearning_predict` funzione Amazon SageMaker AI per ripubblicare su un argomento se i dati nel payload MQTT sono classificati come 1.

```
{
 "sql": "SELECT * FROM 'iot/test' where machinelearning_predict('my-model',
 'arn:aws:iam::123456789012:role/my-iot-aml-role', *).predictedLabel=1",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "republish": {
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
    "topic": "my-mqtt-topic"
   }
  }
 ]
}
```

Di seguito è riportato un esempio di file di payload con una regola che pubblica messaggi in un flusso di input Salesforce IoT Cloud.

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "salesforce": {
    "token": "ABCDEFGHI123456789abcdefghi123456789",
    "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/stream-id/
connection-id/my-event"
   }
  }
 ]
}
```

Di seguito è riportato un esempio di file di payload con una regola che avvia un'esecuzione di una macchina a stati Step Functions.

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "stepFunctions": {
    "stateMachineName": "myCoolStateMachine",
    "executionNamePrefix": "coolRunning",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
   }
  }
 ]
}
```

Gestione di una regola AWS IoT

È possibile utilizzare le seguenti azioni per gestire le AWS IoT regole.

In questo argomento:

- Taggare una regola
- Visualizzazione di una regola
- Eliminazione di una regola

Taggare una regola

Per aggiungere un altro livello di specificità alle regole nuove o esistenti, puoi applicare i tag. L'etichettatura sfrutta le coppie chiave-valore nelle regole per offrirti un maggiore controllo su come e dove le regole vengono applicate alle tue risorse e ai tuoi servizi. AWS IoT Ad esempio, puoi limitare l'ambito della regola in modo da applicarla solo nell'ambiente beta per i test precedenti al rilascio (`Key=environment, Value=beta`) o acquisendo tutti i messaggi inviati all'argomento `iot/test` da un endpoint specifico e archiviandoli in un bucket Amazon S3.

Esempio di policy IAM

Per un esempio che mostra come concedere le autorizzazioni per l'assegnazione di tag per una regola, prendi in considerazione un utente che esegue il comando seguente per creare una regola e aggiungervi un tag che si applica solo in un ambiente beta.

Nell'esempio, sostituisci:

- *`MyTopicRuleName`*con il nome della regola.
- *`myrule.json`*con il nome del documento di policy.

```
aws iot create-topic-rule
    --rule-name MyTopicRuleName
    --topic-rule-payload file://myrule.json
    --tags "environment=beta"
```

Per questo esempio, devi utilizzare la seguente policy IAM:

```
{
    "Version":"2012-10-17",
    "Statement":
    {
        "Action": [ "iot:CreateTopicRule", "iot:TagResource" ],
        "Effect": "Allow",
```

```
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:rule/MyTopicRuleName"
        ]
    }
}
```

L'esempio precedente mostra una regola appena creata denominata `MyTopicRuleName` che si applica solo all'ambiente beta. `iot:TagResource` nell'istruzione della policy con `MyTopicRuleName` specificatamente richiamato consente di assegnare tag durante la creazione o l'aggiornamento di `MyTopicRuleName`. Il parametro `--tags "environment=beta"` utilizzato durante la creazione della regola limita l'ambito di `MyTopicRuleName` solo all'ambiente beta. Se rimuovi il parametro `--tags "environment=beta"`, `MyTopicRuleName` si applicherà a tutti gli ambienti.

Per ulteriori informazioni relative alla creazione di policy e ruoli IAM specifici per una regola AWS IoT , consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto

Per informazioni sull'assegnazione di tag alle risorse, consulta Taggare le tue risorse AWS IoT.

Visualizzazione di una regola

Usa il list-topic-rulescomando per elencare le tue regole:

```
aws iot list-topic-rules
```

Usa il get-topic-rulecomando per ottenere informazioni su una regola:

```
aws iot get-topic-rule --rule-name myrule
```

Eliminazione di una regola

Quando una regola non è più necessaria, è possibile eliminarla.

Per eliminare una regola (AWS CLI)

Usa il delete-topic-rulecomando per eliminare una regola:

```
aws iot delete-topic-rule --rule-name myrule
```

AWS IoT azioni relative alle regole

AWS IoT le azioni delle regole specificano cosa fare quando viene richiamata una regola. Puoi definire azioni per inviare dati a un database Amazon DynamoDB, inviare dati ad Amazon Kinesis Data Streams, AWS Lambda richiamare una funzione e così via. AWS IoT supporta le seguenti azioni Regioni AWS laddove il servizio dell'azione è disponibile.

Operazione delle regole	Description	Nome nell'API
Apache Kafka	Invia un messaggio a un cluster Apache Kafka.	`kafka`
CloudWatch allarmi	Modifica lo stato di un CloudWatch allarme Amazon.	`cloudwatchAlarm`
CloudWatch Registri	Invia un messaggio ad Amazon CloudWatch Logs.	`cloudwatchLogs`
CloudWatch metriche	Invia un messaggio a una CloudWatch metrica.	`cloudwatchMetric`
DynamoDB	Invia un messaggio a una tabella Dynamo DB.	`dynamoDB`
Dynamo DBv2	Invia i dati dei messaggi a più colonne in una tabella Dynamo DB.	`dynamoDBv2`
Elasticsearch	Invia un messaggio a un OpenSearch endpoint.	`OpenSearch`
HTTP	Invia un messaggio a un endpoint HTTPS.	`http`
AWS IoT Events	Invia un messaggio a un AWS IoT Events input.	`iotEvents`

Operazione delle regole	Description	Nome nell'API
AWS IoT SiteWise	Invia i dati dei messaggi alle proprietà AWS IoT SiteWise delle risorse.	`iotSiteWise`
Firehose	Invia un messaggio a un flusso di distribuzione di Firehose.	`firehose`
Flussi di dati Kinesis	Invia un messaggio a un flusso dei dati Kinesis.	`kinesis`
Lambda	Richiama una funzione Lambda con i dati dei messaggi come input.	`lambda`
Location (Ubicazione)	Invia i dati sulla posizione al servizio di posizione Amazon.	`location`
OpenSearch	Invia un messaggio a un endpoint OpenSearch di Amazon Service.	`OpenSearch`
Ripubblica	Ripubblica un messaggio in un altro argomento MQTT.	`republish`
S3	Archivia un messaggio in un bucket Amazon Simple Storage Service (Amazon S3).	`s3`
IoT di Salesforce	Invia un messaggio a un flusso di input Salesforce IoT.	`salesforce`
SNS	Pubblica un messaggio come notifica push di Amazon Simple Notification Service (Amazon SNS).	`sns`

Operazione delle regole	Description	Nome nell'API
SQS	Pubblica un messaggio in una coda Amazon Simple Queue Service (Amazon SQS).	`sqs`
Step Functions	Avvia una macchina a AWS Step Functions stati.	`stepFunctions`
the section called “Timestream”	Invia un messaggio a una tabella di database Amazon Timestream.	`timestream`

Note

- Definisci la regola nella stessa Regione AWS risorsa di un altro servizio in modo che l'azione della regola possa interagire con quella risorsa.
- Il motore AWS IoT delle regole potrebbe effettuare più tentativi di eseguire un'azione se si verificano errori intermittenti. Se tutti i tentativi falliscono, il messaggio viene scartato e l'errore è disponibile nei registri. CloudWatch È possibile specificare un'operazione di errore per ciascuna regola che viene invocata quando si verifica un errore. Per ulteriori informazioni, consulta Gestione degli errori (operazione in caso di errore).
- Alcune operazioni delle regole attivano operazioni nei servizi che si integrano con AWS Key Management Service (AWS KMS) per supportare la crittografia dei dati a riposo. Se si utilizza una chiave KMS gestita dal cliente AWS KMS key per crittografare i dati inattivi, il servizio deve disporre dell'autorizzazione a utilizzare la chiave KMS per conto del chiamante. Per informazioni su come gestire le autorizzazioni per la chiave KSM gestita dal cliente, consulta gli argomenti sulla crittografia dei dati nella guida del servizio appropriata. Per ulteriori informazioni sulle chiavi KMS gestite dal cliente, consulta Concetti di AWS Key Management Service nella Guida per sviluppatori di AWS Key Management Service .

  È possibile utilizzare qualsiasi funzione o modello di sostituzione nell'istruzione SQL di un'azione di errore, incluse le funzioni esterne: `aws_lambda()`,,,, e. `get_dynamodb()get_registry_data()get_thing_shadow()get_secret()machinelear`

Se un'azione di errore richiede la chiamata di una funzione esterna, l'invocazione dell'azione di errore può comportare un fatturato aggiuntivo per la funzione esterna.

Apache Kafka

L'azione Apache Kafka (Kafka) invia messaggi direttamente ai tuoi cluster Amazon Managed Streaming for Apache Kafka (Amazon MSK), ai cluster Apache Kafka gestiti da provider di terze parti come Confluent Cloud o ai cluster Apache Kafka autogestiti. Con Kafka rule action, puoi indirizzare i tuoi dati IoT ai cluster Kafka. Ciò consente di creare pipeline di dati ad alte prestazioni per vari scopi, come analisi di streaming, integrazione dei dati, visualizzazione e applicazioni aziendali cruciali.

Note

Questo argomento presuppone la familiarità con la piattaforma Apache Kafka e i relativi concetti. Per ulteriori informazioni su Apache Kafka, consulta Apache Kafka. MSK Serverless non è supportato. I cluster MSK Serverless possono essere eseguiti solo tramite l'autenticazione IAM, che l'azione delle regole di Apache Kafka attualmente non supporta. Per ulteriori informazioni su come configurare AWS IoT Core con Confluent, consulta Sfruttare Confluent e AWS risolvere le sfide della gestione dei dati e dei dispositivi IoT.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire`ec2:CreateNetworkInterface`,,,`ec2:DescribeNetworkInterfaces`, `ec2:CreateNetworkInterfacePermission` `ec2:DeleteNetworkInterface``ec2:DescribeSubnets`, `ec2:DescribeVpcs` e operazioni. `ec2:DescribeVpcAttribute` `ec2:DescribeSecurityGroups` Questo ruolo crea e gestisce interfacce di rete elastiche per il tuo Amazon Virtual Private Cloud per raggiungere il tuo broker Kafka. Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT Core eseguire questa azione basata sulla regola.

Per maggiori informazioni sulle interfacce di rete, consulta Interfacce di rete elastiche nella Guida per l'utente di Amazon EC2.

La policy associata al ruolo che specifichi sarà simile a quella del seguente esempio.

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "ec2:CreateNetworkInterface",
            "ec2:DescribeNetworkInterfaces",
            "ec2:CreateNetworkInterfacePermission",
            "ec2:DeleteNetworkInterface",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeVpcAttribute",
            "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

- Se utilizzi Gestione dei segreti AWS per memorizzare le credenziali necessarie per connetterti al tuo broker Kafka, devi creare un ruolo IAM che AWS IoT Core possa assumersi di eseguire le `secretsmanager:GetSecretValue` operazioni and. `secretsmanager:DescribeSecret`

  La policy associata al ruolo che specifichi sarà simile a quella del seguente esempio.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
```

```
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:kafka_client_truststore-*",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:kafka_keytab-*"
            ]
        }
    ]
}
```

- Puoi eseguire i cluster Apache Kafka all'interno di Amazon Virtual Private Cloud (Amazon VPC). È necessario creare una destinazione Apache Kafka Virtual Private Cloud (VPC) e utilizzare un gateway NAT nelle sottoreti per inoltrare messaggi da un cluster Kafka pubblico. AWS IoT Il motore AWS IoT delle regole crea un'interfaccia di rete in ciascuna delle sottoreti elencate nella destinazione per indirizzare il traffico direttamente al VPC. Quando arrivi a destinazione, il motore AWS IoT delle regole crea automaticamente un'azione della regola VPC. Per ulteriori informazioni sulle operazioni delle regole VPC, consulta Destinazioni Apache Kafka Virtual Private Cloud (VPC).
- Se si utilizza una chiave gestita dal cliente AWS KMS key (chiave KMS) per crittografare i dati inattivi, il servizio deve disporre dell'autorizzazione a utilizzare la chiave KMS per conto del chiamante. Per ulteriori informazioni, consulta crittografia di Amazon MSK nella Guida per sviluppatori Amazon Managed Streaming for Apache Kafka.

Parameters

Quando crei una AWS IoT regola con questa azione, devi specificare le seguenti informazioni:

destinationArn

L'Amazon Resource Name (ARN) della destinazione Apache Kafka Virtual Private Cloud (VPC). Per informazioni sulla creazione di una destinazione, consulta. Destinazioni Apache Kafka Virtual Private Cloud (VPC)

topic

L'argomento Kafka per i messaggi da inviare al broker Kafka.

È possibile sostituire questo campo utilizzando un modello di sostituzione. Per ulteriori informazioni, consulta the section called “Modelli di sostituzione”.

chiave (opzionale)

La chiave del messaggio Kafka.

È possibile sostituire questo campo utilizzando un modello di sostituzione. Per ulteriori informazioni, consulta the section called “Modelli di sostituzione”.

intestazioni (opzionali)

L'elenco delle intestazioni specificate. Ogni intestazione è una coppia chiave-valore che puoi specificare quando crei un'operazione Kafka. Puoi utilizzare queste intestazioni per instradare i dati dai client IoT ai cluster Kafka a valle senza modificare il payload dei messaggi.

È possibile sostituire questo campo utilizzando un modello di sostituzione. Per capire come passare la funzione di una regola in linea come modello sostitutivo nell'intestazione dell'operazione Kafka, consulta Esempi. Per ulteriori informazioni, consulta the section called “Modelli di sostituzione”.

> **Note**
>
> Le intestazioni in formato binario non sono supportate.

partizione (opzionale)

La partizione del messaggio Kafka.

È possibile sostituire questo campo utilizzando un modello di sostituzione. Per ulteriori informazioni, consulta the section called “Modelli di sostituzione”.

clientProperties

Un oggetto che definisce le proprietà del client del produttore Apache Kafka.

conferme (facoltativo)

Il numero di conferme che, secondo il produttore, il server deve aver ricevuto, prima di considerare una richiesta come completa.

Se specifichi 0 come valore, il produttore non attenderà alcuna conferma da parte del server. Se il server non riceve il messaggio, il produttore non tenterà di inviare il messaggio un'altra volta.

Valori validi: `-1`, `0`, `1`, `all`. Il valore predefinito è `1`.

bootstrap.servers

Un elenco di coppie host e porta (ad esempio `host1:port1`, `host2:port2`) utilizzato per stabilire la connessione iniziale al cluster Kafka.

compression.type (opzionale)

Il tipo di compressione per tutti i dati generati dal produttore.

Valori validi: `none`, `gzip`, `snappy`, `lz4`, `zstd`. Il valore predefinito è `none`.

security.protocol

Il protocollo di sicurezza usato per collegarsi al broker Kafka.

Valori validi: `SSL`, `SASL_SSL`. Il valore predefinito è `SSL`.

key.serializer

Specifica come trasformare gli oggetti chiave che fornisci con il `ProducerRecord` in byte.

Valore valido: `StringSerializer`.

value.serializer

Specifica come trasformare gli oggetti valore che fornisci con il `ProducerRecord` in byte.

Valore valido: `ByteBufferSerializer`.

ssl.truststore

Il file truststore in formato base64 o la posizione del file truststore in Gestione dei segreti AWS. Questo valore non è richiesto se il tuo truststore è considerato attendibile dalle autorità di certificazione Amazon (CA).

Questo campo supporta i modelli di sostituzione. Se utilizzi Secrets Manager per archiviare le credenziali necessarie per connetterti al broker Kafka, puoi utilizzare la funzione SQL `get_secret` per recuperare il valore di questo campo. Per ulteriori informazioni sui modelli di sostituzione, consulta the section called “Modelli di sostituzione”. Per ulteriori informazioni sulla funzione SQL `get_secret`, consulta the section called “get_secret (secretId, secretType, chiave, roleArn)”. Se il truststore ha la forma di un file, usa il parametro `SecretBinary`. Se il truststore è sotto forma di una stringa, usa il parametro `SecretString`.

La dimensione massima di questo valore è di 65 KB.

ssl.truststore.password

La password del truststore. Questo valore è richiesto solo se è stata creata una password per il truststore.

ssl.keystore

Il file keystore. Questo valore è obbligatorio quando si specifica SSL come valore per `security.protocol`.

Questo campo supporta i modelli di sostituzione. Utilizza Secrets Manager per archiviare le credenziali necessarie per connetterti al broker Kafka. Utilizza la funzione SQL `get_secret` per recuperare il valore di questo campo. Per ulteriori informazioni sui modelli di sostituzione, consulta the section called “Modelli di sostituzione”. Per ulteriori informazioni sulla funzione SQL `get_secret`, consulta the section called “get_secret (secretId, secretType, chiave, roleArn)”. Utilizzo del parametro `SecretBinary`.

ssl.keystore.password

La password dell'archivio per il file keystore. Questo valore è obbligatorio se viene specificato un valore per `ssl.keystore`.

Il valore di questo campo può essere un testo semplice. Questo campo supporta anche i modelli di sostituzione. Utilizza Secrets Manager per archiviare le credenziali necessarie per connetterti al broker Kafka. Utilizza la funzione SQL `get_secret` per recuperare il valore di questo campo. Per ulteriori informazioni sui modelli di sostituzione, consulta the section called “Modelli di sostituzione”. Per ulteriori informazioni sulla funzione SQL `get_secret`, consulta the section called “get_secret (secretId, secretType, chiave, roleArn)”. Utilizzo del parametro `SecretString`.

ssl.key.password

La password della chiave privata nel file keystore.

Questo campo supporta i modelli di sostituzione. Utilizza Secrets Manager per archiviare le credenziali necessarie per connetterti al broker Kafka. Utilizza la funzione SQL `get_secret` per recuperare il valore di questo campo. Per ulteriori informazioni sui modelli di sostituzione, consulta the section called “Modelli di sostituzione”. Per ulteriori informazioni sulla funzione SQL `get_secret`, consulta the section called “get_secret (secretId, secretType, chiave, roleArn)”. Utilizzo del parametro `SecretString`.

sasl.mechanism

Il meccanismo di sicurezza utilizzato per connettersi al broker Kafka. Questo valore è obbligatorio quando si specifica `SASL_SSL` per `security.protocol`.

Valori validi: `PLAIN`, `SCRAM-SHA-512`, `GSSAPI`.

> **Note**
>
> `SCRAM-SHA-512`è l'unico meccanismo di sicurezza supportato nelle regioni cn-north-1, cn-northwest-1, -1 e -1. us-gov-east us-gov-west

sasl.plain.username

Il nome utente utilizzato per recuperare la stringa segreta da Secrets Manager. Questo valore è obbligatorio quando si specifica `SASL_SSL` per `security.protocol` e `PLAIN` per `sasl.mechanism`.

sasl.plain.password

La password utilizzata per recuperare la stringa segreta da Secrets Manager. Questo valore è obbligatorio quando si specifica `SASL_SSL` per `security.protocol` e `PLAIN` per `sasl.mechanism`.

sasl.scram.username

Il nome utente utilizzato per recuperare la stringa segreta da Secrets Manager. Questo valore è obbligatorio quando si specifica `SASL_SSL` per `security.protocol` e `SCRAM-SHA-512` per `sasl.mechanism`.

sasl.scram.password

La password utilizzata per recuperare la stringa segreta da Secrets Manager. Questo valore è obbligatorio quando si specifica `SASL_SSL` per `security.protocol` e `SCRAM-SHA-512` per `sasl.mechanism`.

sasl.kerberos.keytab

Il file keytab per l'autenticazione di Kerberos in Secrets Manager. Questo valore è obbligatorio quando si specifica `SASL_SSL` per `security.protocol` e `GSSAPI` per `sasl.mechanism`.

Questo campo supporta i modelli di sostituzione. Utilizza Secrets Manager per archiviare le credenziali necessarie per connetterti al broker Kafka. Utilizza la funzione SQL `get_secret`

per recuperare il valore di questo campo. Per ulteriori informazioni sui modelli di sostituzione, consulta the section called “Modelli di sostituzione”. Per ulteriori informazioni sulla funzione SQL `get_secret`, consulta the section called “get_secret (secretId, secretType, chiave, roleArn)”. Utilizzo del parametro `SecretBinary`.

sasl.kerberos.service.name

Il nome principale Kerberos con il quale Apache Kafka funziona. Questo valore è obbligatorio quando si specifica `SASL_SSL` per `security.protocol` e `GSSAPI` per `sasl.mechanism`.

sasl.kerberos.krb5.kdc

Il nome host del centro di distribuzione delle chiavi (KDC) a cui si connette il client produttore Apache Kafka. Questo valore è obbligatorio quando si specifica `SASL_SSL` per `security.protocol` e `GSSAPI` per `sasl.mechanism`.

sasl.kerberos.krb5.realm

Il dominio a cui si connette il client del produttore Apache Kafka. Questo valore è obbligatorio quando si specifica `SASL_SSL` per `security.protocol` e `GSSAPI` per `sasl.mechanism`.

sasl.kerberos.principal

L'identità Kerberos univoca a cui Kerberos può assegnare ticket per accedere ai servizi compatibili con Kerberos. Questo valore è obbligatorio quando si specifica `SASL_SSL` per `security.protocol` e `GSSAPI` per `sasl.mechanism`.

Esempi

Il seguente esempio JSON definisce un'azione di Apache Kafka in una regola. AWS IoT L'esempio seguente passa la funzione in linea sourceIp() come modello di sostituzione nell'intestazione dell'operazione Kafka.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "kafka": {
     "destinationArn": "arn:aws:iot:region:123456789012:ruledestination/vpc/
VPCDestinationARN",
```

```
      "topic": "TopicName",
      "clientProperties": {
       "bootstrap.servers": "kafka.com:9092",
       "security.protocol": "SASL_SSL",
       "ssl.truststore": "${get_secret('kafka_client_truststore',
  'SecretBinary','arn:aws:iam::123456789012:role/kafka-get-secret-role-name')}",
       "ssl.truststore.password": "kafka password",
       "sasl.mechanism": "GSSAPI",
       "sasl.kerberos.service.name": "kafka",
       "sasl.kerberos.krb5.kdc": "kerberosdns.com",
       "sasl.kerberos.keytab": "${get_secret('kafka_keytab','SecretBinary',
  'arn:aws:iam::123456789012:role/kafka-get-secret-role-name')}",
       "sasl.kerberos.krb5.realm": "KERBEROSREALM",
       "sasl.kerberos.principal": "kafka-keytab/kafka-keytab.com"
      },
      "headers": [
       {
        "key": "static_header_key",
        "value": "static_header_value"
       },
       {
        "key": "substitutable_header_key",
        "value": "${value_from_payload}"
       },
       {
        "key": "source_ip",
        "value": "${sourceIp()}"
       }
      ]
     }
    }
   ]
  }
}
```

Note importanti sulla configurazione di Kerberos

- Il centro di distribuzione delle chiavi (KDC) deve essere risolvibile tramite Domain Name System (DNS) privato all'interno del VPC di destinazione. Un possibile approccio è quello di aggiungere la voce DNS del KDC a una zona ospitata privata. Per ulteriori informazioni su questo approccio, consulta Utilizzo delle zone ospitate private.

- Ogni VPC deve avere la risoluzione DNS abilitata. Per ulteriori informazioni, consulta Utilizzo del DNS con il tuo VPC.
- I gruppi di sicurezza dell'interfaccia di rete e i gruppi di sicurezza a livello di istanza nella destinazione VPC devono consentire il traffico dall'interno del VPC sulle seguenti porte.
  - Traffico TCP sulla porta listener del broker bootstrap (spesso 9092, ma deve essere compreso nell'intervallo 9000-9100)
  - Traffico TCP e UDP sulla porta 88 per il KDC
- `SCRAM-SHA-512`è l'unico meccanismo di sicurezza supportato nelle regioni cn-north-1, cn-northwest-1, -1 e -1. us-gov-east us-gov-west

Destinazioni Apache Kafka Virtual Private Cloud (VPC)

L'operazione della regola di Apache Kafka instrada i dati a un cluster Apache Kafka in un Amazon Virtual Private Cloud (Amazon VPC). La configurazione VPC utilizzata dall'operazione della regola Apache Kafka viene attivata automaticamente quando si specifica la destinazione VPC per l'operazione della regola.

Una destinazione Virtual Private Cloud (VPC) di Apache Kafka contiene un elenco di sottoreti all'interno del VPC. Il motore delle regole crea un'interfaccia di rete elastica in ciascuna sottorete specificata nell'elenco. Per maggiori informazioni sulle interfacce di rete, consulta le Interfacce di rete elastiche nella Guida per l'utente di Amazon EC2.

Requisiti e considerazioni

- Se utilizzi un cluster Apache Kafka autogestito a cui si accede utilizzando un endpoint pubblico su Internet:
  - Crea un gateway NAT per le istanze nelle sottoreti. Il gateway NAT dispone di un indirizzo IP pubblico che può connettersi a Internet e consente al motore delle regole di inoltrare i messaggi al cluster Kafka pubblico.
  - Alloca un indirizzo IP elastico con le interfacce di rete elastiche (ENIs) create dalla destinazione Apache Kafka Virtual Private Cloud (VPC). I gruppi di sicurezza utilizzati devono essere configurati per bloccare il traffico in entrata.

> **Note**
>
> Se la destinazione Apache Kafka Virtual Private Cloud (VPC) è disabilitata e quindi riattivata, è necessario associare nuovamente l'elastico al nuovo. IPs ENIs

- Se una destinazione Apache Kafka Virtual Private Cloud (VPC) non riceve traffico per 30 giorni consecutivi, verrà disabilitata.
- Se una delle risorse utilizzate dalla destinazione del Virtual Private Cloud (VPC) di Apache Kafka cambia, la destinazione verrà disabilitata e non potrà essere utilizzata.
- Alcune modifiche che possono disabilitare una destinazione Virtual Private Cloud (VPC) di Apache Kafka includono:
  - Eliminazione del VPC, delle sottoreti, dei gruppi di sicurezza o del ruolo utilizzato.
  - Modifica del ruolo in modo che non disponga più delle autorizzazioni necessarie.
  - Il raggiungimento di una capacità prossima alla sottorete ci rende incapaci di applicare le patch FedRAMP.
  - Disattivazione della destinazione.

Prezzi

Ai fini della determinazione dei prezzi, un'operazione della regola VPC viene misurata in aggiunta all'azione che invia un messaggio a una risorsa quando la risorsa si trova nel VPC. Per informazioni sui prezzi, consulta Prezzi di AWS IoT Core.

Creazione di destinazioni Virtual Private Cloud (VPC) Apache Kafka

È possibile creare una destinazione Apache Kafka Virtual Private Cloud (VPC) utilizzando l'API o la console. CreateTopicRuleDestination AWS IoT Core

Quando si crea una destinazione, è necessario specificare le seguenti informazioni.

vpcId

L'ID univoco di Amazon VPC.

subnetIds

Un elenco di sottoreti in cui il motore delle regole crea interfacce di rete elastiche. Il motore delle regole alloca una singola interfaccia di rete per ogni sottorete nell'elenco.

SecurityGroups (facoltativa)

Un elenco di gruppi di sicurezza da applicare alle interfacce di rete.

roleArn

L'Amazon Resource Name (ARN) di un ruolo che ha l'autorizzazione a creare interfacce di rete per tuo conto.

A questo ARN dovrebbe essere associata una policy simile a quella del seguente esempio.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/VPCDestinationENI": "true"
            }
        }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateNetworkInterface",
                    "aws:RequestTag/VPCDestinationENI": "true"
            }
        }
        }
    ]
}
```

Creazione di una destinazione Virtual Private Cloud (VPC) Apache Kafka utilizzando AWS CLI

L'esempio seguente mostra come creare una destinazione utilizzando. AWS CLI

```
aws --region regions iot create-topic-rule-destination --destination-configuration
 'vpcConfiguration={subnetIds=["subnet-
12345678910123456"],securityGroups=[],vpcId="vpc-
12345678910123456",roleArn="arn:aws:iam::123456789012:role/role-name"}'
```

Dopo aver eseguito questo comando, lo stato della destinazione sarà`IN_PROGRESS`. Dopo alcuni istanti, il suo stato cambierà in `ERROR` (se il comando non ha esito positivo) o `ENABLED`. Quando lo stato di destinazione è `ENABLED`, la destinazione è pronta per l'uso.

È possibile utilizzare il seguente comando per ottenere lo stato della destinazione Apache Kafka Virtual Private Cloud (VPC).

```
aws --region region iot get-topic-rule-destination --arn "VPCDestinationARN"
```

Creazione di una destinazione Virtual Private Cloud (VPC) Apache Kafka utilizzando la console AWS IoT Core

I passaggi seguenti descrivono come creare una destinazione utilizzando la console. AWS IoT Core

1. Accedere alla AWS IoT Core console. Nel riquadro a sinistra, nella scheda Act (Atto), scegli Destinations (Destinazioni).
2. Inserisci i valori per i seguenti campi.

- ID VPC
- Sottorete IDs
- Gruppo di sicurezza

3. Seleziona un ruolo con le autorizzazioni necessarie per creare interfacce di rete. La policy di esempio precedente contiene queste autorizzazioni.

Quando lo stato di destinazione di Apache Kafka Virtual Private Cloud (VPC) è ABILITATO, è pronto per l'uso.

CloudWatch allarmi

L'azione CloudWatch alarm (`cloudWatchAlarm`) modifica lo stato di un CloudWatch allarme Amazon. In questa chiamata è possibile specificare il motivo della modifica dello stato e il valore.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'`cloudwatch:SetAlarmState`operazione. Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`alarmName`

Il nome CloudWatch dell'allarme.

Supporta modelli sostitutivi: solo API AWS CLI

`stateReason`

Motivo della modifica dell'allarme.

Supporta modelli di sostituzione: sì

`stateValue`

Valore dello stato dell'allarme. Valori validi: `OK`, `ALARM`, `INSUFFICIENT_DATA`.

Supporta modelli di sostituzione: sì

`roleArn`

Il ruolo IAM che consente l'accesso all' CloudWatch allarme. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione di CloudWatch allarme in una AWS IoT regola.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchAlarm": {
                    "alarmName": "IotAlarm",
                    "stateReason": "Temperature stabilized.",
                    "stateValue": "OK",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

Consulta anche

- Che cos'è Amazon CloudWatch? nella Amazon CloudWatch User Guide
- Utilizzo degli CloudWatch allarmi Amazon nella Guida per l' CloudWatch utente di Amazon

CloudWatch Registri

L'azione CloudWatch Logs (`cloudwatchLogs`) invia dati ad Amazon CloudWatch Logs. Puoi utilizzare `batchMode` per caricare e indicare data e ora di più record di log del dispositivo in un unico messaggio. Puoi anche specificare il gruppo di log in cui l'azione invia i dati.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere di eseguire le operazioni `logs:CreateLogStreamlogs:DescribeLogStreams`, `elogs:PutLogEvents`. Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo per consentire l'esecuzione di questa azione relativa AWS IoT alla regola.
- Se si utilizza una chiave gestita dal cliente AWS KMS key (chiave KMS) per crittografare i dati di registro in CloudWatch Logs, il servizio deve disporre dell'autorizzazione a utilizzare la chiave KMS per conto del chiamante. Per ulteriori informazioni, consulta Encrypt log data in CloudWatch Logs using AWS KMS nella Amazon CloudWatch Logs User Guide.

Requisiti relativi al formato del messaggio MQTT per **`batchMode`**

Se utilizzi l'azione della regola CloudWatch Logs con `batchMode` disattivata, non ci sono requisiti di formattazione dei messaggi MQTT. Nota: il valore predefinito del parametro `batchMode` è `false`. Tuttavia, se si utilizza l'azione della regola CloudWatch Logs con `batchMode` attivata (il valore del parametro è`true`), i messaggi MQTT contenenti registri lato dispositivo devono essere formattati in modo da contenere un timestamp e un payload dei messaggi. Nota: `timestamp` rappresenta l'ora in cui si è verificato l'evento ed è espressa come numero di millisecondi dopo le ore 00:00:00 UTC del 1° gennaio 1970.

Di seguito è riportato un esempio del formato di pubblicazione:

```
[
  {"timestamp": 1673520691093, "message": "Test message 1"},
  {"timestamp": 1673520692879, "message": "Test message 2"},
  {"timestamp": 1673520693442, "message": "Test message 3"}
]
```

A seconda di come vengono generati i log lato dispositivo, potrebbe essere necessario filtrarli e riformattarli prima che vengano inviati per soddisfare questo requisito. Per ulteriori informazioni, consulta Payload del messaggio MQTT.

Indipendentemente dal parametro, i contenuti devono rispettare i limiti di dimensione dei messaggi. `batchMode` `message` AWS IoT Per ulteriori informazioni, consulta Endpoint e quote per AWS IoT Core.

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`logGroupName`

Il gruppo di CloudWatch log a cui l'azione invia i dati.

Supporta modelli sostitutivi: API e solo AWS CLI

`roleArn`

Il ruolo IAM che consente l'accesso al gruppo di CloudWatch log. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

`batchMode` (facoltativo)

Indica se i batch di record di registro verranno estratti e caricati in. CloudWatch I valori includono `true` o `false` (impostazione predefinita). Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione CloudWatch Logs in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
```

```
            {
                "cloudwatchLogs": {
                    "logGroupName": "IotLogs",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw",
                    "batchMode": false
                }
            }
        ]
    }
}
```

Consulta anche

- Che cos'è Amazon CloudWatch Logs? nella Guida per l'utente di Amazon CloudWatch Logs

CloudWatch metriche

L'azione CloudWatch metric (`cloudwatchMetric`) acquisisce una metrica Amazon CloudWatch. È possibile specificare namespace, nome, valore, unità e timestamp per il parametro.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'operazione. `cloudwatch:PutMetricData` Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`metricName`

Il nome della CloudWatch metrica.

Supporta modelli di sostituzione: sì

`metricNamespace`

Il nome dello spazio dei nomi della CloudWatch metrica.

Supporta modelli di sostituzione: sì

`metricUnit`

L'unità metrica supportata da. CloudWatch

Supporta modelli di sostituzione: sì

`metricValue`

Una stringa che contiene il valore CloudWatch metrico.

Supporta modelli di sostituzione: sì

`metricTimestamp`

(Facoltativo) Una stringa che contiene l'orario espresso in secondi nel tempo di epoca Unix. Il valore predefinito utilizzato è il tempo di epoca Unix corrente.

Supporta modelli di sostituzione: sì

`roleArn`

Il ruolo IAM che consente l'accesso alla CloudWatch metrica. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione CloudWatch metrica in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchMetric": {
                    "metricName": "IotMetric",
                    "metricNamespace": "IotNamespace",
```

```
                    "metricUnit": "Count",
                    "metricValue": "1",
                    "metricTimestamp": "1456821314",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un'azione CloudWatch metrica con modelli di sostituzione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchMetric": {
                    "metricName": "${topic()}",
                    "metricNamespace": "${namespace}",
                    "metricUnit": "${unit}",
                    "metricValue": "${value}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

Consulta anche

- Che cos'è Amazon CloudWatch? nella Amazon CloudWatch User Guide
- Utilizzo dei CloudWatch parametri di Amazon nella Amazon CloudWatch User Guide

DynamoDB

L'operazione Dynamo DB (dynamoDB) scrive, completamente o in parte, un messaggio MQTT a una tabella Amazon DynamoDB.

È possibile seguire un tutorial che mostra come creare una regola con un'operazione Dynamo DB. Per ulteriori informazioni, consulta Tutorial: Archiviazione dei dati del dispositivo in una tabella DynamoDB.

> **Note**
>
> Questa regola scrive dati non JSON a Dynamo DB come dati binari. La console Dynamo DB mostra i dati come testo con codifica Base64.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'operazione. `dynamodb:PutItem` Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.
- Se si utilizza una chiave gestita dal cliente AWS KMS key (chiave KMS) per crittografare i dati inattivi in DynamoDB, il servizio deve disporre dell'autorizzazione a utilizzare la chiave KMS per conto del chiamante. Per ulteriori informazioni, consulta Chiave KMS gestita dal cliente nella Guida introduttiva di Amazon DynamoDB.

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`tableName`

Nome della tabella DynamoDB.

Supporta modelli sostitutivi: solo API AWS CLI

`hashKeyField`

Nome della chiave hash (detta anche chiave di partizione).

Supporta modelli sostitutivi: API e solo AWS CLI

`hashKeyType`

(Facoltativo) Tipo di dati della chiave hash (detta anche chiave di partizione). Valori validi: `STRING`, `NUMBER`.

Supporta modelli sostitutivi: API e solo AWS CLI

`hashKeyValue`

Valore della chiave hash. Prendere in considerazione l'utilizzo di un modello di sostituzione come `${topic()}` o `${timestamp()}`.

Supporta modelli di sostituzione: sì

`rangeKeyField`

(Facoltativo) Nome della chiave di intervallo (detta anche chiave di ordinamento).

Supporta modelli sostitutivi: API e solo AWS CLI

`rangeKeyType`

(Facoltativo) Tipo di dati della chiave di intervallo (detta anche chiave di ordinamento). Valori validi: `STRING`, `NUMBER`.

Supporta modelli sostitutivi: API e solo AWS CLI

`rangeKeyValue`

(Facoltativo) Valore della chiave di intervallo. Prendere in considerazione l'utilizzo di un modello di sostituzione come `${topic()}` o `${timestamp()}`.

Supporta modelli di sostituzione: sì

`payloadField`

(Facoltativo) Nome del campo in cui viene scritto il payload. Se questo valore viene omesso, il payload viene scritto nella colonna denominata `payload`.

Supporta modelli di sostituzione: sì

`operation`

(Facoltativo) Tipo di operazione da eseguire. Valori validi: `INSERT`, `UPDATE`, `DELETE`.

Supporta modelli di sostituzione: sì

`roleARN`

Il ruolo IAM che permette l'accesso alla tabella DynamoDB. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

I dati scritti nella tabella Dynamo DB sono il risultato dell'istruzione SQL della regola.

Esempi

Il seguente esempio JSON definisce un'azione DynamoDB in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * AS message FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "dynamoDB": {
                    "tableName": "my_ddb_table",
                    "hashKeyField": "key",
                    "hashKeyValue": "${topic()}",
                    "rangeKeyField": "timestamp",
                    "rangeKeyValue": "${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB"
                }
            }
        ]
    }
}
```

Consulta anche

- Cos'è Amazon DynamoDB? nella Guida per gli sviluppatori di Amazon DynamoDB
- Guida introduttiva su DynamoDB nella Guida per gli sviluppatori di Amazon DynamoDB
- Tutorial: Archiviazione dei dati del dispositivo in una tabella DynamoDB

Dynamo DBv2

L'azione Dynamo DBv2 (`dynamoDBv2`) scrive tutto o parte di un messaggio MQTT in una tabella Amazon DynamoDB. Ogni attributo nel payload viene scritto in una colonna separata del database Dynamo DB.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'operazione. `dynamodb:PutItem` Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.
- Il payload del messaggio MQTT deve contenere una chiave di livello root corrispondente alla chiave di partizione primaria della tabella e una chiave di livello root corrispondente alla chiave di ordinamento primaria della tabella, se definita.
- Se si utilizza una chiave gestita dal cliente AWS KMS key (chiave KMS) per crittografare i dati inattivi in DynamoDB, il servizio deve disporre dell'autorizzazione a utilizzare la chiave KMS per conto del chiamante. Per ulteriori informazioni, consulta Chiave KMS gestita dal cliente nella Guida introduttiva di Amazon DynamoDB.

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`putItem`

Un oggetto che specifica la tabella Dynamo DB in cui verranno scritti i dati del messaggio. Questo oggetto deve contenere le seguenti informazioni:

`tableName`

Nome della tabella DynamoDB.

Supporta modelli sostitutivi: solo API AWS CLI

`roleARN`

Il ruolo IAM che permette l'accesso alla tabella DynamoDB. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

I dati scritti nella tabella Dynamo DB sono il risultato dell'istruzione SQL della regola.

Esempi

Il seguente esempio JSON definisce un'azione Dynamo DBv2 in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * AS message FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "dynamoDBv2": {
                    "putItem": {
                        "tableName": "my_ddb_table"
                    },
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2",
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un'azione DynamoDB con modelli di sostituzione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2015-10-08",
        "actions": [
            {
                "dynamoDBv2": {
```

```
                    "putItem": {
                        "tableName": "${topic()}"
                    },
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2"
                }
            }
        ]
    }
}
```

Consulta anche

- Cos'è Amazon DynamoDB? nella Guida per gli sviluppatori di Amazon DynamoDB
- Guida introduttiva su DynamoDB nella Guida per gli sviluppatori di Amazon DynamoDB

Elasticsearch

L'azione Elasticsearch (`elasticsearch`) scrive i dati dai messaggi MQTT in un dominio Amazon OpenSearch Service. Puoi quindi utilizzare strumenti come OpenSearch Dashboards per interrogare e visualizzare i dati in Service. OpenSearch

> **Warning**
>
> L'operazione `Elasticsearch` può essere utilizzata solo da operazioni regola esistenti. Per creare una nuova operazione regola o per aggiornarne una esistente, utilizzare invece l'operazione regola `OpenSearch`. Per ulteriori informazioni, consulta OpenSearch.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'`es:ESHttpPut`operazione. Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.
- Se si utilizza una chiave gestita dal cliente AWS KMS key (chiave KMS) per crittografare i dati inattivi OpenSearch, il servizio deve disporre dell'autorizzazione a utilizzare la chiave KMS per

conto del chiamante. Per ulteriori informazioni, consulta Encryption of data at rest for Amazon OpenSearch Service nella Amazon OpenSearch Service Developer Guide.

Parameters

Quando crei una AWS IoT regola con questa azione, devi specificare le seguenti informazioni:

`endpoint`

Endpoint del dominio del tuo servizio.

Supporta modelli sostitutivi: solo API AWS CLI

`index`

Indice in cui archiviare i dati.

Supporta modelli di sostituzione: sì

`type`

Tipo di documento che stai archiviando.

Supporta modelli di sostituzione: sì

`id`

Identificatore univoco per ogni documento.

Supporta modelli di sostituzione: sì

`roleARN`

Il ruolo IAM che consente l'accesso al dominio del OpenSearch servizio. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione Elasticsearch in una AWS IoT regola e come specificare i campi per l'azione. `elasticsearch` Per ulteriori informazioni, consulta ElasticsearchAction.

```
{
```

```
    "topicRulePayload": {
        "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "elasticsearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "my-index",
                    "type": "my-type",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es"
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un'azione Elasticsearch con modelli di sostituzione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "elasticsearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "${topic()}",
                    "type": "${type}",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es"
                }
            }
        ]
    }
}
```

Consulta anche

- OpenSearch

- Cos'è Amazon OpenSearch Service?

HTTP

L'azione HTTPS (`http`) invia i dati da un messaggio MQTT a un endpoint HTTPS, che può puntare a un'applicazione o un servizio Web.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- È necessario confermare e abilitare gli endpoint HTTPS prima che il motore delle regole possa utilizzarli. Per ulteriori informazioni, consulta Destinazioni di azioni HTTP.

Parameters

Quando create una AWS IoT regola con questa azione, dovete specificare le seguenti informazioni:

`url`

L'endpoint HTTPS in cui il messaggio viene inviato utilizzando il metodo HTTP POST. Se si utilizza un indirizzo IP al posto di un nome host, deve essere un IPv4 indirizzo. IPv6 gli indirizzi non sono supportati.

Supporta modelli di sostituzione: sì

`confirmationUrl`

(Facoltativo) Se specificato, AWS IoT utilizza l'URL di conferma per creare una destinazione per la regola dell'argomento corrispondente. È necessario abilitare la destinazione dell'azione HTTP prima di utilizzarla in un'azione HTTP. Per ulteriori informazioni, consulta Destinazioni di azioni HTTP. Se si utilizzano modelli sostitutivi, è necessario creare manualmente una destinazione dell'azione HTTP prima di poter utilizzare l'`http`azione. `confirmationUrl`deve essere un prefisso di. `url`

La relazione tra `url` ed `confirmationUrl` è descritta come segue:

- Se `url` è codificato e non `confirmationUrl` viene fornito, trattiamo implicitamente il `url` campo come. `confirmationUrl` AWS IoT crea una destinazione per le regole tematiche per. `url`

- Se `url` e `confirmationUrl` sono codificati, `url` deve iniziare con. `confirmationUrl` AWS IoT crea una destinazione per le regole tematiche per. `confirmationUrl`
- Se `url` contiene un modello di sostituzione, è necessario specificare `confirmationUrl` e `url` deve cominciare con `confirmationUrl`. Se `confirmationUrl` contiene modelli di sostituzione, è necessario creare manualmente una destinazione di azione HTTP prima di poter utilizzare l'`http`azione. Se `confirmationUrl` non contiene modelli sostitutivi, AWS IoT crea una destinazione per le regole tematiche per. `confirmationUrl`

Supporta modelli di sostituzione: sì

`headers`

(Facoltativo) L'elenco delle intestazioni da includere nelle richieste HTTP all'endpoint. Ogni richiesta deve contenere le seguenti informazioni:

`key`

La chiave dell'intestazione.

Supporta modelli di sostituzione: no

`value`

Il valore dell'intestazione.

Supporta modelli di sostituzione: sì

> **Note**
>
> Il tipo di contenuto predefinito è application/json quando il payload è in formato JSON. Altrimenti, è application/octet-stream. È possibile sovrascriverlo specificando il tipo di contenuto esatto nell'intestazione con il tipo di contenuto chiave (senza distinzione tra maiuscole e minuscole).

`auth`

(Facoltativo) L'autenticazione utilizzata dal motore di regole per connettersi all'URL dell'endpoint specificato nell'argomento `url`. Attualmente, la versione 4 della firma è l'unico tipo di autenticazione supportato. Per ulteriori informazioni, vedere l'Autorizzazione HTTP.

Supporta modelli di sostituzione: no

`enableBatching`

(Facoltativo) Se elaborare i messaggi di azione HTTP in un'unica richiesta per un determinato URL. Il valore può essere vero o falso. Per ulteriori informazioni sul batch, vedere Creazione in batch dei messaggi di azione HTTP.

Valore booleano

Supporta modelli di sostituzione: no

`batchConfig`

(Facoltativo) Impostazioni di configurazione per il batch. Una volta abilitati, è necessario specificare `batchConfig` i parametri. Se `batchConfig` i parametri non vengono specificati, verranno utilizzati i valori predefiniti.

`maxBatchOpenMs`

Il periodo di tempo massimo (in millisecondi) in cui un messaggio in uscita attende che altri messaggi creino il batch. Maggiore è l'impostazione, maggiore è la latenza dell'azione HTTP in batch.

Valore minimo: 5 ms. Valore massimo: 200 ms.

Valore predefinito: 20 ms

Supporta modelli di sostituzione: no

`maxBatchSize`

Il numero massimo di messaggi raggruppati in un'unica esecuzione di azione.

Supporta modelli di sostituzione: no

Valore minimo: 2 messaggi. Valore massimo: 10 messaggi

Valore predefinito: 10 messaggi

`maxBatchSizeBytes`

Dimensione massima di un batch di messaggi, in byte.

Valore minimo: 100 byte. Valore massimo: 131.072 byte

Valore predefinito: 5.120 byte

Supporta modelli di sostituzione: no

> **Note**
>
> Il tipo di contenuto predefinito è application/json quando il payload è in formato JSON. Altrimenti, è application/octet-stream. È possibile sovrascriverlo specificando il tipo di contenuto esatto nell'intestazione con il tipo di contenuto chiave (senza distinzione tra maiuscole e minuscole).

Esempi

Il seguente esempio JSON definisce una AWS IoT regola con un'azione HTTP.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "http": {
                    "url": "https://www.example.com/subpath",
                    "confirmationUrl": "https://www.example.com",
                    "headers": [
                        {
                            "key": "static_header_key",
                            "value": "static_header_value"
                        },
                        {
                            "key": "substitutable_header_key",
                            "value": "${value_from_payload}"
                        }
                    ]
                }
            }
        ]
    }
}
```

```
"http": {
```

```
    "url": "https://www.example.com/subpath",
    "confirmationUrl": "https://www.example.com",
    "headers": [
        {
            "key": "Content-Type",
            "value": "application/json"
          }
    ],
    "enableBatching": true,
    "batchConfig": {
      "maxBatchOpenMs": 123,
      "maxBatchSize": 5,
      "maxBatchSizeBytes": 131072,
     }
 },
 "errorAction": {
        "http": {
            "url": "https://www.example.com/subpath",
            "confirmationUrl": "https://www.example.com"
            // batchConfig is not allowed here
        }
}
```

Logica di ripetizione dell'operazione HTTP

Il motore AWS IoT delle regole ritenta l'azione HTTP in base alle seguenti regole:

- Il motore delle regole tenta di inviare un messaggio almeno una volta.
- Il motore delle regole effettua un massimo di due tentativi. Il numero massimo di tentativi è tre.
- Il motore delle regole non effettua un nuovo tentativo se:
  - Il tentativo precedente ha fornito una risposta maggiore di 16.384 byte.
  - Il servizio Web downstream o l'applicazione chiude la connessione TCP dopo il tentativo.
  - Il tempo totale per completare una richiesta con nuovi tentativi ha superato il limite di timeout della richiesta.
  - La richiesta restituisce un codice di stato HTTP diverso da 429, 500-599.

> **Note**
>
> I costi standard di trasferimento dei dati si applicano ai nuovi tentativi.

Consulta anche

- Messaggi di azione HTTP in batch
- Destinazioni di azioni HTTP
- Indirizza i dati AWS IoT Core direttamente dai tuoi servizi Web nell'Internet of Things sul blog AWS

Messaggi di azione HTTP in batch

È possibile utilizzare il batch per inviare più messaggi di azione HTTP in un'unica richiesta.

Panoramica di

Il batching consente di inviare messaggi da AWS IoT Core Rules Engine agli endpoint HTTP in batch. Questa funzionalità può aiutare a ridurre i costi riducendo il numero di esecuzioni di azioni HTTP e a migliorare l'efficienza riducendo il sovraccarico associato alla creazione di nuove connessioni.

> **Note**
>
> L'azione HTTP in batch viene misurata come singola azione. Il calcolo viene effettuato in incrementi di 5 KiB, in base alla dimensione del payload in batch in uscita emesso dal Rules Engine al servizio downstream. AWS IoT Core Per ulteriori informazioni, consultare la pagina dei prezzi di AWS IoT Core.

Quando abiliti il batching nella definizione della tua IoT Rule Action, i seguenti parametri saranno disponibili per la configurazione:

`maxBatchOpenMs`

La quantità massima di tempo (in millisecondi) in cui un messaggio in uscita attende che altri messaggi creino il batch. Maggiore è l'impostazione, maggiore è la latenza dell'azione HTTP in batch.

Valore minimo: 5 ms. Valore massimo: 200 ms.

Valore predefinito: 20 ms

Supporta modelli di sostituzione: no

`maxBatchSize`

Il numero massimo di messaggi raggruppati in una singola esecuzione di un'azione di regole IoT.

Valore minimo: 2 messaggi. Valore massimo: 10 messaggi

Valore predefinito: 10 messaggi

Supporta modelli di sostituzione: no

`maxBatchSizeBytes`

Dimensione massima di un batch di messaggi, in byte.

Valore minimo: 100 byte. Valore massimo: 131.072 byte

Valore predefinito: 5120 byte

Supporta modelli di sostituzione: no

> **Important**
>
> Quando si specificano più parametri batch, il batch viene completato quando viene raggiunto il primo limite. Ad esempio, se si specifica 100 ms come Tempo di apertura massimo del batch e 5 KiB come dimensione massima del batch e Rules Engine esegue batch solo 2 KiB entro 100 ms, verrà creato e inviato un batch da 2 KiB.

Utilizzo delle intestazioni HTTP in un batch

Quando si utilizzano le intestazioni nell'azione HTTP, la richiesta in batch utilizza il valore di intestazione dell'ultimo messaggio aggiunto al batch (non necessariamente l'ultimo messaggio pubblicato). Ti consigliamo di utilizzare valori di intestazione che siano i seguenti:

- Identici in tutti i messaggi del batch
- Applicabile a tutti i messaggi (ad esempio, credenziali di autenticazione)

Le intestazioni vengono inviate con la richiesta HTTP e non fanno parte del corpo del messaggio.

> **Note**
>
> Quando il batching è abilitato:

- La richiesta in batch include automaticamente l'`Content-Type: application/json`intestazione, poiché il batch viene inviato come array JSON.
- Non possiamo garantire che l'ultimo messaggio del batch sia l'ultimo messaggio che hai pubblicato. È l'ultimo messaggio che è stato inserito nel batch.

Esempio di payload

L'esempio seguente mostra la struttura di un payload di messaggi in batch inviato all'endpoint HTTP:

```
[
  {
    "user_id": "user1",
    "steps_today": 1000
  },
  {
    "user_id": "user2",
    "steps_today": 21000
  },
  {
    "user_id": "user8",
    "steps_today": 1500
  },
  ...
]
```

Limitazioni

Di seguito sono riportate le limitazioni relative al batching:

- AWS IoT Core non garantisce l'ordinamento complessivo dei messaggi. Il batch viene eseguito localmente su ciascun host, il che può comportare l'elaborazione dei messaggi all'interno di un batch in un ordine diverso da quello di ricezione.
- AWS IoT Core non fornisce supporto per l'elaborazione dei messaggi sul lato del destinatario. L'utente è responsabile di garantire che il servizio downstream sia configurato per accettare ed elaborare i dati in batch.
- Il batching tra account non è supportato, anche se i messaggi sono destinati allo stesso identificatore di risorsa (URL HTTP o ARN della risorsa).

- AWS IoT Core non garantisce che la dimensione del batch soddisfi la configurazione specificata. I batch possono essere inferiori ai limiti configurati in base alla tempistica e al flusso dei messaggi.
- Quando il batching è abilitato, i payload binari (dati non UTF-8) non sono supportati. Sono accettati solo payload di testo UTF-8 (come JSON). Per inviare dati binari, base64 li codifica prima di inviarli all'azione HTTP, quindi li decodifica sull'endpoint ricevente. Ad esempio, è possibile utilizzare la funzione encode nelle regole IoT per codificare il payload binario. In alternativa, puoi codificare il payload binario nel tuo dispositivo IoT e pubblicarlo su. AWS IoT Core

Azioni di errore per il batching

Non sarà possibile definire una logica di batch separata nella definizione dell'azione di errore. Tuttavia, l'azione di errore supporterà il batch se è stata definita la logica di batch nell'azione principale.

Quando una richiesta batch ha esito negativo, il motore AWS IoT Core Rules seguirà la logica di ripetizione dell'azione HTTP. Dopo l'ultimo tentativo, verrà richiamata un'azione di errore per l'intero batch non riuscito.

Di seguito è riportato un esempio di messaggio di errore con il batching abilitato:


```
{
    "ruleName": "FailedTopicRule",
    "topic": "topic/rulesengine",
    "payloadsWithMetadata": [
        {
            "id": 1,
            "cloudwatchTraceId": "bebd6d93-6d4a-899e-9e40-56e82252d2be",
            "clientId": "Test",
            "sourceIp": "10.0.0.0",
            "base64OriginalPayload":
 "eyJ1c2VyX2lkIjogInVzZXI1NjQ3IiwgInN0ZXBzX3RvZGF5IjogMTMzNjUsICJ0aW1lc3RhbXAiOiAiMjAyNS0xMC0wO
        },
        {
            "id": 2,
            "cloudwatchTraceId": "af94d3b8-0b18-1dbf-2c7d-513f5cb9e2e1",
            "clientId": "Test",
            "sourceIp": "10.0.0.0",
            "base64OriginalPayload":
 "eyJ1c2VyX2lkIjogInVzZXI1NjQ3IiwgInN0ZXBzX3RvZGF5IjogMTMzNjUsICJ0aW1lc3RhbXAiOiAiMjAyNS0xMC0wO
        },
        {
```

```
            "id": 3,
            "cloudwatchTraceId": "ca441266-c2ce-c916-6aee-b9e5c7831675",
            "clientId": "Test",
            "sourceIp": "10.0.0.0",
            "base64OriginalPayload":
 "eyJ1c2VyX2lkIjogInVzZXI1NjQ3IiwgInN0ZXBzX3RvZGF5IjogMTMzNjUsICJ0aW1lc3RhbXAiOiAiMjAyNS0xMC0wO
        }
    ],
    "failures": [
        {
            "affectedIds": [
                1,
                2,
                3
            ],
            "failedAction": "HttpAction",
            "failedResource": "https://example.foobar.com/HttpAction",
            "errorMessage": "HttpAction failed to make a request to the specified
 endpoint. StatusCode: 500. Reason: Internal Server Error."
        },
        {
            "affectedIds": [
                3
            ],
            "failedAction": "S3Action",
            "failedResource": "amzn-s3-demo-bucket",
            "errorMessage": "Failed to put S3 object. The error received was The
 specified bucket does not exist"
        },
        {
            "affectedIds": [
                3
            ],
            "failedAction": "LambdaAction",
            "failedResource": "arn:aws:lambda:us-west-2:123456789012:function:dummy",
            "errorMessage": "Failed to invoke lambda function. Received Server error
 from Lambda. The error code is 403"
        }
    ]
}
```

> **Note**
>
> Gli errori di azione in batch generano anche payload di azioni di errore più grandi, il che può aumentare la probabilità di errori dovuti alle dimensioni. È possibile monitorare gli errori relativi alle azioni di errore utilizzando la metrica. `ErrorActionFailure` Per ulteriori informazioni, consulta Parametri delle operazioni sulle regole.

Combinazione in batch dei messaggi di azione HTTP con AWS CLI

Creazione o aggiornamento di un'azione di regola con batch

1. Usa il AWS CLI comando appropriato per creare o aggiornare una regola:
   - Per creare una nuova regola, usa il create-topic-rulecomando:

     ```
     aws iot create-topic-rule --rule-name myrule --topic-rule-payload
      file://myrule.json
     ```

   - Per aggiornare una regola esistente, usa il replace-topic-rulecomando:

     ```
     aws iot replace-topic-rule --rule-name myrule --topic-rule-payload
      file://myrule.json
     ```

2. Abilita le funzionalità di batch impostando il parametro enableBatching su true nel payload della regola dell'argomento:

```
{
        "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "http": {
                    "url": "https://www.example.com/subpath",
                    "confirmationUrl": "https://www.example.com",
                    "headers": [
                        {
                            "key": "static_header_key",
                            "value": "static_header_value"
                        },
```

```
                        {
                            "key": "substitutable_header_key",
                            "value": "${value_from_payload}"
                         }
                    ],
                    "enableBatching": true,
                    "batchConfig": {
                       "maxBatchOpenMs": 100,
                       "maxBatchSize": 5,
                       "maxBatchSizeBytes": 1024
                    }
                }
            }
      ]
}
```

3. Configura i parametri di batch. Non è necessario specificare tutti i parametri del batch. È possibile scegliere di specificare 1, 2 o tutti e 3 i parametri del batch. Se non si specifica un parametro batch, Rules Engine aggiornerà tale parametro con i valori predefiniti. Per ulteriori informazioni sui parametri di batch e sui relativi valori predefiniti, vedere Parametri HTTP.

Destinazioni di azioni HTTP

Una destinazione di azione HTTP è un servizio Web verso il quale il motore delle regole può indirizzare i dati da una regola tematica. Una AWS IoT Core risorsa descrive il servizio Web per AWS IoT. Le risorse di destinazione possono essere condivise secondo regole diverse.

Prima di AWS IoT Core poter inviare dati a un altro servizio Web, è necessario confermare che può accedere all'endpoint del servizio.

Panoramica di

Una destinazione di azione HTTP si riferisce a un servizio Web che supporta un URL di conferma e una o più raccolte URLs di dati. La risorsa di destinazione contiene l'URL di conferma del servizio Web. Quando configuri un'azione HTTP, specifichi l'URL effettivo dell'endpoint che deve ricevere i dati insieme all'URL di conferma del servizio web. Una volta che la destinazione è confermata, la regola dell'argomento invia il risultato dell'istruzione SQL all'endpoint HTTPS (e non all'URL di conferma).

Una destinazione di azione HTTP può trovarsi in uno dei seguenti stati:

ABILITATO

La destinazione è stata confermata e può essere utilizzata da un'operazione della regola. Una destinazione deve essere nello stato `ENABLED` per essere utilizzata in una regola. È possibile abilitare solo una destinazione che si trovi nello stato DISABLED.

DISABILITATO

La destinazione è stata confermata ma non può essere utilizzata da un'operazione della regola. Ciò è utile se si desidera impedire temporaneamente il traffico verso l'endpoint senza dover ripetere il processo di conferma. È possibile disabilitare solo una destinazione che si trovi nello stato ENABLED.

IN_PROGRESS

La conferma della destinazione è in corso.

ERRORE

La conferma della destinazione è scaduta.

Dopo che una destinazione di azione HTTP è stata confermata e abilitata, può essere utilizzata con qualsiasi regola del tuo account.

Gestione delle destinazioni di azione HTTP

È possibile utilizzare le seguenti operazioni per gestire le destinazioni delle azioni HTTP.

Creazione di destinazioni d'azione HTTP

È possibile creare una destinazione di azione HTTP chiamando l'`CreateTopicRuleDestination`operazione o utilizzando la AWS IoT console.

Dopo aver creato una destinazione, AWS IoT invia una richiesta di conferma all'URL di conferma. La richiesta di conferma ha il seguente formato:

```
HTTP POST {confirmationUrl}/?confirmationToken={confirmationToken}
Headers:
x-amz-rules-engine-message-type: DestinationConfirmation
x-amz-rules-engine-destination-arn:"arn:aws:iot:us-east-1:123456789012:ruledestination/
http/7a280e37-b9c6-47a2-a751-0703693f46e4"
Content-Type: application/json
Body:
```

```
{
    "arn":"arn:aws:iot:us-east-1:123456789012:ruledestination/http/7a280e37-b9c6-47a2-
a751-0703693f46e4",
    "confirmationToken": "AYADeMXLrPrNY2wqJAKsFNn-…NBJndA",
    "enableUrl": "https://iot.us-east-1.amazonaws.com/confirmdestination/
AYADeMXLrPrNY2wqJAKsFNn-…NBJndA",
    "messageType": "DestinationConfirmation"
}
```

Il contenuto della richiesta di conferma include le informazioni seguenti:

arn

L'Amazon Resource Name (ARN) per la destinazione dell'azione HTTP da confermare.

confirmationToken

Il token di conferma inviato da AWS IoT Core. Il token nell'esempio viene troncato. Il token sarà più lungo. Avrai bisogno di questo token per confermare la tua destinazione con AWS IoT Core.

enableUrl

L'URL che si utilizza per confermare la destinazione di una regola dell'argomento.

messageType

Il tipo di messaggio.

Conferma delle destinazioni di azione HTTP

Per completare il processo di conferma dell'endpoint, se utilizzi il AWS CLI, devi eseguire i seguenti passaggi dopo che l'URL di conferma ha ricevuto la richiesta di conferma.

1. Verifica che la destinazione sia pronta per ricevere messaggi

   Per confermare che la destinazione dell'azione HTTP è pronta a ricevere messaggi IoT, `enableUrl` chiamala nella richiesta di conferma oppure esegui l'operazione `ConfirmTopicRuleDestination` API e passa la richiesta `confirmationToken` di conferma.
2. Imposta lo stato della regola dell'argomento su abilitato

   Dopo aver confermato che la destinazione può ricevere messaggi, devi eseguire l'operazione `UpdateTopicRuleDestination` API per impostare lo stato della regola dell'argomento su`ENABLED`.

Se utilizzi la AWS IoT console, copiala `confirmationToken` e incollala nella finestra di dialogo di conferma della destinazione nella AWS IoT console. Puoi quindi abilitare la regola dell'argomento.

Invio di una nuova richiesta di conferma

Per attivare un nuovo messaggio di conferma per una destinazione, chiama `UpdateTopicRuleDestination` e imposta lo stato della destinazione della regola dell'argomento su `IN_PROGRESS`.

Ripeti la procedura di conferma dopo aver inviato una nuova richiesta di conferma.

Disabilitazione ed eliminazione di una destinazione di azione HTTP

Per disabilitare una destinazione, chiamare `UpdateTopicRuleDestination` e impostare lo stato della destinazione della regola dell'argomento su `DISABLED`. Una regola dell'argomento nello stato DISABLED può essere nuovamente abilitata senza dover inviare una nuova richiesta di conferma.

Per eliminare una destinazione di azione HTTP, chiama. `DeleteTopicRuleDestination`

Supporto per l'autorità di certificazione

> **Note**
>
> I certificati autofirmati non sono supportati.

Gli endpoint HTTPS in una destinazione di azione HTTP supportano i certificati emessi sia da AWS Private Certificate Authority che da Let's Encrypt.

AWS IoT Events

L'azione AWS IoT Events (`iotEvents`) invia i dati da un messaggio MQTT a un AWS IoT Events input.

> **Important**
>
> Se il payload viene inviato AWS IoT Core senza o se la chiave non si trova nello stesso percorso JSON specificato nella chiave, la regola IoT non funzionerà con l'errore. `Input attribute Key Failed to send message to Iot Events`

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere di eseguire l'`iotevents:BatchPutMessage`operazione. Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`batchMode`

(Facoltativo) Indica se elaborare le azioni evento come batch. Il valore predefinito è `false`.

Quando l'istruzione SQL `batchMode` is `true` e la regola restituisce un Array, ogni elemento Array viene trattato come un messaggio separato quando viene inviato a AWS IoT Events tramite una chiamata `BatchPutMessage`. L'array risultante non può avere più di 10 messaggi.

Quando `batchMode` è `true`, non è possibile specificare un `messageId`.

Supporta modelli di sostituzione: no

`inputName`

Il nome dell' AWS IoT Events input.

Supporta modelli sostitutivi: solo API AWS CLI

`messageId`

(Facoltativo) Utilizzatelo per verificare che un solo input (messaggio) con un dato dato `messageId` venga elaborato da un AWS IoT Events rilevatore. Puoi utilizzare il modello di sostituzione `${newuuid()}` per generare un ID univoco per ogni richiesta.

Quando `batchMode` è `true`, non è possibile specificare un valore `messageId`: verrà assegnato un nuovo valore UUID.

Supporta modelli di sostituzione: sì

`roleArn`

Il ruolo IAM che consente di AWS IoT inviare un input a un AWS IoT Events rilevatore. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

L'esempio JSON seguente illustra come definire un'operazione Eventi IoT in una regola AWS IoT .

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotEvents": {
                    "inputName": "MyIoTEventsInput",
                    "messageId": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_events"
                }
            }
        ]
    }
}
```

Consulta anche

- Che cos'è AWS IoT Events? nella Guida per gli AWS IoT Events sviluppatori

AWS IoT SiteWise

L'azione AWS IoT SiteWise (`iotSiteWise`) invia i dati da un messaggio MQTT alle proprietà dell'asset in. AWS IoT SiteWise

Puoi seguire un tutorial che mostra come importare dati dagli AWS IoT oggetti. Per ulteriori informazioni, consulta il tutorial Ingesting data to AWS IoT SiteWise from AWS IoT things o la sezione Ingesting data using AWS IoT Core rules nella Guida per l'utente.AWS IoT SiteWise

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'operazione. `iotsitewise:BatchPutAssetPropertyValue` Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  È possibile collegare al ruolo la seguente policy di fiducia di esempio.

  ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": "iotsitewise:BatchPutAssetPropertyValue",
              "Resource": "*"
          }
      ]
  }
  ```

  Per migliorare la sicurezza, è possibile specificare un percorso di gerarchia AWS IoT SiteWise degli asset nella `Condition` proprietà. L'esempio seguente è una policy di attendibilità che specifica un percorso della gerarchia di un asset.

  ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": "iotsitewise:BatchPutAssetPropertyValue",
              "Resource": "*",
              "Condition": {
                  "StringLike": {
                      "iotsitewise:assetHierarchyPath": [
                          "/root node asset ID",
                          "/root node asset ID/*"
  ```

```
                    ]
                }
            }
            }
        ]
}
```

- Quando si inviano dati a AWS IoT SiteWise con questa azione, i dati devono soddisfare i requisiti dell'`BatchPutAssetPropertyValue`operazione. Per ulteriori informazioni, consulta BatchPutAssetPropertyValue nella documentazione di riferimento dell'API AWS IoT SiteWise .

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`putAssetPropertyValueEntries`

Un elenco di voci del valore della proprietà dell'asset che contengono ciascuna le seguenti informazioni:

`propertyAlias`

(Facoltativo) L'alias di proprietà associato alla proprietà dell'asset. Specifica un `propertyAlias` o un `assetId` e un `propertyId` insieme. Per ulteriori informazioni sugli alias di proprietà, consulta Mappatura dei flussi di dati industriali alle proprietà degli asset nella Guida per l'utente AWS IoT SiteWise .

Supporta modelli di sostituzione: sì

`assetId`

(Facoltativo) L'ID della AWS IoT SiteWise risorsa. Specifica un `propertyAlias` o un `assetId` e un `propertyId` insieme.

Supporta modelli di sostituzione: sì

`propertyId`

(Facoltativo) L'ID di una proprietà di asset. Specifica un `propertyAlias` o un `assetId` e un `propertyId` insieme.

Supporta modelli di sostituzione: sì

`entryId`

(Facoltativo) Un identificatore univoco per questa voce. Definisci `entryId` per monitorare meglio il messaggio che ha causato un errore, se del caso. Il valore predefinito è un nuovo UUID.

Supporta modelli di sostituzione: sì

`propertyValues`

Un elenco di valori di proprietà da inserire che contengono timestamp, quality e value (TQV) nel formato seguente:

`timestamp`

Una struttura timestamp contenente le seguenti informazioni:

`timeInSeconds`

Una stringa che contiene il tempo in secondi nel tempo di epoca Unix. Se il payload del messaggio non ha un timestamp, è possibile utilizzare timestamp(), che restituisce l'ora corrente in millisecondi. Per convertire il tempo in secondi, è possibile utilizzare il seguente modello di sostituzione: **`${floor(timestamp() / 1E3)}`**.

Supporta modelli di sostituzione: sì

`offsetInNanos`

(Facoltativo) Una stringa che contiene l'offset del tempo in nanosecondi dal tempo espresso in secondi. Se il payload del messaggio non ha un timestamp, è possibile utilizzare timestamp(), che restituisce l'ora corrente in millisecondi. Per calcolare l'offset del nanosecondo da quel momento, è possibile utilizzare il seguente modello di sostituzione: **`${(timestamp() % 1E3) * 1E6}`**.

Supporta modelli di sostituzione: sì

Per quanto riguarda Unix epoch Time, AWS IoT SiteWise accetta solo le voci che hanno un timestamp che va da un massimo di 7 giorni nel passato fino a 5 minuti nel futuro.

`quality`

(Facoltativo) Una stringa che descrive la qualità del valore. Valori validi: `GOOD`, `BAD`, `UNCERTAIN`.

Supporta modelli di sostituzione: sì

`value`

Una struttura di valori che contiene uno dei seguenti campi valore, a seconda del tipo di dati della proprietà dell'asset:

`booleanValue`

(Facoltativo) Una stringa che contiene il valore booleano della voce di valore.

Supporta modelli di sostituzione: sì

`doubleValue`

(Facoltativo) Una stringa che contiene il doppio valore della voce di valore.

Supporta modelli di sostituzione: sì

`integerValue`

(Facoltativo) Una stringa che contiene il valore intero della voce di valore.

Supporta modelli di sostituzione: sì

`stringValue`

(Facoltativo) Il valore stringa della voce valore.

Supporta modelli di sostituzione: sì

`roleArn`

L'ARN del ruolo IAM a cui concede l' AWS IoT autorizzazione a inviare il valore di una proprietà dell'asset a. AWS IoT SiteWise Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un' SiteWise azione IoT di base in una AWS IoT regola.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
```

```
            {
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "/some/property/alias",
                            "propertyValues": [
                                {
                                    "timestamp": {
                                        "timeInSeconds": "${my.payload.timeInSeconds}"
                                    },
                                    "value": {
                                        "integerValue": "${my.payload.value}"
                                    }
                                }
                            ]
                        }
                    ],
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un' SiteWise azione IoT in una AWS IoT regola. In questo esempio vengono utilizzati l'argomento come alias di proprietà e la funzione `timestamp()`. Ad esempio, se si pubblicano i dati su `/company/windfarm/3/turbine/7/rpm`, questa operazione invia i dati alla proprietà dell'asset con un alias di proprietà uguale all'argomento specificato.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM '/company/windfarm/+/turbine/+/+'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "${topic()}",
                            "propertyValues": [
                                {
                                    "timestamp": {
```

```
                                    "timeInSeconds": "${floor(timestamp() / 1E3)}",
                                    "offsetInNanos": "${(timestamp() % 1E3) * 1E6}"
                                },
                                "value": {
                                    "doubleValue": "${my.payload.value}"
                                }
                            }
                        ]
                    }
                ],
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
            }
        }
    ]
  }
}
```

Consulta anche

- Cos'è AWS IoT SiteWise? nella Guida per l'utente di AWS IoT SiteWise
- Inserimento di dati utilizzando AWS IoT Core le regole della Guida per l'utente AWS IoT SiteWise
- Inserimento di dati AWS IoT SiteWise da AWS IoT elementi contenuti nella Guida per l'utente AWS IoT SiteWise
- Risoluzione dei problemi relativi a un'azione AWS IoT SiteWise relativa alle regole nella Guida per l'utente AWS IoT SiteWise

Firehose

L'azione Firehose (`firehose`) invia dati da un messaggio MQTT a uno stream Amazon Data Firehose.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'operazione. `firehose:PutRecord` Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.

- Se utilizzi Firehose per inviare dati a un bucket Amazon S3 e utilizzi AWS KMS un cliente che è AWS KMS key riuscito a crittografare i dati inattivi in Amazon S3, Firehose deve avere accesso al tuo bucket e il permesso di utilizzarlo per conto del chiamante. AWS KMS key Per ulteriori informazioni, consulta Concedere a Firehose l'accesso a una destinazione Amazon S3 nella Amazon Data Firehose Developer Guide.

Parameters

Quando crei una AWS IoT regola con questa azione, devi specificare le seguenti informazioni:

`batchMode`

(Facoltativo) Se distribuire lo stream Firehose come batch utilizzando. `PutRecordBatch` Il valore predefinito è `false`.

Quando `batchMode` è `true` e l'istruzione SQL della regola restituisce un array, ogni elemento dell'array forma un record nella richiesta `PutRecordBatch`. L'array risultante non può avere più di 500 record.

Supporta modelli di sostituzione: no

`deliveryStreamName`

Lo stream Firehose in cui scrivere i dati dei messaggi.

Supporta modelli sostitutivi: solo API AWS CLI

`separator`

(Facoltativo) Un separatore di caratteri utilizzato per separare i record scritti nel flusso Firehose. Se ometti questo parametro, il flusso non utilizza alcun separatore. Valori validi: `,` (virgola), `\t` (scheda), `\n` (nuova riga), `\r\n` (Nuova riga di Windows).

Supporta modelli di sostituzione: no

`roleArn`

Il ruolo IAM che consente l'accesso allo stream Firehose. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione Firehose in AWS IoT una regola.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "firehose": {
                    "deliveryStreamName": "my_firehose_stream",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose"
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un'azione Firehose con modelli di sostituzione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "firehose": {
                    "deliveryStreamName": "${topic()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose"
                }
            }
        ]
    }
}
```

Consulta anche

- Cos'è Amazon Data Firehose? nella Guida per sviluppatori di Amazon Data Firehose

Flussi di dati Kinesis

L'operazione Kinesis Data Streams (`kinesis`) scrive i dati da un messaggio MQTT in Amazon Kinesis Data Streams.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'operazione. `kinesis:PutRecord` Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.
- Se utilizzi una chiave KMS AWS KMS gestita dal cliente AWS KMS key per crittografare i dati inattivi in Kinesis Data Streams, il servizio deve disporre dell'autorizzazione a utilizzarla per conto del chiamante. AWS KMS key Per ulteriori informazioni, consulta Autorizzazioni per l'uso di AWS KMS keys generate dall'utente nella Guida per gli sviluppatori di Amazon Kinesis Data Streams.

Parameters

Quando crei una AWS IoT regola con questa azione, devi specificare le seguenti informazioni:

`stream`

Flusso di dati Kinesis in cui scrivere i dati.

Supporta modelli sostitutivi: solo API AWS CLI

`partitionKey`

Chiave di partizione usata per determinare in quale shard vengono scritti i dati. La chiave di partizione è in genere composta da un'espressione (ad esempio, `${topic()}` o `${timestamp()}`).

Supporta modelli di sostituzione: sì

`roleArn`

L'ARN del ruolo IAM che concede l' AWS IoT autorizzazione per accedere al flusso di dati Kinesis. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione Kinesis Data Streams in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "kinesis": {
                    "streamName": "my_kinesis_stream",
                    "partitionKey": "${topic()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis"
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un'azione Kinesis con modelli di sostituzione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "kinesis": {
                    "streamName": "${topic()}",
                    "partitionKey": "${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis"
                }
            }
        ]
    }
}
```

Consulta anche

- Cos'è Amazon Kinesis Data Streams? nella Guida per gli sviluppatori Amazon Kinesis Data Streams

Lambda

Un'azione Lambda (`lambda`) richiama una AWS Lambda funzione, passando un messaggio MQTT. AWS IoT richiama le funzioni Lambda in modo asincrono.

È possibile seguire un tutorial che mostra come creare e testare una regola con un'operazione Lambda. Per ulteriori informazioni, consulta Tutorial: Formattazione di una notifica utilizzando una funzione AWS Lambda.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- AWS IoT Per richiamare una funzione Lambda, è necessario configurare una politica che conceda l'autorizzazione `lambda:InvokeFunction` a. AWS IoT Puoi richiamare solo una funzione Lambda definita nella Regione AWS stessa in cui esiste la tua policy Lambda. Le funzioni Lambda usano policy basate sulle risorse, quindi è necessario collegare la policy alla funzione Lambda stessa.

  Usa il AWS CLI comando seguente per allegare una politica che conceda l'autorizzazione. `lambda:InvokeFunction` In questo comando, sostituisci:

  - *`function_name`*con il nome della funzione Lambda. Aggiungi una nuova autorizzazione per aggiornare la policy delle risorse della funzione.
  - *`region`*con Regione AWS la funzione.
  - *`account-id`*con il Account AWS numero in cui è definita la regola.
  - *`rule-name`*con il nome della AWS IoT regola per la quale si sta definendo l'azione Lambda.
  - *`unique_id`*con un identificatore di dichiarazione univoco.

> **Important**
>
> Se aggiungi un'autorizzazione per un' AWS IoT entità principale senza fornire `source-arn` o`source-account`, chiunque Account AWS crei una regola con l'azione Lambda può attivare regole da cui richiamare la funzione Lambda. AWS IoT

Per ulteriori informazioni, consulta Autorizzazioni di AWS Lambda.

```
aws lambda add-permission \
    --function-name function_name \
    --region region \
    --principal iot.amazonaws.com \
    --source-arn arn:aws:iot:region:account-id:rule/rule_name \
    --source-account account-id
    --statement-id unique_id
    --action "lambda:InvokeFunction"
```

- Se si utilizza la AWS IoT console per creare una regola per l'azione della regola Lambda, la funzione Lambda viene attivata automaticamente. Se si utilizza AWS CloudFormation invece con `AWS::IoT::TopicRule LambdaAction`, è necessario aggiungere una risorsa. `AWS::lambda::Permission` La risorsa ti concede quindi il permesso di attivare la funzione Lambda.

  Il codice seguente mostra un esempio di come aggiungere questa risorsa. In questo esempio, sostituisci:

  - *`function_name`*con il nome della funzione Lambda.
  - *`region`*con Regione AWS la funzione.
  - *`account-id`*con il Account AWS numero in cui è definita la regola.
  - *`rule-name`*con il nome della AWS IoT regola per la quale si sta definendo l'azione Lambda.

```
Type: AWS::Lambda::Permission
Properties:
  Action: lambda:InvokeFunction
  FunctionName: !Ref function_name
  Principal: "iot.amazonaws.com"
  SourceAccount: account-id
  SourceArn: arn:aws:iot:region:account-id:rule/rule_name
```

- Se utilizzi un AWS KMS cliente gestito AWS KMS key per crittografare i dati inattivi in Lambda, il servizio deve avere l'autorizzazione a utilizzarli per conto AWS KMS key del chiamante. Per ulteriori informazioni, consulta Crittografia dei dati a riposo nella Guida per gli sviluppatori di AWS Lambda .

Parameters

Quando crei una AWS IoT regola con questa azione, devi specificare le seguenti informazioni:

`functionArn`

L'ARN della funzione Lambda da richiamare. AWS IoT deve avere il permesso di richiamare la funzione. Per ulteriori informazioni, consulta Requisiti.

Se non specifici una versione o un alias per la funzione Lambda, la versione più recente della funzione viene arrestata. Se desideri arrestare una versione specifica della funzione Lambda, puoi specificare una versione o un alias. Per specificare una versione o un alias, aggiungere la versione o l'alias all'ARN della funzione Lambda.

```
arn:aws:lambda:us-east-2:123456789012:function:myLambdaFunction:someAlias
```

Per ulteriori informazioni sulla funzione di controllo delle versioni e sugli alias, consulta Funzione di controllo delle versioni e degli alias in AWS Lambda.

Supporta modelli sostitutivi: API e solo AWS CLI

Esempi

Il seguente esempio JSON definisce un'azione Lambda in AWS IoT una regola.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "lambda": {
                    "functionArn": "arn:aws:lambda:us-
east-2:123456789012:function:myLambdaFunction"
```

```
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un'azione Lambda con modelli di sostituzione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "lambda": {
                    "functionArn": "arn:aws:lambda:us-east-1:123456789012:function:
${topic()}"
                }
            }
        ]
    }
}
```

Consulta anche

- AWS Lambda Che cos'è? nella Guida per gli AWS Lambda sviluppatori
- Tutorial: Formattazione di una notifica utilizzando una funzione AWS Lambda

Location (Ubicazione)

L'operazione (`location`) Posizione instrada i dati sulla posizione geografica al servizio di posizione Amazon.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'`geo:BatchUpdateDevicePosition`operazione. Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`deviceId`

L'ID univoco del dispositivo che fornisce i dati sulla posizione. Per ulteriori informazioni, consulta `DeviceId` dalla Documentazione di riferimento delle API del servizio di posizione Amazon.

Supporta modelli di sostituzione: sì

`latitude`

Una stringa che restituisce un valore doppio che rappresenta la latitudine della posizione del dispositivo.

Supporta modelli di sostituzione: sì

`longitude`

Una stringa che restituisce un valore doppio che rappresenta la longitudine della posizione del dispositivo.

Supporta modelli di sostituzione: sì

`roleArn`

Il ruolo IAM che permette di accedere al dominio del servizio di posizione Amazon. Per ulteriori informazioni, consulta Requisiti.

`timestamp`

L'ora in cui i dati sulla posizione sono stati campionati. Il valore predefinito è l'ora in cui è stato elaborato il messaggio MQTT.

Il valore `timestamp` è composto dai due valori seguenti:

- `value`: un'espressione che restituisce un valore temporale periodo lungo. Puoi utilizzare la funzione the section called “time_to_epoch(String, String)” per creare un timestamp valido da un valore di data o ora passato nel payload del messaggio. Supporta modelli di sostituzione: Sì.
- `unit`: (facoltativo) la precisione del valore timestamp risultante dall'espressione descritta in `value`. Valori validi: `SECONDS` | `MILLISECONDS` | `MICROSECONDS` | `NANOSECONDS`. Il valore predefinito è `MILLISECONDS`. Supporta modelli sostitutivi: solo API. AWS CLI

`trackerName`

Il nome della risorsa tracker nel servizio di posizione Amazon in cui la posizione viene aggiornata. Per ulteriori informazioni, consulta Tracker nella Guida per gli sviluppatori del servizio di posizione Amazon.

Supporta modelli sostitutivi: API e solo AWS CLI

Esempi

Il seguente esempio JSON definisce un'azione Location in una regola. AWS IoT

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "location": {
     "roleArn": "arn:aws:iam::123454962127:role/service-role/ExampleRole",
     "trackerName": "MyTracker",
     "deviceId": "001",
     "sampleTime": {
      "value": "${timestamp()}",
      "unit": "MILLISECONDS"
     },
     "latitude": "-12.3456",
     "longitude": "65.4321"
    }
   }
  ]
 }
}
```

Il seguente esempio JSON definisce un'azione Location con modelli di sostituzione in una regola. AWS IoT

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "location": {
     "roleArn": "arn:aws:iam::123456789012:role/service-role/ExampleRole",
     "trackerName": "${TrackerName}",
     "deviceId": "${DeviceID}",
     "timestamp": {
      "value": "${timestamp()}",
      "unit": "MILLISECONDS"
     },
     "latitude": "${get(position, 0)}",
     "longitude": "${get(position, 1)}"
    }
   }
  ]
 }
}
```

Il seguente esempio di payload MQTT mostra in che modo i modelli di sostituzione nell'esempio precedente accedono ai dati. È possibile utilizzare il comando get-device-position-history della CLI per verificare che i dati del payload MQTT vengano distribuiti nel tracker di posizione.

```
{
 "TrackerName": "mytracker",
 "DeviceID": "001",
 "position": [
  "-12.3456",
  "65.4321"
 ]
}
```

```
aws location get-device-position-history --device-id 001 --tracker-name mytracker
```

```
{
 "DevicePositions": [
  {
   "DeviceId": "001",
   "Position": [
    -12.3456,
    65.4321
   ],
   "ReceivedTime": "2022-11-11T01:31:54.464000+00:00",
   "SampleTime": "2022-11-11T01:31:54.308000+00:00"
  }
 ]
}
```

Consulta anche

- Cos'è il servizio di posizione Amazon? nella Guida per sviluppatori del servizio di posizione Amazon.

OpenSearch

L'azione OpenSearch (`openSearch`) scrive i dati dai messaggi MQTT in un dominio Amazon OpenSearch Service. Puoi quindi utilizzare strumenti come le OpenSearch dashboard per interrogare e visualizzare i dati in Service. OpenSearch

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'`es:ESHttpPut`operazione. Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.
- Se utilizzi un cliente che è riuscito AWS KMS key a crittografare i dati archiviati nel OpenSearch Servizio, il servizio deve disporre dell'autorizzazione a utilizzare la chiave KMS per conto del chiamante. Per ulteriori informazioni, consulta Encryption of data at rest for Amazon OpenSearch Service nella Amazon OpenSearch Service Developer Guide.

Parameters

Quando crei una AWS IoT regola con questa azione, devi specificare le seguenti informazioni:

`endpoint`

L'endpoint del tuo dominio Amazon OpenSearch Service.

Supporta modelli sostitutivi: solo API AWS CLI

`index`

L' OpenSearch indice in cui desideri archiviare i tuoi dati.

Supporta modelli di sostituzione: sì

`type`

Tipo di documento che stai archiviando.

> **Note**
>
> Per OpenSearch le versioni successive alla 1.0, il valore del `type` parametro deve essere`_doc`. Per ulteriori informazioni, consulta la documentazione relativa ad OpenSearch .

Supporta modelli di sostituzione: sì

`id`

Identificatore univoco per ogni documento.

Supporta modelli di sostituzione: sì

`roleARN`

Il ruolo IAM che consente l'accesso al dominio del OpenSearch servizio. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Limitazioni

L'azione OpenSearch (`openSearch`) non può essere utilizzata per fornire dati ai cluster VPC Elasticsearch.

Esempi

Il seguente esempio JSON definisce un' OpenSearch azione in una AWS IoT regola e come specificare i campi per l'azione. `OpenSearch` Per ulteriori informazioni, consulta OpenSearchAction.

```
{
    "topicRulePayload": {
        "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "openSearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "my-index",
                    "type": "_doc",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un' OpenSearch azione con modelli di sostituzione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "openSearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "${topic()}",
                    "type": "${type}",
```

```
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
                }
            }
        ]
    }
}
```

> **Note**
>
> Il `type` campo sostituito funziona per la versione 1.0. OpenSearch Per tutte le versioni successive alla 1.0, il valore di `type` deve essere. `_doc`

Consulta anche

Cos'è Amazon OpenSearch Service? nella Amazon OpenSearch Service Developer Guide

Ripubblica

L'operazione di ripubblicazione (`republish`) ripubblica un messaggio MQTT in un altro argomento MQTT.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'operazione. `iot:Publish` Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`headers`

Informazioni sulle intestazioni MQTT versione 5.0.

Per ulteriori informazioni, consulta le pagine RepublishAction e MqttHeaders nella Documentazione di riferimento dell'API AWS .

`topic`

Argomento MQTT in cui ripubblicare il messaggio.

Per ripubblicare in un argomento riservato, che inizia con `$`, usa `$$`. Ad esempio, se stai ripubblicando in un argomento Device Shadow `$aws/things/MyThing/shadow/update`, specifica l'argomento come `$$aws/things/MyThing/shadow/update`.

> **Note**
>
> La ripubblicazione in Argomenti di processi riservati non è supportata.
> AWS IoT Device Defender gli argomenti riservati non supportano la pubblicazione HTTP.

Supporta modelli di sostituzione: sì

`qos`

(Facoltativo) Il livello di qualità del servizio (QoS) da usare per la ripubblicazione dei messaggi. Valori validi: `0`, 1. Il valore predefinito è `0`. Per ulteriori informazioni su QoS MQTT, consulta MQTT.

Supporta modelli di sostituzione: no

`roleArn`

Il ruolo IAM che AWS IoT consente la pubblicazione sull'argomento MQTT. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione di ripubblicazione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
```

```
        "actions": [
            {
                "republish": {
                    "topic": "another/topic",
                    "qos": 1,
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un'azione di ripubblicazione con modelli di sostituzione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "${topic()}/republish",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un'azione di ripubblicazione con in una regola. `headers` AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
```

```
                    "topic": "${topic()}/republish",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
                    "headers": {
                        "payloadFormatIndicator": "UTF8_DATA",
                        "contentType": "rule/contentType",
                        "correlationData": "cnVsZSBjb3JyZWxhdGlvbiBkYXRh",
                        "userProperties": [
                            {
                                "key": "ruleKey1",
                                "value": "ruleValue1"
                            },
                            {
                                "key": "ruleKey2",
                                "value": "ruleValue2"
                            }
                        ]
                    }
                }
            }
        ]
    }
}
```

> **Note**
>
> L'IP di origine originale non verrà passato tramite l'operazione Republish.

S3

L’operazione S3 (`s3`) scrive i dati da un messaggio MQTT in un bucket Amazon Simple Storage Service (Amazon S3).

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'operazione. `s3:PutObject` Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.

- Se utilizzi un servizio AWS KMS gestito dal cliente AWS KMS key per crittografare i dati inattivi in Amazon S3, il servizio deve disporre dell'autorizzazione a utilizzarlo per AWS KMS key conto del chiamante. Per ulteriori informazioni, consulta AWS Managed AWS KMS keys e Customer managed AWS KMS keys nella Amazon Simple Storage Service Developer Guide.

Parameters

Quando crei una AWS IoT regola con questa azione, devi specificare le seguenti informazioni:

`bucket`

Il bucket Amazon S3 in cui scrivere i dati.

Supporta modelli sostitutivi: solo API AWS CLI

`cannedacl`

(Facoltativo) Lista di controllo degli accessi predefinita Amazon S3 che controlla l'accesso all'oggetto identificato dalla chiave dell'oggetto. Per ulteriori informazioni, inclusi i valori consentiti, consulta la pagina relativa alle liste di controllo degli accessi predefinite.

Supporta modelli di sostituzione: no

`key`

Percorso del file in cui vengono scritti i dati.

Considera un esempio in cui questo parametro è `${topic()}/${timestamp()}` e la regola riceve un messaggio in cui l'argomento è `some/topic`. Se il timestamp corrente è `1460685389`, questa operazione scrive i dati in un file chiamato `1460685389` nella cartella `some/topic` del bucket S3.

> **Note**
>
> Se utilizzi una chiave statica, AWS IoT sovrascrive un singolo file ogni volta che viene richiamata la regola. Consigliamo di utilizzare il timestamp di un messaggio o un altro identificatore univoco del messaggio in modo che, per ogni messaggio ricevuto, venga salvato un nuovo file in Amazon S3.

Supporta modelli di sostituzione: sì

`roleArn`

Il ruolo IAM che permette l'accesso al bucket Amazon S3. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione S3 in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "s3": {
                    "bucketName": "amzn-s3-demo-bucket",
                    "cannedacl": "public-read",
                    "key": "${topic()}/${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3"
                }
            }
        ]
    }
}
```

Consulta anche

- Cos'è Amazon S3? nella Guida per l'utente di Amazon Simple Storage Service

IoT di Salesforce

Un'operazione Saleforce IoT (`salesforce`) invia i dati dal messaggio MQTT che ha attivato la regola a un flusso di input Salesforce IoT.

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`url`

URL esposto dal flusso di input Salesforce IoT. L'URL è disponibile dalla piattaforma Salesforce IoT durante la creazione di un flusso di input. Per ulteriori informazioni, consulta la documentazione di Salesforce IoT.

Supporta modelli di sostituzione: no

`token`

Token usato per autenticare l'accesso al flusso di input Salesforce IoT specificato. Il token è disponibile dalla piattaforma Salesforce IoT durante la creazione di un flusso di input. Per ulteriori informazioni, consulta la documentazione di Salesforce IoT.

Supporta modelli di sostituzione: no

Esempi

L'esempio JSON seguente illustra come definire un'operazione Saleforce IoT in una regola AWS IoT .

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "salesforce": {
                    "token": "ABCDEFGHI123456789abcdefghi123456789",
                    "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/
stream-id/connection-id/my-event"
                }
            }
        ]
    }
}
```

SNS

L'operazione SNS (`sns`) invia i dati da un messaggio MQTT come notifica push Amazon Simple Notification Service (Amazon SNS).

È possibile seguire un tutorial che mostra come creare e testare una regola con un'operazione SNS. Per ulteriori informazioni, consulta Tutorial: Invio di una notifica Amazon SNS.

> **Note**
>
> L'operazione SNS non supporta Argomenti Amazon SNS FIFO (First-In-First-Out). Poiché il motore di regole è un servizio completamente distribuito, non vi è alcuna garanzia in merito all'ordine dei messaggi quando viene attivata l'operazione SNS.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'`sns:Publish`operazione. Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.
- Se utilizzi un AWS KMS servizio gestito dal cliente AWS KMS key per crittografare i dati inattivi in Amazon SNS, il servizio deve disporre dell'autorizzazione a utilizzarlo per AWS KMS key conto del chiamante. Per maggiori informazioni, consulta Gestione delle chiavi nel Guida per gli sviluppatori di Amazon Simple Notification Service.

Parameters

Quando crei una AWS IoT regola con questa azione, devi specificare le seguenti informazioni:

`targetArn`

Argomento SNS o singolo dispositivo a cui viene inviata la notifica push.

Supporta modelli sostitutivi: solo API AWS CLI

`messageFormat`

(Opzionale) Formato del messaggio. Amazon SNS usa questa impostazione per determinare se il payload deve essere analizzato e se le parti specifiche della piattaforma rilevanti del payload devono essere estratte. Valori validi: `JSON`, `RAW`. L'impostazione predefinita è `RAW`.

Supporta modelli di sostituzione: no

`roleArn`

Il ruolo IAM che permette l'accesso a SNS. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione SNS in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-east-2:123456789012:my_sns_topic",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un'azione SNS con modelli di sostituzione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
```

```
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-east-1:123456789012:${topic()}",
                    "messageFormat": "JSON",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
                }
            }
        ]
    }
}
```

Consulta anche

- Cos'è Amazon Simple Notification Service? nella Guida per gli sviluppatori di Amazon Simple Notification Service
- Tutorial: Invio di una notifica Amazon SNS

SQS

L'operazione SQS (`sqs`) invia i dati da un messaggio MQTT a una coda Amazon Simple Queue Service (Amazon SQS).

> **Note**
>
> L'operazione SQS non supporta Code Amazon SQS FIFO (First-In-First-Out). Poiché il motore di regole è un servizio completamente distribuito, non vi è alcuna garanzia in merito all'ordine dei messaggi quando viene attivata l'operazione SQS.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'operazione. `sqs:SendMessage` Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.

- Se utilizzi un AWS KMS cliente che è riuscito AWS KMS key a crittografare i dati inattivi in Amazon SQS, il servizio deve avere l'autorizzazione a utilizzarli per AWS KMS key conto del chiamante. Per ulteriori informazioni, consulta Gestione delle chiavi nella Guida per gli sviluppatori di Amazon Simple Queue Service.

Parameters

Quando crei una AWS IoT regola con questa azione, devi specificare le seguenti informazioni:

`queueUrl`

L'URL della coda Amazon SQS in cui scrivere i dati. Non è necessario che l'area in questo URL sia la stessa della tua Regione AWS AWS IoT regola.

> **Note**
>
> Potrebbero essere previsti costi aggiuntivi per il trasferimento incrociato dei dati Regioni AWS utilizzando l'azione della regola SQS. Per ulteriori informazioni, consulta i prezzi di Amazon SQS.

Supporta modelli sostitutivi: solo API AWS CLI

`useBase64`

Imposta questo parametro su `true` per configurare l'operazione della regola per codificare in base64 i dati del messaggio prima di scrivere i dati nella coda Amazon SQS. L'impostazione predefinita è `false`.

Supporta modelli di sostituzione: no

`roleArn`

Il ruolo IAM che permette l'accesso alla coda Amazon SQS. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione SQS in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sqs": {
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
my_sqs_queue",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs"
                }
            }
        ]
    }
}
```

Il seguente esempio JSON definisce un'azione SQS con modelli di sostituzione in una regola. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sqs": {
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
${topic()}",
                    "useBase64": true,
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs"
                }
            }
        ]
    }
}
```

Consulta anche

- Che cos'è Amazon Simple Queue Service? nella Guida per gli sviluppatori di Amazon Simple Queue Service

Step Functions

L'azione Step Functions (`stepFunctions`) avvia una macchina a AWS Step Functions stati.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire l'`states:StartExecution`operazione. Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.

Parameters

Quando si crea una AWS IoT regola con questa azione, è necessario specificare le seguenti informazioni:

`stateMachineName`

Il nome della macchina a stati Step Functions per avviare l'esecuzione.

Supporta modelli sostitutivi: solo API AWS CLI

`executionNamePrefix`

(Facoltativo) All'esecuzione della macchina a stati verrà assegnato un nome costituito da questo prefisso seguito da un UUID. Se non ne viene fornito uno, Step Functions crea automaticamente un nome univoco per ogni esecuzione della macchina a stati.

Supporta modelli di sostituzione: sì

`roleArn`

L'ARN del ruolo che concede il AWS IoT permesso di avviare la macchina a stati. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

Esempi

Il seguente esempio JSON definisce un'azione Step Functions in una AWS IoT regola.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "stepFunctions": {
                    "stateMachineName": "myStateMachine",
                    "executionNamePrefix": "myExecution",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_step_functions"
                }
            }
        ]
    }
}
```

Consulta anche

- Che cos'è AWS Step Functions? nella Guida per gli AWS Step Functions sviluppatori

Timestream

L'operazione della regola Timestream scrive attributi (misure) da un messaggio MQTT in una tabella Amazon Timestream. Per ulteriori informazioni su Amazon Timestream, consulta Che cos'è Amazon Timestream?.

> **Note**
>
> Amazon Timestream non è disponibile in tutti i sistemi. Regione AWS Se Amazon Timestream non è disponibile nella tua regione, non verrà visualizzato nell'elenco delle operazioni della regola.

Gli attributi memorizzati da questa regola nel database Timestream sono quelli derivanti dall'istruzione query della regola. Il valore di ogni attributo nel risultato dell'istruzione query viene analizzato per dedurre il suo tipo di dati (come in un'operazione the section called "Dynamo DBv2"). Il valore di ogni attributo viene scritto nel proprio record nella tabella Timestream. Per specificare o modificare il tipo di dati di un attributo, utilizza la funzione `cast()` nell'istruzione di query. Per ulteriori

informazioni sul contenuto di ogni record Timestream, consulta the section called “Contenuto del record Timestream”.

> **Note**
>
> Con SQL V2 (23/03/2016), i valori numerici che sono numeri interi, come `10.0`, vengono convertite la loro rappresentazione Integer (`10`). Trasmettendoli esplicitamente a un valore `Decimal`, ad esempio utilizzando la funzione cast(), non si impedisce questo comportamento: il risultato è ancora un valore `Integer`. Ciò può causare errori di mancata corrispondenza dei tipi, che impediscono la registrazione dei dati nel database Timestream. Per elaborare in modo affidabile valori numerici interi come i valori `Decimal`, utilizza SQL V1 (08/10/2015) per l'istruzione di query della regola.

> **Note**
>
> Il numero massimo di valori che un'azione regola Timestream può scrivere in una tabella Amazon Timestream è 100. Per ulteriori informazioni, consulta Riferimento della quota Amazon Timestream.

Requisiti

Questa operazione della regola presenta i seguenti requisiti:

- Un ruolo IAM che AWS IoT può assumere per eseguire le operazioni `timestream:DescribeEndpoints` and`timestream:WriteRecords`. Per ulteriori informazioni, consulta Concedere a qualsiasi AWS IoT regola l'accesso richiesto.

  Nella AWS IoT console, puoi scegliere, aggiornare o creare un ruolo che consenta di AWS IoT eseguire questa azione relativa alla regola.
- Se utilizzi un cliente- AWS KMS per crittografare i dati inattivi in Timestream, il servizio deve disporre dell'autorizzazione a utilizzarli per AWS KMS key conto del chiamante. Per ulteriori informazioni, consulta Come i servizi utilizzano KMS AWS. AWS

Parameters

Quando crei una AWS IoT regola con questa azione, devi specificare le seguenti informazioni:

`databaseName`

Il nome di un database Amazon Timestream con la tabella per ricevere i record creati dall'azione. Consulta anche **`tableName`**.

Supporta modelli sostitutivi: solo API AWS CLI

`dimensions`

Attributi dei metadati delle serie temporali scritti in ogni record di misura. Ad esempio, il nome e la zona di disponibilità di un'istanza EC2 o il nome del produttore di una turbina eolica sono dimensioni.

`name`

Il nome della dimensione dei metadati. Questo è il nome della colonna nel record della tabella del database.

Le dimensioni non possono essere denominate: `measure_name`, `measure_value` oppure `time`. Questi nomi sono riservati. I nomi delle dimensioni non possono iniziare con `ts_` o `measure_value` e non possono contenere i due punti (`:`).

Supporta modelli di sostituzione: no

`value`

Il valore da scrivere in questa colonna del record di database.

Supporta modelli di sostituzione: sì

`roleArn`

L'Amazon Resource Name (ARN) del ruolo che concede l'autorizzazione AWS IoT per scrivere nella tabella del database Timestream. Per ulteriori informazioni, consulta Requisiti.

Supporta modelli di sostituzione: no

`tableName`

Il nome della tabella del database in cui scrivere i registri delle misure. Consulta anche **`databaseName`**.

Supporta modelli sostitutivi: API e solo AWS CLI

`timestamp`

Il valore da utilizzare per il timestamp della voce. Se vuoto, viene utilizzata l'ora in cui la voce è stata elaborata.

`unit`

La precisione del valore timestamp risultante dall'espressione descritta in `value`.

Valori validi: `SECONDS` | `MILLISECONDS` | `MICROSECONDS` | `NANOSECONDS`. Il valore predefinito è `MILLISECONDS`.

`value`

Un'espressione che restituisce un lungo valore temporale epoch.

Puoi utilizzare la funzione the section called "time_to_epoch(String, String)" per creare un timestamp valido da un valore di data o ora passato nel payload del messaggio.

Contenuto del record Timestream

I dati scritti nella tabella Amazon Timestream da questa operazione includono un timestamp, i metadati dell'operazione della regola Timestream e il risultato dell'istruzione query della regola.

Per ogni attributo (misura) nel risultato dell'istruzione query, questa operazione della regola scrive un record nella tabella TimeStream specificata con queste colonne.

Nome della colonna	Tipo di attributo	Valore	Commenti
`dimension-name`	DIMENSIONE	Il valore specificato nella voce di operazione della regola Timestream.	Ciascuna Dimension e specificata nella voce di operazione della regola crea una colonna nel database Timestream con il nome della dimension e.
measure_name	MEASURE_NAME	Il nome dell'attributo	Nome dell'attributo nel risultato dell'istruzione query il cui valore

Nome della colonna	Tipo di attributo	Valore	Commenti
			è specificato nella colonna `measure_value::` *`data-type`* .
valore_misura:: *`data-type`*	MISURA_VALORE	Il valore dell'attributo nel risultato dell'istruzione di query. Il nome dell'attributo si trova nella colonna `measure_name` .	Il valore viene interpretato* e lanciato come la migliore corrispondenza di: `bigint`, `boolean`, `double` oppure `varchar`. Amazon Timestream crea una colonna separata per ogni tipo di dati. Il valore nel messaggio può essere sottoposto a trasferimento su un altro tipo di dati utilizzando la funzione `cast()` nell'istruzione query della regola.
time	TIMESTAMP	La data e l'ora della registrazione nel database.	Questo valore viene assegnato dal motore delle regole o dalla proprietà `timestamp` , se è definita.

* Il valore dell'attributo, letto dal payload del messaggio, viene interpretato come segue. Consulta the section called “Esempi” per un'illustrazione di ciascuno di questi casi.

- Un valore non quotato di `true` o `false` viene interpretato come `boolean`.
- Un numerico decimale viene interpretato come `double`.

- Un valore numerico senza virgola decimale viene interpretato come `bigint`.
- Una stringa tra virgolette viene interpretata come `varchar`.
- Gli oggetti e i valori dell'array vengono convertiti in stringhe JSON e memorizzati come `varchar`.

Esempi

Il seguente esempio JSON definisce un'azione della regola Timestream con un modello di sostituzione in una regola. AWS IoT

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'iot/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "timestream": {
          "roleArn": "arn:aws:iam::123456789012:role/aws_iot_timestream",
          "tableName": "devices_metrics",
          "dimensions": [
            {
              "name": "device_id",
              "value": "${clientId()}"
            },
            {
              "name": "device_firmware_sku",
              "value": "My Static Metadata"
            }
          ],
          "databaseName": "record_devices"
        }
      }
    ]
  }
}
```

Se utilizzi l'operazione della regola argomento del timestream definita nell'esempio precedente con il payload del messaggio seguente, si ottengono i record di Amazon Timestream scritti nella tabella che segue.

```
{
```

```
  "boolean_value": true,
  "integer_value": 123456789012,
  "double_value": 123.456789012,
  "string_value": "String value",
  "boolean_value_as_string": "true",
  "integer_value_as_string": "123456789012",
  "double_value_as_string": "123.456789012",
  "array_of_integers": [23,36,56,72],
  "array of strings": ["red", "green","blue"],
  "complex_value": {
    "simple_element": 42,
    "array_of_integers": [23,36,56,72],
    "array of strings": ["red", "green","blue"]
  }
}
```

Nella tabella seguente vengono visualizzate le colonne del database e i record creati utilizzando l'operazione della regola dell'argomento specificata per elaborare il payload del messaggio precedente. Le colonne `device_firmware_sku` e `device_id` sono le DIMENSIONI definite nell'operazione della regola dell'argomento. L'azione della regola dell'argomento Timestream crea la colonna `time`, `measure_name` e `measure_value::*`, che riempie con i valori del risultato dell'istruzione query dell'operazione della regola dell'argomento.

device_firmware_sku	device_id	measure_name	measure_value::bigint	measure_value::varchar	measure_value::double	measure_value::boolean	time
I miei metadati statici	EXAMPLE7 8iotconsole-159 -0	complex_value	-	{"simple_element":42,"array_of_integers":[23,36,56,72],"array of strings":["red","green","blue"]}	-	-	2020-08-26 22:42:16.423000000

device_firmware_sku	device_id	measure_name	measure_value::bigint	measure_value::varchar	measure_value::double	measure_value::boolean	time
I miei metadati statici	iotconsole-159 -0 EXAMPLE78	integer_value_as_string	-	123456789012	-	-	2020-08-26 22:42:16.423000000
I miei metadati statici	iotconsole-159 -0 EXAMPLE78	boolean_value	-	-	-	TRUE	2020-08-26 22:42:16.423000000
I miei metadati statici	iotconsole-159 -0 EXAMPLE78	integer_value	123456789012	-	-	-	2020-08-26 22:42:16.423000000
I miei metadati statici	iotconsole-159 -0 EXAMPLE78	string_value	-	Valore di stringa	-	-	2020-08-26 22:42:16.423000000
I miei metadati statici	iotconsole-159 -0 EXAMPLE78	array_of_integers	-	[23,36,56,72]	-	-	2020-08-26 22:42:16.423000000
I miei metadati statici	iotconsole-159 -0 EXAMPLE78	matrice di stringhe	-	["red","green","blue"]	-	-	2020-08-26 22:42:16.423000000
I miei metadati statici	iotconsole-159 -0 EXAMPLE78	boolean_value_as_string	-	TRUE	-	-	2020-08-26 22:42:16.423000000

device_firmware_sku	device_id	measure_name	measure_value::bigint	measure_value::varchar	measure_value::double	measure_value::boolean	time
I miei metadati statici	iotconsole-159 -0 EXAMPLE78	double_value	-	-	123,456789012	-	2020-08-26 22:42:16.423000000
I miei metadati statici	console iot-159 -0 EXAMPLE78	double_value_as_string	-	123,45679	-	-	2020-08-26 22:42:16.423000000

Risoluzione dei problemi relativi a una regola

Se hai un problema con le tue regole, ti consigliamo di attivare i log. CloudWatch Analizzando i registri, è possibile determinare se il problema riguarda l'autorizzazione o, ad esempio, una condizione della clausola DOVE non corrispondente. Per ulteriori informazioni, consulta Configurazione dei CloudWatch registri.

Accesso alle risorse di più account utilizzando le regole AWS IoT

È possibile configurare AWS IoT le regole per l'accesso tra account in modo che i dati importati sugli argomenti MQTT di un account possano essere instradati verso i AWS servizi, come Amazon SQS e Lambda, di un altro account. Di seguito viene spiegato come impostare AWS IoT le regole per l'acquisizione di dati tra account diversi, da un argomento MQTT in un account a una destinazione in un altro account.

Le regole cross-account possono essere configurate utilizzando Autorizzazioni basate su risorse sulla risorsa di destinazione. Pertanto, solo le destinazioni che supportano le autorizzazioni basate sulle risorse possono essere abilitate per l'accesso tramite regole tra account. AWS IoT Le destinazioni supportate includono Amazon SQS, Amazon SNS, Amazon S3 e AWS Lambda.

> **Note**
>
> Per le destinazioni supportate, ad eccezione di Amazon SQS, è necessario definire la regola nella stessa Regione AWS risorsa di un altro servizio in modo che l'azione della regola possa interagire con quella risorsa. Per ulteriori informazioni sulle azioni delle AWS IoT regole, consulta le azioni delle AWS IoT regole. Per ulteriori informazioni sull'azione SQS della regola, vedere???.

Prerequisiti

- Familiarità con Regole AWS IoT
- Una comprensione di utenti IAM, Ruoli e Autorizzazione basata su risorsa
- Avendo AWS CLI installato

Configurazione tra account per Amazon SQS

Scenario: l'account A invia i dati da un messaggio MQTT a una coda Amazon SQS dell'account B.

Account AWS	Account denominato	Description
1111-1111-1111	Account A	Operazione delle regole: `sqs:SendMessage`
2222-2222-2222	Account B	Coda Amazon SQS • ARN: *arn:aws:sqs:region:2222-2222-2222:ExampleQueue* • URL: *https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue*

> **Note**
>
> La coda Amazon SQS di destinazione non deve necessariamente corrispondere alla regola Regione AWS .AWS IoT Per ulteriori informazioni sull'azione SQS della regola, consulta. ???

Eseguire le attività dell'Account A

> **Nota**
>
> Per eseguire i seguenti comandi, l'utente IAM deve disporre delle autorizzazioni per `iot:CreateTopicRule` con l'Amazon Resource Name (ARN) della regola e le autorizzazioni a un'operazione `iam:PassRole` con una risorsa come l'ARN del ruolo.

1. Configura AWS CLI utilizzando l'account IAM dell'utente A.
2. Crea un ruolo IAM che si fidi del motore AWS IoT delle regole e alleghi una policy che consenta l'accesso alla coda Amazon SQS dell'account B. Vedi esempi di comandi e documenti relativi alle policy in AWS IoT Garantire l'accesso richiesto.
3. Per creare una regola allegata a un argomento, esegui il create-topic-rule comando.

   ```
   aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-rule.json
   ```

   Di seguito è riportato un esempio di file di payload con una regola che inserisce tutti i messaggi inviati all'argomento `iot/test` nella coda Amazon SQS specificata. L'istruzione SQL filtra i messaggi e l'ARN del ruolo concede ad AWS IoT l'autorizzazione per aggiungere il messaggio alla cosa di Amazon SQS.

   ```
   {
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
     {
      "sqs": {
       "queueUrl": "https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue",
       "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role",
   ```

```
    "useBase64": false
   }
  }
 ]
}
```

Per ulteriori informazioni su come definire un'azione Amazon SQS in una AWS IoT regola, consulta AWS IoT rule actions - Amazon SQS.

Esegui le attività dell'Account B

1. Configura AWS CLI utilizzando l'account IAM dell'utente B.
2. Per concedere le autorizzazioni per la risorsa della coda Amazon SQS per l'account A, esegui il comando add-permission.

```
aws sqs add-permission --queue-url https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue --label SendMessagesToMyQueue --aws-account-ids 1111-1111-1111 --actions SendMessage
```

Configurazione tra account per Amazon SNS

Scenario: l'account A invia i dati da un messaggio MQTT a un argomento Amazon SNS dell'account B.

Account AWS	Account denominato	Description
1111-1111-1111	Account A	Operazione delle regole: `sns:Publish`
2222-2222-2222	Account B	ARN Argomento Amazon SNS: *arn:aws:sns:region:2222-2222-2222:ExampleTopic*

Esegui le attività dell'Account A

> **Note**
>
> Per eseguire i seguenti comandi, l'utente IAM deve disporre delle autorizzazioni per `iot:CreateTopicRule` con la regola ARN come risorsa e le autorizzazioni per le operazioni `iam:PassRole` con una risorsa come ruolo ARN.

1. Configura AWS CLI utilizzando l'account IAM dell'utente A.
2. Crea un ruolo IAM che si fidi del motore AWS IoT delle regole e alleghi una policy che consenta l'accesso all'argomento Amazon SNS dell'account B. Ad esempio, comandi e documenti relativi alle policy, consulta Garantire AWS IoT l'accesso richiesto.
3. Per creare una regola allegata a un argomento, esegui il create-topic-rule comando.

   ```
   aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-rule.json
   ```

   Di seguito è riportato un esempio di file di payload con una regola che inserisce tutti i messaggi inviati all'argomento `iot/test` nell'argomento Amazon SNS specificato. L'istruzione SQL filtra i messaggi e l'ARN del ruolo concede ad AWS IoT l'autorizzazione per inviare il messaggio all'argomento Amazon SNS.

   ```
   {
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
     {
      "sns": {
       "targetArn": "arn:aws:sns:region:2222-2222-2222:ExampleTopic",
       "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
      }
     }
    ]
   }
   ```

   Per ulteriori informazioni su come definire un'azione Amazon SNS in una AWS IoT regola, consulta AWS IoT rule actions - Amazon SNS.

Esegui le attività dell'Account B

1. Configura AWS CLI utilizzando l'account IAM dell'utente B.
2. Per concedere l'autorizzazione alla risorsa dell'argomento Amazon SNS all'account A, esegui il comando add-permission.

```
aws sns add-permission --topic-arn arn:aws:sns:region:2222-2222-2222:ExampleTopic
 --label Publish-Permission --aws-account-id 1111-1111-1111 --action-name Publish
```

Configurazione tra account per Amazon S3

Scenario: l'account A invia i dati da un messaggio MQTT a un bucket Amazon S3 dell'account B.

Account AWS	Account denominato	Description
1111-1111-1111	Account A	Operazione delle regole: `s3:PutObject`
2222-2222-2222	Account B	ARN bucket Amazon S3: *arn:aws:s3:::amzn-s3-demo-bucket*

Eseguire le attività dell'Account A

> **Nota**
>
> Per eseguire i seguenti comandi, l'utente IAM deve disporre delle autorizzazioni per `iot:CreateTopicRule` con la regola ARN come risorsa e le autorizzazioni per l'operazione `iam:PassRole` con una risorsa come ruolo ARN.

1. Configura AWS CLI utilizzando l'account IAM dell'utente A.
2. Crea un ruolo IAM che si fidi del motore AWS IoT delle regole e alleghi una policy che consenta l'accesso al bucket Amazon S3 dell'account B. Ad esempio, comandi e documenti relativi alle policy, consulta AWS IoT Garantire l'accesso richiesto.
3. Per creare una regola da allegare al bucket S3 di destinazione, esegui il comando. create-topic-rule

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
```

Di seguito è riportato un esempio di file di payload con una regola che inserisce tutti i messaggi inviati all'argomento `iot/test` nel bucket Amazon S3 specificato. L'istruzione SQL filtra i messaggi e l'ARN del ruolo concede ad AWS IoT l'autorizzazione per aggiungere il messaggio nel bucket Amazon S3.

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "s3": {
    "bucketName": "amzn-s3-demo-bucket",
    "key": "${topic()}/${timestamp()}",
    "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
   }
  }
 ]
}
```

Per ulteriori informazioni su come definire un'azione Amazon S3 in una AWS IoT regola, consulta AWS IoT rule actions - Amazon S3.

Esegui le attività dell'Account B

1. Configura AWS CLI utilizzando l’account IAM dell'utente B.
2. Crea una policy bucket che consideri attendibile l'entità dell'account A principale.

   Di seguito è riportato un esempio di file di payload che definisce una policy di bucket che considera attendibile l'entità di un altro account.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AddCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:root"
                ]
        },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Per ulteriori informazioni, consulta Esempi di policy bucket.

3. Per allegare la policy del bucket al bucket specificato, esegui il comando. put-bucket-policy

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://./amzn-s3-demo-bucket-policy.json
```

4. Per far sì che l'accesso tra account funzioni, assicurati di avere l'impostazione corretta Blocca tutti gli accessi pubblici. Per ulteriori informazioni, consulta Best practice relative alla sicurezza per Amazon S3.

Configurazione tra più account per AWS Lambda

Scenario: l'account A richiama una AWS Lambda funzione dell'account B, trasmettendo un messaggio MQTT.

Account AWS	Account denominato	Description
1111-1111-1111	Account A	Operazione delle regole: `lambda:InvokeFunction`
2222-2222-2222	Account B	L'ARN della funzione Lambda: *arn:aws:lambda:region:2222-2222-2222:function:example-function*

Eseguire le attività dell'Account A

> **Note**
>
> Per eseguire i seguenti comandi, l'utente IAM deve disporre delle autorizzazioni a `iot:CreateTopicRule` con la regola ARN come risorsa e le autorizzazioni per l'operazione `iam:PassRole` con risorsa come ruolo ARN.

1. Configura AWS CLI utilizzando l'account IAM dell'utente A.
2. Esegui il create-topic-rule comando per creare una regola che definisca l'accesso tra account alla funzione Lambda dell'account B.

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
```

Di seguito è riportato un esempio di file di payload con una regola che inserisce tutti i messaggi inviati all'argomento `iot/test` nella funzione Lambda specificata. L'istruzione SQL filtra i messaggi e l'ARN del ruolo concede ad AWS IoT l'autorizzazione per il passaggio dei dati alla funzione Lambda.

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "lambda": {
    "functionArn": "arn:aws:lambda:region:2222-2222-2222:function:example-function"
   }
  }
 ]
}
```

Per ulteriori informazioni su come definire un' AWS Lambda azione in una AWS IoT regola, leggi AWS IoT rule actions - Lambda.

Esegui le attività dell'Account B

1. Configura AWS CLI utilizzando l'account IAM dell'utente B.
2. Esegui il comando add-permission di Lambda per autorizzare AWS IoT le regole ad attivare la funzione Lambda. Per eseguire il seguente comando, l'utente IAM dovrebbe avere l'autorizzazione per l'operazione `lambda:AddPermission`.

```
aws lambda add-permission --function-name example-function --region us-east-1 --
principal iot.amazonaws.com --source-arn arn:aws:iot:region:1111-1111-1111:rule/
example-rule --source-account 1111-1111-1111 --statement-id "unique_id" --action
 "lambda:InvokeFunction"
```

Opzioni:

--principal

Questo campo consente a AWS IoT (rappresentato da`iot.amazonaws.com`) di chiamare la funzione Lambda.

--source-arn

Questo campo conferma che solo `arn:aws:iot:region:1111-1111-1111:rule/example-rule` in AWS IoT attiva questa funzione Lambda e che nessun'altra regola nello stesso o in un altro account può attivare questa funzione Lambda.

--source-account

Questo campo conferma che AWS IoT attiva questa funzione Lambda solo per conto dell'`1111-1111-1111`account.

> **Note**
>
> Se viene visualizzato un messaggio di errore "Impossibile trovare la regola" nella console della funzione AWS Lambda sotto Configurazione, ignora il messaggio di errore e procedi al test della connessione.

Gestione degli errori (operazione in caso di errore)

Quando AWS IoT riceve un messaggio da un dispositivo, il motore delle regole verifica se il messaggio corrisponde a una regola. In tal caso, l'istruzione query della regola viene valutata e vengono attivate le operazioni della regola, passando il risultato dell'istruzione query.

Se si verifica un problema quando si richiama un'operazione, il motore di regole richiama un'operazione da eseguire in caso di errore, se ne è stata specificata una per la regola. Questo può accadere quando:

- Una regola non dispone dell'autorizzazione per l'accesso a un bucket Amazon S3.
- Un errore dell'utente causa il superamento della velocità effettiva di Dynamo DB di cui è stato effettuato il provisioning.

> **Note**
>
> La gestione degli errori trattata in questo argomento riguarda le operazioni delle regole. Per eseguire il debug dei problemi SQL, incluse le funzioni esterne, puoi configurare la AWS IoT registrazione. Per ulteriori informazioni, consulta ???.

Formato del messaggio dell'operazione da eseguire in caso di errore

Viene generato un singolo messaggio per ogni regola e messaggio. Se, ad esempio, si verifica un errore per due operazioni nella stessa regola, l'operazione da eseguire in caso di errore riceve un messaggio contenente entrambi gli errori.

Il messaggio dell'operazione di errore è simile a quello nell'esempio seguente.

```
{
  "ruleName": "TestAction",
  "topic": "testme/action",
  "cloudwatchTraceId": "7e146a2c-95b5-6caf-98b9-50e3969734c7",
  "clientId": "iotconsole-1511213971966-0",
  "base64OriginalPayload":
 "ewogICJtZXNzYWdlIjogIkhlbGxvIHZyb20gQVdTIElvVCBjb25zb2xlIgp9",
  "failures": [
    {
```

```
      "failedAction": "S3Action",
      "failedResource": "us-east-1-s3-verify-user",
      "errorMessage": "Failed to put S3 object. The error received was The
 specified bucket does not exist (Service: Amazon S3; Status Code: 404; Error
 Code: NoSuchBucket; Request ID: 9DF5416B9B47B9AF; S3 Extended Request ID:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y=).
 Message arrived on: error/action, Action: s3, Bucket: us-
east-1-s3-verify-user, Key: \"aaa\". Value of x-amz-id-2:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y="
    }
  ]
}
```

ruleName

Nome della regola che ha attivato l'operazione da eseguire in caso di errore.

topic

Argomento dove è stato ricevuto il messaggio originale.

cloudwatchTraceId

Un'identità univoca che si riferisce ai log in degli errori. CloudWatch

clientId

ID client di chi ha pubblicato il messaggio.

base64 OriginalPayload

Payload del messaggio originale con codifica Base64.

failures

failedAction

Nome dell'operazione che non è stato possibile completare, ad esempio "S3Action".

failedResource

Nome della risorsa, ad esempio il nome di un bucket S3.

errorMessage

Descrizione e spiegazione dell'errore.

Esempio di operazione in caso di errore

Di seguito è illustrato un esempio di una regola a cui è stata aggiunta un'operazione da eseguire in caso di errore. La regola seguente include un'operazione di scrittura dei dati dei messaggi in una tabella Dynamo DB e un'operazione da eseguire in caso di errore di scrittura dei dati in un bucket Amazon S3:

```
{
    "sql" : "SELECT * FROM ..."
    "actions" : [{
        "dynamoDB" : {
            "table" : "PoorlyConfiguredTable",
            "hashKeyField" : "AConstantString",
            "hashKeyValue" : "AHashKey"}}
    ],
    "errorAction" : {
        "s3" : {
            "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
            "bucketName" : "message-processing-errors",
            "key" : "${replace(topic(), '/', '-') + '-' + timestamp() + '-' +
 newuuid()}"
        }
    }
}
```

È possibile utilizzare qualsiasi funzione o modello di sostituzione nell'istruzione SQL di un'azione di errore, incluse le funzioni esterne: `aws_lambda()`,,,, `get_dynamodb()``get_registry_data()``get_thing_shadow()`, `get_secret()`e. `machinelearning_predict()``decode()` Se un'azione di errore richiede la chiamata di una funzione esterna, l'invocazione dell'azione di errore può comportare un fatturato aggiuntivo per la funzione esterna.

Per ulteriori informazioni sulle regole e su come specificare un'azione di errore, vedere Creazione di una AWS IoT regola.

Per ulteriori informazioni sull'utilizzo CloudWatch per monitorare l'esito positivo o negativo delle regole, vedereAWS IoT metriche e dimensioni.

Riduzione dei costi di messaggistica con Basic Ingest

Puoi utilizzare Basic Ingest per inviare in modo sicuro i dati del dispositivo al Servizi AWS supporto daAWS IoT azioni relative alle regole, senza incorrere in costi di messaggistica. La funzionalità Basic Ingest ottimizza il flusso di dati rimuovendo il broker di messaggistica publish/subscribe dal percorso di inserimento.

Basic Ingest ti permette di inviare messaggi dai tuoi dispositivi o applicazioni. I messaggi hanno nomi di argomenti che iniziano con `$aws/rules/`*`rule_name`* per i primi tre livelli, dove *`rule_name`* è il nome della regola AWS IoT che desideri chiamare.

Con Basic Ingest puoi utilizzare una regola esistente semplicemente aggiungendo il prefisso Basic Ingest (`$aws/rules/`*`rule_name`*) all'argomento del messaggio con cui chiami la regola. Ad esempio, se hai una regola denominata `BuildingManager` che viene chiamata da messaggi con argomenti come `Buildings/Building5/Floor2/Room201/Lights` (`"sql": "SELECT * FROM 'Buildings/#'"`), puoi chiamare la stessa regola con Basic Ingest inviando un messaggio con argomento `$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights`.

> **Note**
>
> - I dispositivi e le regole non possono sottoscrivere gli argomenti Basic Ingest riservati. Ad esempio, le AWS IoT Device Defender `num-messages-received` metriche non vengono emesse in quanto non supportano la sottoscrizione agli argomenti. Per ulteriori informazioni, consulta Argomenti riservati.
> - Se hai bisogno di un broker di pubblicazione/sottoscrizione per distribuire messaggi a più abbonati (ad esempio, per recapitare messaggi ad altri dispositivi e al motore delle regole), dovresti continuare a utilizzare il broker di messaggi per gestire la AWS IoT distribuzione dei messaggi. Pertanto, ti consigliamo di pubblicare i messaggi su argomenti diversi da quelli Basic Ingest.

Utilizzo di Basic Ingest

Prima di utilizzare Basic Ingest, verifica che il tuo dispositivo o la tua applicazione utilizzi una policy con autorizzazioni di pubblicazione su `$aws/rules/*`. In alternativa, puoi specificare l'autorizzazione per le singole regole nella politica. `$aws/rules/`*`rule_name`*`/*` In caso contrario,

i dispositivi e le applicazioni possono continuare a utilizzare le connessioni esistenti con AWS IoT Core.

Quando il messaggio raggiunge il motore di regole, non sussistono differenze di implementazione o gestione dell'errore tra le regole chiamate da Basic Ingest e quelle chiamate tramite sottoscrizioni al broker di messaggi.

Puoi anche creare regole da utilizzare con Basic Ingest. Ricorda:

- Il prefisso iniziale di un argomento Basic Ingest (`$aws/rules/`*`rule_name`*) non è disponibile per la funzione topic(Decimal).
- Se definisci una regola che viene chiamata solo con Basic Ingest, la clausola `FROM` è facoltativa nel campo `sql` della definizione `rule`. È comunque necessaria se la regola viene chiamata anche da altri messaggi che devono essere inviati tramite il broker di messaggi, ad esempio perché gli altri messaggi devono essere distribuiti a più sottoscrittori. Per ulteriori informazioni, consulta AWS IoT Riferimento SQL.
- I primi tre livelli di argomento Basic Ingest (`$aws/rules/`*`rule_name`*) non vengono conteggiati ai fini del limite di lunghezza di 8 segmenti o del limite totale di 256 caratteri per un argomento. In caso contrario, si applicano le stesse restrizioni, come documentato in Limiti per AWS IoT.
- Se viene ricevuto un messaggio con un argomento Basic Ingest che specifica una regola inattiva o una regola che non esiste, viene creato un log degli errori in un CloudWatch log Amazon per aiutarti con il debug. Per ulteriori informazioni, consulta Voci di registro del motore delle regole. È indicato un parametro `RuleNotFound` ed è possibile creare allarmi su questo parametro. Per ulteriori informazioni, consulta Parametri delle regole in Parametri delle regole.
- È ancora possibile pubblicare con QoS 1 su argomenti Basic Ingest. Riceverai un messaggio PUBACK dopo che il messaggio è stato inviato correttamente al motore delle regole. La ricezione di un PUBACK messaggio non significa che le azioni relative alle regole siano state completate correttamente. Puoi configurare un'operazione di errore per gestire gli errori durante l'esecuzione dell'operazione. Per ulteriori informazioni, consulta Gestione degli errori (operazione in caso di errore).

AWS IoT Riferimento SQL

Nel AWS IoT, le regole vengono definite utilizzando una sintassi simile a SQL. Le istruzioni SQL sono costituite da tre tipi di clausole:

SET

(Facoltativo) Definisce le variabili che è possibile riutilizzare nelle istruzioni SQL e nei modelli di sostituzione. Assegna valori alle variabili utilizzando espressioni. Fai riferimento a queste variabili nelle clausole SELECT e WHERE e nei modelli di sostituzione delle azioni.

La clausola SET supportaTipi di dati,,,, OperatoriFunzioniValori letteraliIstruzioni case, Estensioni JSON Variables e. Query di oggetti nidificati

SELECT

(Obbligatorio) Estrae le informazioni dal payload del messaggio in ingresso ed esegue le trasformazioni sulle informazioni. I messaggi da utilizzare sono identificati dal Filtro di argomenti specificato nella clausola FROM.

La clausola SELECT supporta Tipi di datiOperatori,,Funzioni,Valori letterali,Istruzioni case, Estensioni JSON Modelli di sostituzioneQuery di oggetti nidificati, Variables e. Payload binari

FROM

Il messaggio MQTT Filtro di argomenti che identifica i messaggi da cui estrarre i dati. La regola viene attivata per ogni messaggio inviato a un argomento MQTT che corrisponde al filtro argomento specificato qui. Obbligatorio per le regole che vengono attivate da messaggi che attraversano il broker di messaggi. Facoltativo per le regole che vengono attivate solo utilizzando la caratteristica Basic Ingest.

WHERE

(Facoltativo) Aggiunge una logica condizionale che determina se le operazioni specificate da una regola vengono eseguite.

La clausola WHERE supportaTipi di dati,Operatori,Funzioni,Valori letterali, Istruzioni caseEstensioni JSON, Variables e. Query di oggetti nidificati

Un'istruzione SQL di esempio è simile a quanto segue:

```
SELECT color AS rgb FROM 'topic/subtopic' WHERE temperature > 50
```

Un messaggio MQTT di esempio (detto anche payload in ingresso) è simile a quanto segue:

```
{
    "color":"red",
```

```
    "temperature":100
}
```

Se questo messaggio viene pubblicato nell'argomento `'topic/subtopic'`, la regola viene attivata e l'istruzione SQL viene valutata. L'istruzione SQL estrae il valore della proprietà `color` se la proprietà `"temperature"` è superiore a 50. La clausola WHERE specifica la condizione `temperature > 50`. La parola chiave AS rinomina la proprietà `"color"` in `"rgb"`. Il risultato (detto anche payload in uscita) è simile al seguente:

```
{
    "rgb":"red"
}
```

Questi dati vengono quindi inoltrati all'operazione della regola, che invia i dati per l'ulteriore elaborazione. Per ulteriori informazioni sulle operazioni delle regole, consulta AWS IoT azioni relative alle regole.

> **Note**
>
> I commenti non sono attualmente supportati nella sintassi AWS IoT SQL.
> I nomi degli attributi che contengono spazi non possono essere utilizzati come nomi di campo nell'istruzione SQL. Il payload in entrata può avere nomi di attributi contenenti spazi, ma tali nomi non possono essere utilizzati nell'istruzione SQL. Tuttavia, vengono passati al payload in uscita se utilizzi una specifica jolly (*) per il nome del campo.

Clausola SELECT

La clausola AWS IoT SELECT è essenzialmente la stessa della clausola ANSI SQL SELECT, con alcune differenze minori.

La clausola SELECT supportaTipi di dati,,Operatori,Funzioni, Valori letterali Istruzioni caseEstensioni JSON, Variables e. Query di oggetti nidificati Payload binari

È possibile usare la clausola SELECT per estrarre informazioni dai messaggi MQTT in ingresso. È inoltre possibile utilizzare `SELECT *` per recuperare l'intero payload del messaggio in arrivo. Ad esempio:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
```

```
SQL statement: SELECT * FROM 'topic/subtopic'
Outgoing payload: {"color":"red", "temperature":50}
```

Se il payload è un oggetto JSON, è possibile fare riferimento alle chiavi nell'oggetto. Il payload in uscita contiene la coppia chiave-valore. Ad esempio:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL statement: SELECT color FROM 'topic/subtopic'
Outgoing payload: {"color":"red"}
```

È possibile usare la parola chiave AS per rinominare le chiavi. Ad esempio:

```
Incoming payload published on topic 'topic/subtopic':{"color":"red", "temperature":50}
SQL:SELECT color AS my_color FROM 'topic/subtopic'
Outgoing payload: {"my_color":"red"}
```

È possibile selezionare più elementi separandoli con una virgola. Ad esempio:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT color as my_color, temperature as fahrenheit FROM 'topic/subtopic'
Outgoing payload: {"my_color":"red","fahrenheit":50}
```

È possibile selezionare più elementi includendo "*" per aggiungere gli elementi al payload in ingresso. Ad esempio:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT *, 15 as speed FROM 'topic/subtopic'
Outgoing payload: {"color":"red", "temperature":50, "speed":15}
```

È possibile usare la parola chiave `"VALUE"` per produrre payload in uscita che non sono oggetti JSON. Con SQL versione `2015-10-08`, è possibile selezionare solo un elemento. Con SQL versione `2016-03-23` o successiva, è anche possibile selezionare una matrice da produrre come oggetto di primo livello.

Example

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT VALUE color FROM 'topic/subtopic'
Outgoing payload: "red"
```

È possibile usare la sintassi `'.'` per analizzare in maggiore dettaglio gli oggetti JSON nidificati nel payload in ingresso. Ad esempio:

```
Incoming payload published on topic 'topic/subtopic': {"color":
{"red":255,"green":0,"blue":0}, "temperature":50}
SQL: SELECT color.red as red_value FROM 'topic/subtopic'
Outgoing payload: {"red_value":255}
```

Per informazioni su come utilizzare i nomi degli oggetti e delle proprietà JSON che includono caratteri riservati, ad esempio i numeri o il trattino (meno), consulta Estensioni JSON

È possibile usare le funzioni (consulta Funzioni) per trasformare il payload in ingresso. È possibile utilizzare le parentesi per il raggruppamento. Ad esempio:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT (temperature - 32) * 5 / 9 AS celsius, upper(color) as my_color FROM
 'topic/subtopic'
Outgoing payload: {"celsius":10,"my_color":"RED"}
```

Clausola FROM

La clausola FROM esegue la sottoscrizione della regola in un argomento or filtro argomenti. L'argomento o il filtro dell'argomento deve essere racchiuso tra virgolette singole ('). La regola viene attivata per ogni messaggio inviato a un argomento MQTT che corrisponde al filtro argomento specificato qui. Esegui la sottoscrizione di un gruppo di argomenti simili utilizzando un filtro di argomenti.

Esempio:

Payload in ingresso pubblicato nell'argomento `'topic/subtopic'`: `{temperature: 50}`

Payload in ingresso pubblicato nell'argomento `'topic/subtopic-2'`: `{temperature: 50}`

SQL: `"SELECT temperature AS t FROM 'topic/subtopic'"`.

La regola è sottoscritta in `'topic/subtopic'`, quindi il payload in ingresso viene passato alla regola. Il payload in uscita, passato alle operazioni della regola, è: `{t: 50}`. La regola non viene sottoscritta in `'topic/subtopic-2'`, quindi la regola non viene attivata per il messaggio pubblicato in `'topic/subtopic-2'`.

Esempio di carattere jolly #:

Puoi utilizzare il carattere jolly '#' (multi-livello) per la corrispondenza con uno o più elementi del percorso particolari:

Payload in ingresso pubblicato nell'argomento `'topic/subtopic'`: `{temperature: 50}`.

Payload in ingresso pubblicato nell'argomento `'topic/subtopic-2'`: `{temperature: 60}`.

Payload in ingresso pubblicato nell'argomento `'topic/subtopic-3/details'`: `{temperature: 70}`.

Payload in ingresso pubblicato nell'argomento `'topic-2/subtopic-x'`: `{temperature: 80}`.

SQL: `"SELECT temperature AS t FROM 'topic/#'"`.

La regola è sottoscritta a qualsiasi argomento che inizia con `'topic'`, quindi viene eseguita tre volte, inviando payload in uscita di (per argomento/sottoargomento), `{t: 50}` (per argomento/subtopic-2) e `{t: 60}` (per) alle relative azioni. `{t: 70}` topic/subtopic-3/details La regola non viene sottoscritta in `'topic-2/subtopic-x'`, pertanto non viene attivata per il messaggio `{temperature: 80}`.

Esempio di carattere jolly +:

Puoi utilizzare il carattere jolly '+' (livello singolo) per la corrispondenza con qualsiasi elemento del percorso particolare:

Payload in ingresso pubblicato nell'argomento `'topic/subtopic'`: `{temperature: 50}`.

Payload in ingresso pubblicato nell'argomento `'topic/subtopic-2'`: `{temperature: 60}`.

Payload in ingresso pubblicato nell'argomento `'topic/subtopic-3/details'`: `{temperature: 70}`.

Payload in ingresso pubblicato nell'argomento `'topic-2/subtopic-x'`: `{temperature: 80}`.

SQL: `"SELECT temperature AS t FROM 'topic/+'"`.

La regola viene sottoscritta in tutti gli argomenti con due elementi di percorso, dove il primo elemento è `'topic'`. La regola viene eseguita per i messaggi inviati a `'topic/subtopic'` e `'topic/subtopic-2'`, ma non a `'topic/subtopic-3/details'` (ha più livelli rispetto al filtro argomento) o a `'topic-2/subtopic-x'` (non inizia con `topic`).

Clausola SET

Utilizzate la clausola SET per definire le variabili che memorizzano i risultati delle espressioni. È possibile riutilizzare queste variabili nelle clausole SELECT e WHERE e nei modelli sostitutivi. Ciò consente di evitare la duplicazione di espressioni complesse e di ridurre il numero di chiamate di funzione nell'istruzione SQL.

La clausola SET supportaTipi di dati,,Operatori,Funzioni, Valori letterali Istruzioni caseEstensioni JSON, Variabili e. Query di oggetti nidificati

Sintassi della clausola SET

La clausola SET deve comparire prima della clausola SELECT nell'istruzione SQL. Utilizzare la seguente sintassi:

```
SET @variable_name = expression [, @variable_name2 = expression2]
```

Regole di sintassi:

- Inizia i nomi delle variabili con @
- I nomi delle variabili possono contenere lettere, numeri e caratteri di sottolineatura
- I nomi delle variabili possono avere una lunghezza massima di 64 caratteri
- È possibile impostare più variabili in una singola clausola SET, separate da virgole
- Ogni variabile può essere assegnata una sola volta (le variabili sono immutabili)
- La parola chiave SET può essere utilizzata solo una volta per istruzione SQL

Utilizzo di variabili

Dopo aver definito le variabili, puoi usarle in:

- Clausole SELECT
- Clausole WHERE
- Altre assegnazioni di variabili SET
- Modelli di sostituzione delle azioni
- Modelli di sostituzione delle azioni di errore
- Interrogazioni SELECT annidate

- Parametri delle funzioni (alcuni parametri come i parametri ROLearn e i parametri che modificano la modalità di una funzione in modo simile a `transform("enrichArray", attributes, values)` non supportare le variabili)

Le variabili vengono referenziate utilizzando la stessa `@variable_name` sintassi utilizzata nella clausola SET. È inoltre possibile utilizzare la sintassi dell'estensione JSON per accedere alle proprietà delle variabili che contengono oggetti, ad esempio. `@variable_name.property`

Esempi di clausole SET

Utilizzo delle variabili di base

L'esempio seguente mostra un payload pubblicato sull'argomento`device/data`: `{"temp_fahrenheit": 75, "humidity": 60}`

Istruzione SQL:

```
SET @temp_celsius = (temp_fahrenheit - 32) * 5 / 9
SELECT @temp_celsius AS celsius, humidity FROM 'device/data'
```

Payload in uscita: `{"celsius": 23.89, "humidity": 60}`

Accedi ai membri negli oggetti JSON incorporati

L'esempio seguente mostra un payload pubblicato sull'argomento: `device/data` `{"device1": {"deviceId":"weather_sensor", "deviceData": {"sensors": {"temp_fahrenheit": 75, "humidity": 60}, "location": [47.606,-122.332]}}}`

Istruzione SQL:

```
SET @device_sensor_data = device1.deviceData.sensors
SELECT @device_sensor_data.temp_fahrenheit AS temp_fahrenheit,
 @device_sensor_data.humidity as humidity, device1.deviceId as deviceId FROM 'device/
data'
```

Payload in uscita: `{"temp_fahrenheit":75,"humidity":60,"deviceId":"weather_sensor"}`

per ulteriori informazioni su come lavorare con le estensioni JSON, consulta Estensioni JSON

Evitare chiamate di funzione duplicate

Le variabili SET aiutano a evitare la duplicazione di operazioni di decodifica complesse:

```
SET @decoded_data = decode(encode(*, 'base64'), 'proto', 'schema', 'schema.desc',
 'message.proto', 'Message')
SELECT @decoded_data.sensor_id, @decoded_data.reading FROM 'device/protobuf'
WHERE @decoded_data.reading > 100
```

Senza le variabili SET, sarebbe necessario ripetere la funzione di decodifica tre volte, il che supera i limiti di chiamata alla funzione.

Variabili multiple

È possibile definire più variabili in una singola clausola SET separandole con virgole:

```
SET @user_data = get_user_properties(device_id), @threshold = 50
SELECT @user_data.name, temp_fahrenheit FROM 'sensors/+'
WHERE temp_fahrenheit > @threshold AND @user_data.active = true
```

Utilizzo di variabili nei modelli sostitutivi

Le variabili possono essere utilizzate anche nei modelli di sostituzione delle azioni, che consentono di riutilizzare i valori calcolati sia nell'istruzione SQL che nelle azioni delle regole.

Istruzione SQL:

```
SET @temp_celsius = (temp_fahrenheit - 32) * 5 / 9
SELECT @temp_celsius AS celsius, humidity FROM 'device/data'
```

Configurazione dell'azione:

```
{
  "s3": {
    "roleArn": "arn:aws:iam::123456789012:role/testRuleRole",
    "bucketName": "bucket",
    "key": "temperature-data/${device_id}/temp-${@temp_celsius}C.json"
  }
}
```

In questo esempio, la variabile SET `@temp_celsius` viene utilizzata in un modello sostitutivo per costruire il campo chiave dell'azione S3.

Utilizzo di payload non JSON

Le variabili SET non supportano direttamente payload non JSON, quindi il payload deve essere prima codificato o decodificato:

```
SET @encoded_payload = encode(*, 'base64')
SELECT @encoded_payload AS raw_data FROM 'device/binary'
```

per ulteriori informazioni su come lavorare con payload non JSON, consulta Utilizzo di payload binari

Limiti della clausola SET

I seguenti limiti si applicano alle variabili SET:

- Massimo 10 variabili univoche per istruzione SQL
- Dimensione massima del valore variabile di 128 KiB (stringa JSON UTF-8 minimizzata)
- Dimensione massima del valore totale di 128 KB per tutte le variabili
- Nomi di variabili limitati a 64 caratteri
- Le variabili possono accettare i payload JSON direttamente così come sono (i payload non JSON devono prima essere codificati/decodificati)

Clausola WHERE

La clausola WHERE determina se le operazioni specificate da una regola vengono eseguite. Se la clausola WHERE restituisce true, le operazioni della regola vengono eseguite. In caso contrario, le operazioni della regola non vengono eseguite.

La clausola WHERE supportaTipi di dati,Operatori,Funzioni,Valori letterali, Istruzioni caseEstensioni JSON, Variables e. Query di oggetti nidificati

Esempio:

Payload in ingresso pubblicato in `topic/subtopic`: `{"color":"red", "temperature":40}`.

SQL: `SELECT color AS my_color FROM 'topic/subtopic' WHERE temperature > 50 AND color <> 'red'`.

In questo caso la regola verrebbe attivata, ma non verrebbero eseguite le operazioni specificate dalla regola. Non ci sarà alcun payload in uscita.

È possibile usare le funzioni e gli operatori nella clausola WHERE. Tuttavia, non è possibile fare riferimento agli alias creati con la parola chiave AS in SELECT. La clausola WHERE viene valutata per prima, per determinare se la clausola SELECT viene valutata.

Esempio con payload non JSON:

Payload non JSON in entrata pubblicato su `topic/subtopic`: `80`

SQL: `SELECT decode(encode(*, 'base64'), 'base64') AS value FROM 'topic/subtopic' WHERE decode(encode(*, 'base64'), 'base64') > 50`

In questo caso la regola verrebbe attivata, ma non verrebbero eseguite le operazioni specificate dalla regola. Il payload in uscita verrà trasformato dalla clausola SELECT come payload JSON `{"value":80}`.

Tipi di dati

Il motore AWS IoT delle regole supporta tutti i tipi di dati JSON.

Tipi di dati supportati

Tipo	Significato
`Int`	Un discreto `Int`. 34 cifre al massimo.
`Decimal`	Tipo `Decimal` con una precisione di 34 cifre, con grandezza minima diversa da zero corrispondente a 1E-999 e grandezza massima di 9.999…E999. **Note** Alcune funzioni restituiscono valori `Decimal` a precisione doppia invece di 34 cifre di precisione. Con SQL V2 (23/03/2016), valori numerici che sono numeri interi, come `10.0`, vengono elaborati come un valore `Int` (`10`) invece del previsto valore `Decimal` (`10.0`). Per elaborare

Tipo	Significato
	in modo affidabile valori numerici interi come i valori `Decimal`, utilizzare SQL V1 (08/10/2015) per l'istruzione di query della regola.
`Boolean`	`True` o `False`.
`String`	Stringa UTF-8.
`Array`	Serie di valori non necessariamente dello stesso tipo.
`Object`	Valore JSON costituito da una chiave e un valore. Le chiavi devono essere stringhe. I valori possono essere di qualsiasi tipo.
`Null`	`Null` come definito da JSON. Si tratta di un valore effettivo che rappresenta l'assenza di un valore. È possibile creare in modo esplicito un valore `Null` usando la parola chiave `Null` nell'istruzione SQL. Ad esempio: `"SELECT NULL AS n FROM 'topic/subtopic'"`

Tipo	Significato

`Undefined`	Non un valore. Non si tratta di in tipo rappresentabile in modo esplicito in JSON, se non omettendo il valore. Ad esempio, nell'oggetto `{"foo": null}` la chiave "foo" restituisce NULL, ma la chiave "bar" restituisce `Undefined`. Internamente, il linguaggio SQL tratta `Undefined` come un valore, ma questo tipo non è rappresentabile in JSON, quindi quando viene serializzato in JSON i risultati sono `Undefined`.
	`{"foo":null, "bar":undefined}`
	viene serializzato in JSON come:
	`{"foo":null}`
	Analogamente, `Undefined` viene convertito in una stringa vuota quando serializzato. Le funzioni chiamate con argomenti non validi (ad esempio tipi errati, numero errato di argomenti e così via) restituiscono `Undefined`.

Conversioni

La tabella seguente elenca i risultati quando un valore di un tipo viene convertito in un altro tipo (quando un valore del tipo non corretto viene fornito a una funzione). Se, ad esempio, alla funzione di valore assoluto "abs" (che richiede un tipo `Int` o `Decimal`) viene passato un tipo `String`, viene eseguito un tentativo di convertire `String` in `Decimal`, seguendo queste regole. In questo caso, 'abs("-5.123")' viene trattato come 'abs(-5.123)'.

> **Note**
>
> Non vengono eseguiti tentativi di conversione in un tipo `ArrayObject`, `Null` o `Undefined`.

Nel tipo Decimal

Tipo di argomento	Risultato
`Int`	Tipo `Decimal` senza separatore decimale.
`Decimal`	Valore di origine.
`Boolean`	`Undefined` . (È possibile usare in modo esplicito la funzione cast per trasformare true = 1.0, false = 0.0.)
`String`	Il motore SQL tenta di analizzare la stringa come una. `Decimal` AWS IoT tenta di analizzare le stringhe che corrispondono all'espressione regolare:. `^-?\d+(\.\d+)?((?i)E-?\d+)?$` "0", "-1.2", "5E-12" sono tutti esempi di stringhe che vengono convertite automaticamente in tipi `Decimal`.
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Null`.
Undefined	`Undefined` .

Nel tipo Int

Tipo di argomento	Risultato
`Int`	Valore di origine.
`Decimal`	Valore di origine arrotondato al valore `Int` più vicino.

Tipo di argomento	Risultato
`Boolean`	`Undefined` . (È possibile usare in modo esplicito la funzione cast per trasformare true = 1.0, false = 0.0.)
`String`	Il motore SQL tenta di analizzare la stringa come una. `Decimal` AWS IoT tenta di analizzare le stringhe che corrispondono all'espressione regolare:. `^-?\d+(\.\d+)?((?i)E-?\d+)?$` «0", «-1.2", «5E-12" sono tutti esempi di stringhe che vengono convertit e automaticamente in `Decimal` s. AWS IoT Tenta di convertirle in a`Decimal`, quindi tronca le `String` posizioni decimali per formare un. `Decimal` `Int`
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Null`.
Undefined	`Undefined` .

Nel tipo Boolean

Tipo di argomento	Risultato
`Int`	`Undefined` . (È possibile usare in modo esplicito la funzione `cast` per trasformare 0 = False, any_nonzero_value = True).
`Decimal`	`Undefined` . (È possibile usare in modo esplicito la funzione cast per trasformare 0 = False, qualsiasi_valore_diverso_da_zero = True.)

Tipo di argomento	Risultato
`Boolean`	Valore originale.
`String`	"true"=True e "false"=False (senza distinzione tra maiuscole e minuscole). Altri valori stringa sono `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

Nel tipo String

Tipo di argomento	Risultato
`Int`	Rappresentazione di stringa del tipo `Int` in notazione standard.
`Decimal`	Stringa che rappresenta il valore `Decimal`, possibilmente in notazione scientifica.
`Boolean`	"true" o "false". Tutto in caratteri minuscoli.
`String`	Valore originale.
Array	Tipo `Array` serializzato in JSON. La stringa risultante è un elenco separato da virgole, racchiuso tra parentesi quadre. I tipi `String` sono racchiusi tra virgolette. I tipi `Decimal`, `Int`, `Boolean` e `Null` non sono racchiusi tra virgolette.
Oggetto	Oggetto serializzato in JSON. La stringa risultante è un elenco separato da virgole di

Tipo di argomento	Risultato
	coppie chiave-valore e inizia e termina con parentesi graffe. I tipi `String` sono racchiusi tra virgolette. I tipi `Decimal`, `Int`, `Boolean` e `Null` non sono racchiusi tra virgolette.
Null	`Undefined` .
Undefined	Undefined.

Operatori

Gli operatori seguenti possono essere usati nelle clausole SELECT e WHERE.

Operatore AND

Restituisce un risultato `Boolean`. Esegue un'operazione AND logica. Restituisce true se gli operandi sinistro e destro sono true. In caso contrario, restituisce false. Sono necessari operandi di tipo `Boolean` o operandi di tipo String "true" o "false" che non facciano distinzione tra maiuscole e minuscole.

Sintassi: *`expression`* `AND` *`expression`*.

Operatore AND

Operando sinistro	Operando destro	Output
`Boolean`	`Boolean`	`Boolean`. True se entrambi gli operandi sono true. In caso contrario, false.
`String`/`Boolean`	`String`/`Boolean`	Se tutte le stringhe sono "true" o "false" (senza distinzione tra maiuscole e minuscole), vengono convertite in `Boolean` ed elaborate normalmente come tipi *`boolean`* `AND` *`boolean`*.
Altro valore	Altro valore	`Undefined` .

Operatore OR

Restituisce un risultato `Boolean`. Esegue un'operazione OR logica. Restituisce true se l'operando sinistro o quello destro è true. In caso contrario, restituisce false. Sono necessari operandi di tipo `Boolean` o operandi di tipo String "true" o "false" che non facciano distinzione tra maiuscole e minuscole.

Sintassi: *`expression`* `OR` *`expression`*.

Operatore OR

Operando sinistro	Operando destro	Output
`Boolean`	`Boolean`	`Boolean`. True se uno degli operandi è true. In caso contrario, false.
`String/Boolean`	`String/Boolean`	Se tutte le stringhe sono "true" o "false" (senza distinzione tra maiuscole e minuscole), vengono convertite in valori booleani ed elaborate normalmente come *`boolean`* `OR` *`boolean`*.
Altro valore	Altro valore	`Undefined` .

Operatore NOT

Restituisce un risultato `Boolean`. Esegue un'operazione NOT logica. Restituisce true se l'operando è false. In caso contrario, restituisce true. È necessario un operando di tipo `Boolean` o un operando di tipo String "true" o "false" senza distinzione tra maiuscole e minuscole.

Sintassi: `NOT` *`expression`*.

Operatore NOT

Operando	Output
`Boolean`	`Boolean`. True se l'operando è false. In caso contrario, true.

Operando	Output
`String`	Se la stringa è "true" o "false" (senza distinzione tra maiuscole e minuscole), viene convertita nel valore booleano corrispondente e viene restituito il valore opposto.
Altro valore	`Undefined` .

Operatore IN

Restituisce un risultato `Boolean`. È possibile utilizzare l'operatore IN in una clausola WHERE per verificare se un valore corrisponde a qualsiasi valore in un array. Restituisce true se viene trovata la corrispondenza e false in caso contrario.

Sintassi: *`expression`* `IN` *`expression`*.

Operatore IN

Operando sinistro	Operando destro	Output
`Int/Decimal/String/A`	`Array`	Vero se l'`Object`elemento `Integer` `DecimalString/Array///`viene trovato nell'array. In caso contrario, false.

Esempio:

```
SQL: "select * from 'a/b' where 3 in arr"

JSON: {"arr":[1, 2, 3, "three", 5.7, null]}
```

In questo esempio, la clausola condition `where 3 in arr` restituirà true perché 3 è presente nell'array denominato`arr`. Quindi, nell'istruzione SQL, `select * from 'a/b'` verrà eseguito. Questo esempio mostra anche che l'array può essere eterogeneo.

Operatore EXISTS

Restituisce un risultato `Boolean`. È possibile utilizzare l'operatore EXISTS in una clausola condizionale per verificare l'esistenza di elementi in una sottoquery. Restituisce true se la sottoquery restituisce uno o più elementi e false se la sottoquery non restituisce alcun elemento.

Sintassi: *`expression`*.

Esempio:

```
SQL: "select * from 'a/b' where exists (select * from arr as a where a = 3)"

JSON: {"arr":[1, 2, 3]}
```

In questo esempio, la clausola condition `where exists (select * from arr as a where a = 3)` restituirà true perché 3 è presente nell'array denominato. `arr` Quindi, nell'istruzione SQL, `select * from 'a/b'` verrà eseguito.

Esempio:

```
SQL: select * from 'a/b' where exists (select * from e as e where foo = 2)

JSON: {"foo":4,"bar":5,"e":[{"foo":1},{"foo":2}]}
```

In questo esempio, la clausola condition `where exists (select * from e as e where foo = 2)` verrà valutata vera perché l'array all'einterno dell'oggetto JSON contiene l'oggetto. `{"foo":2}` Quindi, nell'istruzione SQL, `select * from 'a/b'` verrà eseguito.

Operatore >

Restituisce un risultato `Boolean`. Restituisce true se l'operando sinistro è maggiore dell'operando destro. Entrambi gli operandi vengono convertiti in un tipo `Decimal` e quindi confrontati.

Sintassi: *`expression`* `>` *`expression`*.

Operatore >

Operando sinistro	Operando destro	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. True se l'operando sinistro è maggiore dell'operando destro. In caso contrario, false.
`String/Int/Deci`	`String/Int/Deci`	Se tutte le stringhe possono essere convertite in `Decimal`, restituisce un valore `Boolean`. Restituisce true se l'operando sinistro è maggiore dell'operando destro. In caso contrario, false.
Altro valore	`Undefined .`	`Undefined .`

Operatore >=

Restituisce un risultato `Boolean`. Restituisce true se l'operando sinistro è maggiore o uguale all'operando destro. Entrambi gli operandi vengono convertiti in un tipo `Decimal` e quindi confrontati.

Sintassi: *`expression`* `>=` *`expression`*.

Operatore >=

Operando sinistro	Operando destro	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. True se l'operando sinistro è maggiore o uguale all'operando destro. In caso contrario, false.
`String/Int/Deci`	`String/Int/Deci`	Se tutte le stringhe possono essere convertite in `Decimal`, restituisce un valore `Boolean`. Restituisce true se l'operando sinistro è maggiore o uguale all'operando destro. In caso contrario, false.
Altro valore	`Undefined .`	`Undefined .`

Operatore <

Restituisce un risultato `Boolean`. Restituisce true se l'operando sinistro è minore dell'operando destro. Entrambi gli operandi vengono convertiti in un tipo `Decimal` e quindi confrontati.

Sintassi: *`expression`* `<` *`expression`*.

Operatore <

Operando sinistro	Operando destro	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. True se l'operando sinistro è minore dell'operando destro. In caso contrario, false.
`String/Int/Deci`	`String/Int/Deci`	Se tutte le stringhe possono essere convertite in `Decimal`, restituisce un valore `Boolean`. Restituisce true se l'operando sinistro è minore dell'operando destro. In caso contrario, false.
Altro valore	`Undefined`	`Undefined`

Operatore <=

Restituisce un risultato `Boolean`. Restituisce true se l'operando sinistro è minore o uguale all'operando destro. Entrambi gli operandi vengono convertiti in un tipo `Decimal` e quindi confrontati.

Sintassi: *`expression`* `<=` *`expression`*.

Operatore <=

Operando sinistro	Operando destro	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. True se l'operando sinistro è minore o uguale all'operando destro. In caso contrario, false.
`String/Int/Deci`	`String/Int/Deci`	Se tutte le stringhe possono essere convertite in `Decimal`, restituisce un valore `Boolean`. Restituisce

Operando sinistro	Operando destro	Output
		true se l'operando sinistro è minore o uguale all'operando destro. In caso contrario, false.
Altro valore	`Undefined`	`Undefined`

Operatore <>

Restituisce un risultato `Boolean`. Restituisce true se gli operandi sinistro e destro non sono uguali. In caso contrario, restituisce false.

Sintassi: *`expression`* `<>` *`expression`*.

Operatore **<>**

Operando sinistro	Operando destro	Output
`Int`	`Int`	True se l'operando sinistro non è uguale all'operando destro. In caso contrario, false.
`Decimal`	`Decimal`	True se l'operando sinistro non è uguale all'operando destro. In caso contrario, false. `Int` viene convertito in `Decimal` prima di essere confrontato.
`String`	`String`	True se l'operando sinistro non è uguale all'operando destro. In caso contrario, false.
Array	Array	True se gli elementi di ogni operando non sono uguali e non sono nello stesso ordine. In caso contrario, false
Oggetto	Oggetto	True se le chiavi e i valori di ogni operando non sono uguali. In caso contrario, false. L'ordine di non keys/valu es è importante.
Null	Null	Falso.
Qualsiasi valore	`Undefined`	Undefined.

Operando sinistro	Operando destro	Output
`Undefined`	Qualsiasi valore	Undefined.
Tipo non corrispondente	Tipo non corrispondente	True.

Operatore =

Restituisce un risultato `Boolean`. Restituisce true se gli operandi sinistro e destro sono uguali. In caso contrario, restituisce false.

Sintassi: *`expression`* = *`expression`*.

Operatore =

Operando sinistro	Operando destro	Output
`Int`	`Int`	True se l'operando sinistro è uguale all'operando destro. In caso contrario, false.
`Decimal`	`Decimal`	True se l'operando sinistro è uguale all'operando destro. In caso contrario, false. `Int` viene convertito in `Decimal` prima di essere confrontato.
`String`	`String`	True se l'operando sinistro è uguale all'operando destro. In caso contrario, false.
Array	Array	True se gli elementi di ogni operando sono uguali e sono nello stesso ordine. In caso contrario, false.
Oggetto	Oggetto	True se le chiavi e i valori di ogni operando sono uguali. In caso contrario, false. L'ordine di non keys/values è importante.
Qualsiasi valore	`Undefined`	`Undefined` .

Operando sinistro	Operando destro	Output
`Undefined`	Qualsiasi valore	`Undefined` .
Tipo non corrispondente	Tipo non corrispondente	Falso.

Operatore +

L'operatore "+" è un operatore di overload. Può essere usato per la concatenazione di stringhe o la somma.

Sintassi: *`expression`* + *`expression`*.

Operatore +

Operando sinistro	Operando destro	Output
`String`	Qualsiasi valore	Converte l'operando destro in una stringa e lo concatena alla fine dell'operando sinistro.
Qualsiasi valore	`String`	Converte l'operando sinistro in una stringa e lo concatena all'operando destro alla fine dell'operando sinistro convertito.
`Int`	`Int`	`Int` value. Somma gli operandi.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Somma gli operandi.
Altro valore	Altro valore	`Undefined` .

Operatore -

Sottrae l'operando destro dall'operando sinistro.

Sintassi: *`expression`* - *`expression`*.

Operatore -

Operando sinistro	Operando destro	Output
`Int`	`Int`	`Int` value. Sottrae l'operando destro dall'operando sinistro.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Sottrae l'operando destro dall'operando sinistro.
`String/Int/Deci`	`String/Int/Deci`	Se tutte le stringhe possono essere convertite in valori decimali, viene restituito un valore `Decimal`. Sottrae l'operando destro dall'operando sinistro. In caso contrario, restituisce `Undefined` .
Altro valore	Altro valore	`Undefined` .
Altro valore	Altro valore	`Undefined` .

Operatore *

Moltiplica l'operando sinistro per l'operando destro.

Sintassi: *`expression`* * *`expression`*.

Operatore *

Operando sinistro	Operando destro	Output
`Int`	`Int`	`Int` value. Moltiplica l'operando sinistro per l'operando destro.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Moltiplica l'operando sinistro per l'operando destro.
`String/Int/Deci`	`String/Int/Deci`	Se tutte le stringhe possono essere convertite in valori decimali, viene restituito un valore `Decimal`. Moltiplic

Operando sinistro	Operando destro	Output
		a l'operando sinistro per l'operando destro. In caso contrario, restituisce `Undefined` .
Altro valore	Altro valore	`Undefined` .

Operatore /

Divide l'operando sinistro per l'operando destro.

Sintassi: *`expression`* `/` *`expression`*.

Operatore /

Operando sinistro	Operando destro	Output
`Int`	`Int`	`Int` value. Divide l'operando sinistro per l'operando destro.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Divide l'operando sinistro per l'operando destro.
`String/Int/Deci`	`String/Int/Deci`	Se tutte le stringhe possono essere convertite in valori decimali, viene restituito un valore `Decimal`. Divide l'operando sinistro per l'operando destro. In caso contrario, restituisce `Undefined` .
Altro valore	Altro valore	`Undefined` .

Operatore %

Restituisce il resto della divisione dell'operando sinistro per l'operando destro.

Sintassi: *`expression`* `%` *`expression`*.

Operatore %

Operando sinistro	Operando destro	Output
`Int`	`Int`	`Int` value. Restituisce il resto della divisione dell'oper ando sinistro per l'operando destro.
`String/Int/Deci`	`String/Int/Deci`	Se tutte le stringhe possono essere convertite in valori decimali, viene restituito un valore `Decimal`. Restituisce il resto della divisione dell'operando sinistro per l'operando destro. In caso contrario, `Undefined` .
Altro valore	Altro valore	`Undefined` .

Funzioni

È possibile usare le seguenti funzioni predefinite nelle clausole SELECT o WHERE delle espressioni SQL.

Le seguenti funzioni esterne sono fatturate in modo equivalente a quelle di un'azione di regola: `aws_lambda`, `get_dynamodb()get_registry_data()`, e. `get_thing_shadow()` La `decode()`funzione viene inoltre fatturata solo quando decodifichi un messaggio Protobuf in JSON. Per maggiori dettagli, consulta la pagina dei prezzi.AWS IoT Core

abs(Decimal)

Restituisce il valore assoluto di un numero. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `abs(-5)` restituisce 5.

Tipo di argomento	Risultato
`Int`	`Int`, il valore assoluto dell'argomento.
`Decimal`	`Decimal`, il valore assoluto dell'argomento.
`Boolean`	`Undefined` .

Tipo di argomento	Risultato
`String`	`Decimal`. Il risultato è il valore assoluto dell'argomento. Se la stringa non può essere convertita, il risultato è `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

accountid()

Restituisce l'ID dell'account proprietario della regola come `String`. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`accountid()` = "123456789012"

acos(Decimal)

Restituisce il coseno inverso di un numero in radianti. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `acos(0)` = 1.5707963267948966

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), il coseno inverso dell'argomento. I risultati immaginari vengono restituiti come `Undefined` .

Tipo di argomento	Risultato
`Decimal`	`Decimal` (a precisione doppia), il coseno inverso dell'argomento. I risultati immaginari vengono restituiti come `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal`, il coseno inverso dell'argo mento. Se la stringa non può essere convertita, il risultato è `Undefined` . I risultati immaginari vengono restituiti come `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

asin(Decimal)

Restituisce il seno inverso di un numero in radianti. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `asin(0)` = 0.0

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), il seno inverso dell'argomento. I risultati immaginari vengono restituiti come `Undefined` .

Tipo di argomento	Risultato
`Decimal`	`Decimal` (a precisione doppia), il seno inverso dell'argomento. I risultati immaginari vengono restituiti come `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal` (a precisione doppia), il seno inverso dell'argomento. Se la stringa non può essere convertita, il risultato è `Undefined` . I risultati immaginari vengono restituiti come `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

atan(Decimal)

Restituisce la tangente inversa di un numero in radianti. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `atan(0)` = 0.0

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), la tangente inversa dell'argomento. I risultati immaginari vengono restituiti come `Undefined` .

Tipo di argomento	Risultato
`Decimal`	`Decimal` (a precisione doppia), la tangente inversa dell'argomento. I risultati immaginari vengono restituiti come `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal`, la tangente inversa dell'argo mento. Se la stringa non può essere convertita, il risultato è `Undefined` . I risultati immaginari vengono restituiti come `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

atan2(Decimal, Decimal)

Restituisce l'angolo in radianti, tra l'asse X positivo e il punto (x, y) definito nei due argomenti. L'angolo è positivo per gli angoli in senso antiorario (semipiano superiore, y > 0) e negativo per gli angoli in senso orario (semipiano inferiore, y < 0). Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `atan2(1, 0)` = 1.5707963267948966

Tipo di argomento	Tipo di argomento	Risultato
`Int`/`Decimal`	`Int`/`Decimal`	`Decimal` (a precisione doppia), l'asse X e il punto (x, y) specifica

Tipo di argomento	Tipo di argomento	Risultato
`Int/Decimal/String`	`Int/Decimal/String`	`Decimal`, la tangente inversa d descritto. Se una stringa non pu convertita, il risultato è `Undefi`
Altro valore	Altro valore	`Undefined` .

aws_lambda(functionArn, inputJson)

Chiama la funzione Lambda specificata passando `inputJson` alla funzione Lambda e restituisce il codice JSON generato dalla funzione Lambda.

Arguments (Argomenti)

Argomento	Description
`functionArn`	ARN della funzione Lambda da chiamare. La funzione Lambda deve restituire dati JSON.
`inputJson`	Input JSON passato alla funzione Lambda. Per passare query di oggetti nidificati e valori letterali, devi utilizzare SQL versione 23/03/2016.

È necessario concedere AWS IoT `lambda:InvokeFunction` le autorizzazioni per richiamare la funzione Lambda specificata. L'esempio seguente illustra come concedere l'autorizzazione `lambda:InvokeFunction` usando l' AWS CLI.

```
aws lambda add-permission --function-name "function_name"
--region "region"
--principal iot.amazonaws.com
--source-arn arn:aws:iot:us-east-1:account_id:rule/rule_name
--source-account "account_id"
--statement-id "unique_id"
--action "lambda:InvokeFunction"
```

Di seguito sono illustrati gli argomenti per il comando add-permission:

--function-name

Il nome della funzione Lambda. Aggiungi una nuova autorizzazione per aggiornare la policy delle risorse della funzione.

--region

Il del tuo Regione AWS account.

--principal

Entità principale che riceve l'autorizzazione. Questo dovrebbe consentire AWS IoT il `iot.amazonaws.com` permesso di chiamare una funzione Lambda.

--source-arn

ARN della regola. È possibile utilizzare il get-topic-rule AWS CLI comando per ottenere l'ARN di una regola.

--source-account

Il Account AWS luogo in cui viene definita la regola.

--statement-id

Identificatore univoco di un'istruzione.

--action

Le operazioni Lambda da permettere nell'istruzione. Per consentire a AWS IoT di invocare una funzione Lambda, specifica `lambda:InvokeFunction`.

> **Important**
>
> Se aggiungi un'autorizzazione per un AWS IoT principale senza fornire `source-arn` o`source-account`, qualsiasi cosa Account AWS che crea una regola con l'azione Lambda può attivare regole da cui richiamare la funzione Lambda. AWS IoT Per ulteriori informazioni, consulta Modello di autorizzazione Lambda.

Dato un payload di messaggio JSON, ad esempio:

```
{
    "attribute1": 21,
    "attribute2": "value"
```

```
}
```

È possibile utilizzare la funzione `aws_lambda` per chiamare la funzione Lambda come segue.

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function",
 {"payload":attribute1}) as output FROM 'topic-filter'
```

Se desideri passare il payload di messaggio MQTT completo, puoi specificare il payload JSON utilizzando "*", come nell'esempio seguente.

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function", *) as output
 FROM 'topic-filter'
```

`payload.inner.element` seleziona i dati dai messaggi pubblicati nell'argomento 'argomento/sottoargomento'.

`some.value` seleziona i dati dall'output generato dalla funzione Lambda.

> **Note**
>
> Il motore di regole limita la durata dell'esecuzione delle funzioni Lambda. Le chiamate alle funzioni Lambda dalle regole devono essere completate entro 2000 millisecondi.

bitand(Int, Int)

Esegue un'operazione AND bit per bit sulle rappresentazioni in bit dei due argomenti `Int` (convertiti). Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `bitand(13, 5)` = 5

Tipo di argomento	Tipo di argomento	Risultato
`Int`	`Int`	`Int`, un'operazione AND bit per argomenti.
`Int/Decimal`	`Int/Decimal`	`Int`, un'operazione AND bit per argomenti. Tutti i numeri non di

Tipo di argomento	Tipo di argomento	Risultato
		vengono arrotondati per difetto a `Int` più vicino. Se uno o più arg possono essere convertiti in un risultato è `Undefined` .
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, un'operazione AND bit per due argomenti. Tutte le stringhe convertite in decimali e arrotond difetto al valore `Int` più vicino. S conversione non riesce, il risulta `Undefined` .
Altro valore	Altro valore	`Undefined` .

bitor(Int, Int)

Esegue un'operazione OR bit per bit sulle rappresentazioni in bit dei due argomenti. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `bitor(8, 5)` = 13

Tipo di argomento	Tipo di argomento	Risultato
`Int`	`Int`	`Int`, l'operazione OR bit per bit argomenti.
`Int/Decimal`	`Int/Decimal`	`Int`, l'operazione OR bit per bit argomenti. Tutti i numeri non di vengono arrotondati per difetto a `Int` più vicino. Se la conversion riesce, il risultato è `Undefined`
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, l'operazione OR bit per bit argomenti. Tutte le stringhe ven convertite in decimali e arrotond difetto al valore `Int` più vicino. S

Tipo di argomento	Tipo di argomento	Risultato
		conversione non riesce, il risulta `Undefined` .
Altro valore	Altro valore	`Undefined` .

bitxor(Int, Int)

Esegue un'operazione XOR bit per bit sulle rappresentazioni in bit dei due argomenti `Int` (convertiti). Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:`bitor(13, 5)` = 8

Tipo di argomento	Tipo di argomento	Risultato
`Int`	`Int`	`Int`, un'operazione XOR bit per argomenti.
`Int/Decimal`	`Int/Decimal`	`Int`, un'operazione XOR bit per due argomenti. I numeri non di t vengono arrotondati per difetto a `Int` più vicino.
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, un'operazione XOR bit per argomenti. Le stringhe vengono e in decimali e arrotondate per d valore `Int` più vicino. Se la con non riesce, il risultato è `Undefi`
Altro valore	Altro valore	`Undefined` .

bitnot(Int)

Esegue un'operazione NOT bit per bit sulle rappresentazioni in bit dell'argomento `Int` (convertito). Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `bitnot(13)` = 2

Tipo di argomento	Risultato
`Int`	`Int`, un'operazione NOT bit per bit sull'argomento.
`Decimal`	`Int`, un'operazione NOT bit per bit sull'argomento. Il valore `Decimal` viene arrotondato per difetto al valore `Int` più vicino.
`String`	`Int`, un'operazione NOT bit per bit sull'argomento. Le stringhe vengono convertite in decimali e arrotondate per difetto al valore `Int` più vicino. Se la conversione non riesce, il risultato è `Undefined` .
Altro valore	Altro valore.

cast()

Converte un valore da un tipo di dati a un altro. Le operazioni di cast hanno un comportamento per lo più simile alle conversioni standard, con in più la possibilità di eseguire il cast dei numeri da e verso tipi booleani. Se AWS IoT non riesci a determinare come passare da un tipo a un altro, il risultato è. `Undefined` Supportata da SQL versione 2015-10-08 e versioni successive. Formato: cast (*`value`*as*`type`*).

Esempio:

```
cast(true as Int) = 1
```

Di seguito sono indicate le parole chiave che potrebbero seguire la parola "as" quando si chiama `cast`:

Per la versione SQL 2015-10-08 e 2016-03-23

Parola chiave	Risultato
`String`	Esegue il cast del valore nel tipo `String`.
Nvarchar	Esegue il cast del valore nel tipo `String`.

Parola chiave	Risultato
Testo	Esegue il cast del valore nel tipo `String`.
Ntext	Esegue il cast del valore nel tipo `String`.
varchar	Esegue il cast del valore nel tipo `String`.
`Int`	Esegue il cast del valore nel tipo `Int`.
Numero intero	Esegue il cast del valore nel tipo `Int`.
Double	Esegue il cast del valore `Decimal` (a precisione doppia).

Inoltre, per la versione SQL 2016-03-23

Parola chiave	Risultato
`Decimal`	Esegue il cast del valore nel tipo `Decimal`.
Bool	Esegue il cast del valore nel tipo `Boolean`.
`Boolean`	Esegue il cast del valore nel tipo `Boolean`.

Regole per il cast:

Cast nel tipo Decimal

Tipo di argomento	Risultato
`Int`	Tipo `Decimal` senza separatore decimale.
`Decimal`	Valore di origine. **Note** Con SQL V2 (23/03/2016), valori numerici che sono numeri interi, come `10.0`, restituiscono un valore

Tipo di argomento	Risultato
	`Int` (`10`) invece del previsto valore `Decimal` (`10.0`). Per lanciare in modo affidabile valori numerici interi come valori `Decimal`, utilizza SQL V1 (08/10/2015) per l'istruzione di query della regola.
`Boolean`	true = 1.0, false = 0.0.
`String`	Cerca di analizzare la stringa come `Decimal`. AWS IoT tenta di analizzare le stringhe che corrispondono all'espressione regolare: ^-?\d+(\.\d+)?((?i)E-?\d+)?$. "0", "-1.2", "5E-12" sono tutti esempi di stringhe che vengono convertite automaticamente in decimali.
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

Cast nel tipo Int

Tipo di argomento	Risultato
`Int`	Valore di origine.
`Decimal`	Valore di origine, arrotondato per difetto al valore `Int` più vicino.
`Boolean`	true = 1.0, false = 0.0.

Tipo di argomento	Risultato
`String`	Cerca di analizzare la stringa come `Decimal`. AWS IoT tenta di analizzare le stringhe che corrispondono all'espressione regolare: ^-?\d+(\.\d+)?((?i)E-?\d+)?$. "0", "-1.2", "5E-12" sono tutti esempi di stringhe che vengono convertite automaticamente in decimali. AWS IoT prova a convertire la stringa in un tipo `Decimal` e ad arrotonda rla per difetto al valore `Int` più vicino.
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

Cast nel tipo **`Boolean`**

Tipo di argomento	Risultato
`Int`	0 = False, qualsiasi_valore_diverso_da _zero = True.
`Decimal`	0 = False, qualsiasi_valore_diverso_da _zero = True.
`Boolean`	Valore di origine.
`String`	"true" = True e "false" = False (senza distinzione tra maiuscole e minuscole). Altri valori stringa = `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .

Tipo di argomento	Risultato
Null	`Undefined` .
Undefined	`Undefined` .

Cast nel tipo String

Tipo di argomento	Risultato
`Int`	Rappresentazione di stringa del tipo `Int`, in notazione standard.
`Decimal`	Stringa che rappresenta il valore `Decimal`, possibilmente in notazione scientifica.
`Boolean`	"true" o "false", tutto in caratteri minuscoli.
`String`	Valore di origine.
Array	Matrice serializzata in JSON. La stringa risultante è un elenco separato da virgole racchiuso tra parentesi quadre. I tipi `String` sono racchiusi tra virgolette. I tipi `Decimal`, `Int` e `Boolean` non sono racchiusi tra virgolette.
Oggetto	Oggetto serializzato in JSON. La stringa JSON è un elenco separato da virgole di coppie chiave-valore e inizia e termina con parentesi graffe. Il tipo `String` è racchiuso tra virgolette. I tipi `Decimal`, `Int`, `Boolean` e `Null` non sono racchiusi tra virgolette.
Null	`Undefined` .
Undefined	`Undefined` .

ceil(Decimal)

Arrotonda per eccesso il tipo `Decimal` specificato al valore `Int` più vicino. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`ceil(1.2)` = 2

`ceil(-1.2)` = -1

Tipo di argomento	Risultato
`Int`	`Int`, il valore dell'argomento.
`Decimal`	`Int`, il valore `Decimal` arrotondato per eccesso al valore `Int` più vicino.
`String`	`Int`. La stringa viene convertita in `Decimal` e arrotondata per eccesso al valore `Int` più vicino. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined` .
Altro valore	`Undefined` .

chr(String)

Restituisce il carattere ASCII corrispondente all'argomento `Int` specificato. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`chr(65)` = "A".

`chr(49)` = "1".

Tipo di argomento	Risultato
`Int`	Carattere corrispondente al valore ASCII specificato. Se l'argomento non è un valore ASCII valido, il risultato è `Undefined` .
`Decimal`	Carattere corrispondente al valore ASCII specificato. L'argomento `Decimal` viene arrotondato per difetto al valore `Int` più vicino. Se l'argomento non è un valore ASCII valido, il risultato è `Undefined` .
`Boolean`	`Undefined` .
`String`	Se il tipo `String` può essere convertito in `Decimal`, viene arrotondato per difetto al valore `Int` più vicino. Se l'argomento non è un valore ASCII valido, il risultato è `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Altro valore	`Undefined` .

clientid()

Restituisce l'ID del client MQTT che invia il messaggio oppure n/a se il messaggio non è stato inviato tramite MQTT. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`clientid()` = "123456789012"

concat()

Concatena matrici o stringhe. Questa funzione accetta qualsiasi numero di argomenti e restituisce un tipo `String` o `Array`. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`concat()` = `Undefined`.

`concat(1)` = "1".

`concat([1, 2, 3], 4)` = [1, 2, 3, 4].

`concat([1, 2, 3], "hello")` = [1, 2, 3, "hello"]

`concat("con", "cat")` = "concat"

`concat(1, "hello")` = "1hello"

`concat("he","is","man")` = "heisman"

`concat([1, 2, 3], "hello", [4, 5, 6])` = [1, 2, 3, "hello", 4, 5, 6]

Numero di argomenti	Risultato
0	`Undefined` .
1	L'argomento viene restituito non modificato.
2+	Se un argomento è un tipo `Array`, il risultato è una singola matrice contenente tutti gli argomenti. Se nessun argomento è un array e almeno un argomento è di tipo `String`, il risultato è la concatenazione delle rappresentazioni `String` di tutti gli argomenti. Gli argomenti vengono convertit i in stringhe usando le conversioni standard illustrate in precedenza.

cos(Decimal)

Restituisce il coseno di un numero in radianti. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`cos(0)` = 1.

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), il coseno dell'argomento. I risultati immaginari vengono restituiti come `Undefined` .
`Decimal`	`Decimal` (a precisione doppia), il coseno dell'argomento. I risultati immaginari vengono restituiti come `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal` (a precisione doppia), il coseno dell'argomento. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined` . I risultati immaginari vengono restituiti come `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

cosh(Decimal)

Restituisce il coseno iperbolico di un numero in radianti. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `cosh(2.3)` = 5.037220649268761.

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), il coseno iperbolico dell'argomento. I risultati immaginari vengono restituiti come `Undefined` .
`Decimal`	`Decimal` (a precisione doppia), il coseno iperbolico dell'argomento. I risultati immaginari vengono restituiti come `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal` (a precisione doppia), il coseno iperbolico dell'argomento. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined` . I risultati immaginari vengono restituiti come `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

decodifica(valore, decodingScheme)

Utilizzo della funzione `decode` per decodificare un valore codificato. Se la stringa decodificata è un documento JSON, viene restituito un oggetto indirizzabile. In caso contrario, la stringa decodificata viene restituita come una stringa. La funzione restituisce NULL se la stringa non può essere decodificata. Questa funzione supporta la decodifica di stringhe con codifica base64 e il formato del messaggio Protocol Buffer (protobuf).

Supportata da SQL versione 2016-03-23 e versioni successive.

value

Un valore stringa o una qualsiasi delle espressioni valide, in base a quanto definito in AWS IoT Riferimento SQL, che restituiscono una stringa.

decodingScheme

Una stringa letterale che rappresenta lo schema utilizzato per decodificare il valore. Attualmente, solo `'base64'` e `'proto'` sono supportati.

Decodifica di stringhe con codifica base64

In questo esempio, il payload del messaggio include un valore codificato.

```
{
    encoded_temp: "eyAidGVtcGVyYXR1cmUiOiAzMyB9Cg=="
}
```

La funzione `decode` in questa istruzione SQL decodifica il valore nel payload del messaggio.

```
SELECT decode(encoded_temp,"base64").temperature AS temp from 'topic/subtopic'
```

La decodifica del valore `encoded_temp` restituisce il seguente documento JSON valido, che consente all'istruzione SELECT di leggere il valore della temperatura.

```
{ "temperature": 33 }
```

Il risultato dell'istruzione SELECT in questo esempio viene mostrato di seguito.

```
{ "temp": 33 }
```

Se il valore decodificato non era un documento JSON valido, il valore decodificato dovrebbe essere restituito come stringa.

Decodifica del payload del messaggio protbuf

È possibile utilizzare la funzione di decodifica SQL per configurare una regola in grado di decodificare il payload del messaggio protobuf. Per ulteriori informazioni, consulta Decodifica dei payload dei messaggi protbuf.

> **Important**
>
> Se ometti `source-arn` o `source-account` quando imposti i permessi per un AWS IoT principale, chiunque Account AWS può richiamare la funzione Decode tramite altre regole. AWS IoT Per proteggere la tua funzionalità, consulta le politiche di Bucket nella Guida per l'utente di Amazon Simple Storage Service.

L'aspetto della firma della funzione è simile al seguente:

```
decode(<ENCODED DATA>, 'proto', '<S3 BUCKET NAME>', '<S3 OBJECT KEY>', '<PROTO NAME>',
 '<MESSAGE TYPE>')
```

`ENCODED DATA`

Specifica i dati codificati con codifica protobuf da decodificare. Se l'intero messaggio inviato alla Regola è costituito da dati con codifica protobuf, puoi fare riferimento al payload in entrata di dati binari non elaborati utilizzando *. In caso contrario, questo campo deve essere una stringa JSON con codifica base-64 e un riferimento alla stringa può essere passato direttamente.

1) Per decodificare un payload in entrata protobuf di dati binari non elaborati:

```
decode(*, 'proto', ...)
```

2) Per decodificare un messaggio con codifica protobuf rappresentato da una stringa con codifica base64 'a.b':

```
decode(a.b, 'proto', ...)
```

`proto`

Specifica i dati da decodificare in un formato messaggio protobuf. Se si specifica `base64` anziché `proto`, questa funzione decodificherà le stringhe con codifica base64 come JSON.

`S3 BUCKET NAME`

Il nome del bucket Amazon S3 in cui è stato caricato il file `FileDescriptorSet`.

`S3 OBJECT KEY`

La chiave oggetto che specifica il file `FileDescriptorSet` nel bucket Amazon S3.

`PROTO NAME`

Il nome del file `.proto` (esclusa l'estensione) da cui è stato generato il file `FileDescriptorSet`.

`MESSAGE TYPE`

Il nome della struttura di messaggi protobuf all'interno del file `FileDescriptorSet` rispetto alla quale i dati da decodificare devono essere conformi.

L'aspetto di un'espressione SQL di esempio che utilizza la funzione SQL di decodifica può essere simile al seguente:

```
SELECT VALUE decode(*, 'proto', 's3-bucket', 'messageformat.desc', 'myproto',
 'messagetype') FROM 'some/topic'
```

- `*`

  Rappresenta un payload in entrata binario, conforme al tipo di messaggio protobuf chiamato `mymessagetype`.

- `messageformat.desc`

  Il file `FileDescriptorSet` archiviato in un bucket Amazon S3 denominato `s3-bucket`.

- `myproto`

  Il file `.proto` originale utilizzato per generare il file `FileDescriptorSet` denominato `myproto.proto`.

- `messagetype`

Il tipo di messaggio chiamato `messagetype` (insieme a eventuali dipendenze importate) come definito in `myproto.proto`.

encode(value, encodingScheme)

Usa la funzione `encode` per codificare il payload, che potenzialmente può essere costituito da dati non JSON, nella rappresentazione di stringa in base allo schema di codifica. Supportata da SQL versione 2016-03-23 e versioni successive.

value

Qualsiasi espressioni valida, in base a quanto definito in AWS IoT Riferimento SQL. È possibile specificare * per codificare l'intero payload, indipendentemente dal fatto che sia in formato JSON. Se si fornisce un'espressione, il risultato della valutazione verrà prima convertito in una stringa e quindi codificato.

encodingScheme

Stringa letterale che rappresenta lo schema di codifica da usare. Attualmente è supportato solo `'base64'`.

endswith(String, String)

Restituisce un tipo `Boolean` che indica se il primo argomento `String` termina con il secondo argomento `String`. Se uno degli argomenti è `Null` oppure `Undefined`, il risultato è `Undefined`. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `endswith("cat","at")` = true.

Tipo di argomento 1	Tipo di argomento 2	Risultato
`String`	`String`	True se il primo argomento term il secondo argomento. In caso c false.
Altro valore	Altro valore	Entrambi gli argomenti vengono i in stringhe usando le regole di ne standard. True se il primo ar

Tipo di argomento 1	Tipo di argomento 2	Risultato
		termina con il secondo argomen contrario, false. Se uno degli ar `Null` oppure `Undefined` , il ri `Undefined` .

exp(Decimal)

Restituisce il valore e elevato all'argomento `Decimal`. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `exp(1)` = e.

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), argomento e ^.
`Decimal`	`Decimal` (a precisione doppia), argomento e ^.
`String`	`Decimal` (a precisione doppia), argomento e ^. Se il tipo `String` non può essere convertito in `Decimal`, il risultato è `Undefined` .
Altro valore	`Undefined` .

floor(Decimal)

Arrotonda il `Decimal` per difetto al valore `Int` più vicino. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`floor(1.2)` = 1

`floor(-1.2)` = -2

Tipo di argomento	Risultato
`Int`	`Int`, il valore dell'argomento.
`Decimal`	`Int`, il valore `Decimal` viene arrotondato per difetto al valore `Int` più vicino.
`String`	`Int`. La stringa viene convertita in `Decimal` e arrotondata per difetto al valore `Int` più vicino. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined` .
Altro valore	`Undefined` .

get

Estrae un valore da un tipo di raccolta (Array, String, Object). Al primo argomento non viene applicata alcuna conversione. La conversione viene applicata al secondo argomento come illustrato nella tabella. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`get(["a", "b", "c"], 1)` = "B USD"

`get({"a":"b"}, "a")` = "B USD"

`get("abc", 0)` = "a"

Tipo di argomento 1	Tipo di argomento 2	Risultato
Array	Qualsiasi tipo (convertito in `Int`)	Elemento con indice in base 0 d `Array` fornito dal secondo argo (convertito in `Int`). Se la conver non riesce, il risultato è `Undefi` Se l'indice non rientra nei limiti d

Tipo di argomento 1	Tipo di argomento 2	Risultato
		(negativo o >= array.length), il ri `Undefined` .
Stringa	Qualsiasi tipo (convertito in `Int`)	Carattere con indice in base 0 d fornita dal secondo argomento (ta in `Int`). Se la conversione no il risultato è `Undefined` . Se l'i rientra nei limiti della stringa (ne string.length), il risultato è `Unde`
Oggetto	`String` (non viene applicata alcuna conversione)	Valore archiviato nell'oggetto de argomento corrispondente alla c stringa fornita come secondo ar
Altro valore	Qualsiasi valore	`Undefined` .

get_dynamodb (TableName,,,,, partitionKeyName roLearn) partitionKeyValue sortKeyName sortKeyValue

Recupera i dati da una tabella Dynamo DB. `get_dynamodb()` consente di eseguire le query su una tabella Dynamo DB mentre viene valutata una regola. È possibile filtrare o aumentare i payload dei messaggi utilizzando i dati recuperati da Dynamo DB. Supportata da SQL versione 2016-03-23 e versioni successive.

`get_dynamodb()` prende i seguenti parametri:

tableName

Nome della tabella Dynamo DB su cui eseguire le query.

partitionKeyName

Il nome della chiave della partizione. Per ulteriori informazioni, consulta Chiavi Dynamo DB.

partitionKeyValue

Il valore della chiave di partizione utilizzata per identificare un record. Per ulteriori informazioni, consulta Chiavi Dynamo DB.

sortKeyName

(Facoltativo) Il nome della chiave di ordinamento. Questo parametro è obbligatorio solo se la tabella Dynamo DB interrogata utilizza una chiave composita. Per ulteriori informazioni, consulta Chiavi Dynamo DB.

sortKeyValue

(Facoltativo) Il valore della chiave di ordinamento. Questo parametro è obbligatorio solo se la tabella Dynamo DB interrogata utilizza una chiave composita. Per ulteriori informazioni, consulta Chiavi Dynamo DB.

roleArn

ARN del ruolo IAM che concede l'accesso alla tabella Dynamo DB. Il motore di regole assume questo ruolo per accedere alla tabella Dynamo DB per conto dell'utente. Evitare di utilizzare un ruolo eccessivamente permissivo. Concedere al ruolo solo le autorizzazioni richieste dalla regola. Di seguito è riportato una policy di esempio che consente l'accesso a una tabella Dynamo DB.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:GetItem",
            "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/table-
name"
        }
    ]
}
```

Per un esempio di come è possibile utilizzare `get_dynamodb()`, supponi di avere una tabella Dynamo DB che contiene l'ID del dispositivo e le informazioni sulla posizione per tutti i dispositivi collegati a AWS IoT. La seguente istruzione SELECT utilizza la funzione `get_dynamodb()` per recuperare il percorso per l'ID dispositivo specificato:

```
SELECT *, get_dynamodb("InServiceDevices", "deviceId", id,
"arn:aws:iam::12345678910:role/getdynamo").location AS location FROM 'some/
topic'
```

> **Note**
>
> - È possibile chiamare `get_dynamodb()` al massimo una volta per istruzione SQL. Più chiamate di `get_dynamodb()` in una singola istruzione SQL causano la chiusura della regola senza richiamare alcuna operazione.

get_mqtt_property(name)

Fa riferimento a una delle seguenti MQTT5 intestazioni:,, e. `contentType` `payLoadFormatIndicator` `responseTopic` `correlationData` Questa funzione accetta come argomento una qualsiasi delle seguenti stringhe letterali: `content_type`, `format_indicator`, `response_topic` e `correlation_data`. Per ulteriori informazioni, consulta la tabella Argomenti della funzione seguente.

contentType

Stringa: una stringa con codifica UTF-8 che descrive il contenuto del messaggio di pubblicazione.

payLoadFormatIndicatore

Stringa: un valore stringa Enum che indica se il payload è formattato come UTF-8. I valori validi sono `UNSPECIFIED_BYTES` e `UTF8_DATA`.

responseTopic

Stringa: una stringa con codifica UTF-8 utilizzata come il nome dell'argomento per un messaggio di risposta. L'argomento della risposta viene utilizzato per descrivere l'argomento in cui il destinatario deve pubblicare come parte del flusso richiesta-risposta. L'argomento non deve contenere caratteri jolly.

correlationData

Stringa: dati binari con codifica base64 utilizzati dal mittente del messaggio di richiesta per identificare a quale richiesta è destinato il messaggio di risposta quando viene ricevuto.

Nella tabella seguente vengono mostrati gli argomenti della funzione accettabili e i tipi di restituzione associati per la funzione `get_mqtt_property`:

Argomenti della funzione

SQL	Tipo di dati restituito (se presente)	Tipo di dati restituito (se non presente)
`get_mqtt_property( "format_indicator")`	Stringa (UNSPECIF IED_BYTES o _DATA) UTF8	Stringa (UNSPECIFIED_BYTES)
`get_mqtt_property( "content_type")`	Stringa	Undefined
`get_mqtt_property( "response_topic")`	Stringa	Undefined
`get_mqtt_property( "correlation_data")`	Stringa con codifica base64	Undefined
`get_mqtt_property( "some_invalid_name ")`	Undefined	Undefined

L'esempio seguente Rules SQL fa riferimento a una delle seguenti MQTT5 intestazioni:,, e. `contentType` `payLoadFormatIndicator` `responseTopic` `correlationData`

```
SELECT *, get_mqtt_property('content_type') as contentType,
          get_mqtt_property('format_indicator') as payloadFormatIndicator,
          get_mqtt_property('response_topic') as responseTopic,
          get_mqtt_property('correlation_data') as correlationData
FROM 'some/topic'
```

get_or_default (espressione, defaultValue)

Restituisce il valore predefinito nel secondo parametro se specificato oppure restituisce undefined, quando l'espressione nel primo parametro restituisce null, undefined o fallisce. Supportata da SQL versione 2016-03-23 e versioni successive.

> **Important**
>
> `get_or_default`non supporta direttamente payload non JSON così come sono. Se utilizzi un payload non JSON, usa le funzioni or. `encode` `decode`

`get_or_default()` prende i seguenti parametri:

expression

Qualsiasi espressione valida contenenteTipi di dati,, FunzioniValori letterali, Query di oggetti nidificatiVariabili o. Estensioni JSON

defaultValue

(Facoltativo) Qualsiasi espressione valida contenente Tipi di datiFunzioni,Valori letterali,Query di oggetti nidificati, Variabili oEstensioni JSON. Questo è il valore da restituire ogni volta che il primo argomento restituisce null, undefined o fallisce.

> **Note**
>
> Le funzioni che recuperano dati da risorse di proprietà del cliente, come get_secret, get_dynamodb, aws_lambda, get_thing_shadow, decode-protobuf e machinelearning_predict, non sono consentite per il parametro defaultValue.

La tabella seguente mostra gli argomenti delle funzioni accettabili per ogni argomento e i relativi output associati:

Primo argomento	Secondo argomento	Output
Valutazione riuscita	Qualsiasi valore o non specificato	Il valore del primo argomento.
Indefinito, nullo o non riuscito	Qualsiasi valore, incluso Undefined o Null	Il valore del secondo argomento
Indefinito, nullo o non riuscito	non specificata	`Undefined`

Esempi:

Esempio 1:

L'esempio seguente fornisce un valore DefaultValue se una tabella o una query DynamoDB fallisce:

```
SELECT
    device_id,
    get_or_default(
        get_dynamodb("DeviceConfig", "deviceId", nonExistentId,
 "arn:aws:iam::123456789012:role/ROLE_NAME"),
        {"mode": "standard", "timeout": 30, "enabled": true }
    ) as config
FROM 'device/telemetry'
```

Esempio 2:

L'esempio seguente fornisce un valore predefinito sicuro «UNKNOWN» se lo status non è definito:

```
SELECT
  get_or_default( CASE status
    WHEN 'active' THEN 'GOOD'
    WHEN 'inactive' THEN 'BAD'/
    ELSE 'UNKNOWN'
  END, 'UNKNOWN') as status_category
FROM 'topic/subtopic'
```

Esempio 3:

L'esempio seguente mostra come è possibile utilizzare get_or_default anche con un singolo parametro. Ciò è utile in scenari in cui potresti non avere un valore predefinito chiaro, ma non vuoi che l'esecuzione delle regole abbia esito negativo.

```
SELECT
  get_dynamodb("DeviceConfig", "deviceId", nonExistentId,
 "arn:aws:iam::123456789012:role/ROLE_NAME") as config
FROM 'device/telemetry'
```

Se la ricerca in DynamoDB fallisce, l'esecuzione della regola avrà esito negativo e non verrà eseguita alcuna azione. Se invece viene utilizzato il seguente codice SQL:

```
SELECT
```

```
  get_or_default(get_dynamodb("DeviceConfig", "deviceId", nonExistentId,
 "arn:aws:iam::123456789012:role/ROLE_NAME")) as config
FROM 'device/telemetry'
```

L'istruzione get_or_default restituirà a`Undefined`, quindi in questo esempio l'istruzione SELECT nel complesso valuterà a `{}` e verrà tentata qualsiasi azione relativa alla regola.

> **Important**
>
> Consigliamo di seguire queste best practice per mantenere la sicurezza quando si utilizza questa funzione:
>
> - Evita di utilizzare segreti codificati nelle definizioni delle regole, compresi i valori predefiniti
> - Utilizzatelo Gestione dei segreti AWS per la gestione di informazioni riservate

get_registry_data (RegistryAPI, ThingName, roLearn)

Recupera i dati AWS IoT del registro degli oggetti in una AWS IoT regola. È possibile leggere i dati del registro (ad esempio attributi, tipi di oggetti e gruppi di oggetti a cui appartiene un dispositivo) e utilizzare queste informazioni per filtrare, arricchire o instradare dinamicamente i messaggi. Supportata da SQL versione 2016-03-23 e versioni successive.

`get_registry_data()` prende i seguenti parametri:

API di registro

L'API del registro chiamata. I valori validi sono `DescribeThing` e `ListThingGroupsForThing`. Questi valori devono essere stringhe costanti.

thingName

Stringa: il nome dell'oggetto di cui si desidera recuperare i dati di registro.

roleArn

String: un ARN di ruolo con `iot:DescribeThing` and/or `iot:ListThingGroupsForThing` autorizzazione basata sull'API chiamata.

Il formato di risposta della `get_registry_data` funzione è lo stesso dell'API di registro chiamata. Per ulteriori informazioni, consulta DescribeThing e ListThingGroupsForThing APIs.

Esempio:

È possibile recuperare informazioni sul tipo di oggetto per consentire il filtraggio dei messaggi relativi agli eventi del AWS IoT Core ciclo di vita per gli elementi (il cui nome corrisponde all'ID del client MQTT) in cui si trova il tipo di oggetto. `testenv`

```
SELECT *
FROM '$aws/events/lifecycle/+'
WHERE
    get_registry_data("DescribeThing",clientId,[roleArn]).thingTypeName='testenv'
```

Esempio:

È possibile recuperare gli attributi dell'oggetto per un dispositivo con il nome dell'oggetto `sensor1` per tutti i messaggi inviati dal relativo dispositivo gateway. `gateway1`

```
SELECT *, get_registry_data("DescribeThing","sensor1",
[roleArn]).attributes.temperature_threhold AS device1_tempthreshold
FROM home1/gateway1/sensor1/#
```

> **Note**
>
> - È possibile chiamare al massimo `get_registry_data()` una volta per istruzione SQL e modelli sostitutivi per azioni e azioni di errore.

get_secret (secretId, secretType, chiave, roleArn)

Recupera il valore del file crittografato `SecretString` o del campo `SecretBinary` della versione corrente di un segreto in Gestione dei segreti AWS. Per ulteriori informazioni sulla creazione e la gestione di segreti, vedere CreateSecretUpdateSecret, e. PutSecretValue

`get_secret()` prende i seguenti parametri:

secretId

Stringa: Amazon Resource Name (ARN) o il nome descrittivo del segreto da recuperare.

secretType

Stringa: il tipo segreto. Valori validi: `SecretString` | `SecretBinary`.

SecretString

- Per i segreti che crei come oggetti JSON utilizzando la APIs AWS CLI, la o la Gestione dei segreti AWS console:
  - Se specifichi un valore per il parametro `key`, questa funzione restituisce il valore della chiave specificata.
  - Se non specifichi un valore per il parametro `key`, questa funzione restituisce l'intero oggetto JSON.
- Per i segreti che crei come oggetti non JSON utilizzando o il APIs : AWS CLI
  - Se specifichi un valore per il parametro `key`, questa funzione non riesce con un'eccezione.
  - Se non specifichi un valore per il parametro `key`, questa funzione restituisce il contenuto del segreto.

SecretBinary

- Se specifichi un valore per il parametro `key`, questa funzione non riesce con un'eccezione.
- Se non specifichi un valore per il parametro `key`, questa funzione restituisce il valore segreto come stringa UTF-8 con codifica base64.

Chiave

(Facoltativo) Stringa: il nome della chiave all'interno di un oggetto JSON memorizzato nel campo `SecretString` di un segreto. Utilizza questo valore quando desideri recuperare solo il valore di una chiave memorizzata in un segreto anziché l'intero oggetto JSON.

Se specifichi un valore per questo parametro e il segreto non contiene un oggetto JSON all'interno del suo campo `SecretString`, questa funzione non riesce con un'eccezione.

roleArn

Stringa: ARN di un ruolo con `secretsmanager:GetSecretValue` e autorizzazioni `secretsmanager:DescribeSecret`.

> **Note**
>
> Questa funzione restituisce sempre la versione corrente del segreto (la versione con il tag `AWSCURRENT`). Il motore AWS IoT delle regole memorizza nella cache ogni segreto per un massimo di 15 minuti. Di conseguenza, il motore delle regole può richiedere fino a 15 minuti per aggiornare un segreto. Ciò significa che se recuperi un segreto fino a 15 minuti dopo un

aggiornamento con Gestione dei segreti AWS, questa funzione potrebbe restituire la versione precedente.
Questa funzione non viene misurata, ma sono Gestione dei segreti AWS previsti costi. A causa del meccanismo di memorizzazione nella cache segreta, il motore delle regole chiama occasionalmente Gestione dei segreti AWS. Poiché il motore delle regole è un servizio completamente distribuito, è possibile che vengano visualizzate più chiamate API di Secrets Manager dal motore delle regole durante la finestra di memorizzazione nella cache, dalla durata di 15 minuti.

Esempi:

Puoi utilizzare la funzione `get_secret` in un'intestazione di autenticazione in un'operazione regola HTTPS, come nell'esempio di autenticazione della chiave API seguente.

```
"API_KEY": "${get_secret('API_KEY', 'SecretString', 'API_KEY_VALUE',
 'arn:aws:iam::12345678910:role/getsecret')}"
```

Per ulteriori informazioni sull'operazione della regola HTTPS, consulta the section called "HTTP".

get_thing_shadow (ThingName, shadowName, roLearn)

Restituisce la copia shadow specificata dell'oggetto specificato. Supportata da SQL versione 2016-03-23 e versioni successive.

thingName

String: nome della copia shadow dell'oggetto da recuperare.

shadowName

(Facoltativo) Stringa: il nome della copia shadow. Questo parametro è obbligatorio solo quando si fa riferimento alle copie shadow con nome.

roleArn

String: ARN di un ruolo con autorizzazione `iot:GetThingShadow`.

Esempi:

Se utilizzato con una copia shadow denominata, fornire il parametro `shadowName`.

```
SELECT * from 'topic/subtopic'
WHERE
    get_thing_shadow("MyThing","MyThingShadow","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
    .state.reported.alarm = 'ON'
```

Se utilizzato con una copia shadow senza nome, omettere il parametro `shadowName`.

```
SELECT * from 'topic/subtopic'
WHERE
    get_thing_shadow("MyThing","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
    .state.reported.alarm = 'ON'
```

get_user_properties () userPropertyKey

Fa riferimento alle proprietà utente, che è un tipo di intestazioni di proprietà supportate in. MQTT5

userProperty

Stringa: una proprietà utente è una coppia chiave-valore. Questa funzione accetta la chiave come un argomento e restituisce una matrice di tutti i valori che corrispondono alla chiave associata.

Argomenti della funzione

Per le seguenti Proprietà utente nelle intestazioni messaggio:

Chiave	Valore
some key	un valore
a different key	un valore diverso
some key	valore con chiave duplicata

Nella tabella seguente viene mostrato il comportamento SQL previsto:

SQL	Tipo di dati restituito	Valore dati restituito
get_user_properties('some key')	Array di stringhe	`['some value', 'value with duplicate key']`
get_user_properties('other key')	Array di stringhe	`['a different value']`
get_user_properties( )	Array di oggetti coppie chiave-valore	`[{'"some key": "some value"'}, {"other key": "a different value"}, {"some key": "value with duplicate key"}]`
get_user_properties('non-existent key')	Undefined	

L'esempio seguente Rules SQL fa riferimento alle proprietà utente (un tipo di intestazione di MQTT5 proprietà) nel payload:

```
SELECT *, get_user_properties('user defined property key') as userProperty
FROM 'some/topic'
```

Funzioni di hashing

AWS IoT fornisce le seguenti funzioni di hashing:

- md2
- md5
- sha1
- sha224
- sha256
- sha384
- sha512

Tutte le funzioni hash richiedono un argomento stringa. Il risultato è il valore della stringa sottoposto ad hashing. Agli argomenti non di tipo String si applicano le conversioni standard nel tipo String. Tutte le funzioni hash sono supportate da SQL versione 2015-10-08 e successive.

Esempi:

`md2("hello")` = "a9046c73e00331af68917d3804f70655"

`md5("hello")` = "5d41402abc4b2a76b9719d911017c592"

indexof(String, String)

Restituisce il primo indice (base 0) del secondo argomento come sottostringa nel primo argomento. Entrambi gli argomenti devono essere stringhe. Gli argomenti non di tipo String sono soggetti alle regole di conversione standard nel tipo String. Questa funzione non si applica alle matrici, ma solo alle stringhe. Supportata da SQL versione 2016-03-23 e versioni successive.

Esempi:

`indexof("abcd", "bc")` = 1

isNull()

Restituisce true se l'argomento è il valore `Null`. Supportata da SQL versione 2016-03-23 e versioni successive.

Esempi:

`isNull(5)` = false.

`isNull(Null)` = true.

Tipo di argomento	Risultato
`Int`	false
`Decimal`	false
`Boolean`	false

Tipo di argomento	Risultato
`String`	false
`Array`	false
`Object`	false
`Null`	true
`Undefined`	false

isUndefined()

Restituisce true se l'argomento è `Undefined`. Supportata da SQL versione 2016-03-23 e versioni successive.

Esempi:

`isUndefined(5)` = false.

`isUndefined(floor([1,2,3]))` = true.

Tipo di argomento	Risultato
`Int`	false
`Decimal`	false
`Boolean`	false
`String`	false
`Array`	false
`Object`	false
`Null`	false
`Undefined`	true

length(String)

Restituisce il numero di caratteri della stringa fornita. Agli argomenti non di tipo `String` si applicano le regole di conversione standard. Supportata da SQL versione 2016-03-23 e versioni successive.

Esempi:

`length("hi")` = 2

`length(false)` = 5

ln(Decimal)

Restituisce il logaritmo naturale dell'argomento. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `ln(e)` = 1.

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), il logaritmo naturale dell'argomento.
`Decimal`	`Decimal` (a precisione doppia), il logaritmo naturale dell'argomento.
`Boolean`	`Undefined` .
`String`	`Decimal` (a precisione doppia), il logaritmo naturale dell'argomento. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

log(Decimal)

Restituisce il logaritmo in base 10 dell'argomento. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `log(100)` = 2.0.

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), il logaritmo in base 10 dell'argomento.
`Decimal`	`Decimal` (a precisione doppia), il logaritmo in base 10 dell'argomento.
`Boolean`	`Undefined .`
`String`	`Decimal` (a precisione doppia), il logaritmo in base 10 dell'argomento. Se il tipo `String` non può essere convertito in `Decimal`, il risultato è `Undefined .`
Array	`Undefined .`
Oggetto	`Undefined .`
Null	`Undefined .`
Undefined	`Undefined .`

lower(String)

Restituisce la versione con caratteri minuscoli del tipo `String` specificato. Gli argomenti non di tipo String vengono convertiti in stringhe usando le regole di conversione standard. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`lower("HELLO")` = "hello".

`lower(["HELLO"])` = "[\"hello\"]".

lpad(String, Int)

Restituisce l'argomento `String` con spaziatura sul lato sinistro, con il numero di spazi specificato dal secondo argomento. L'argomento `Int` deve essere compreso tra 0 e 1.000. Se il valore fornito è al di fuori di questo intervallo valido, l'argomento viene impostato sul valore valido più vicino (0 o 1000). Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`lpad("hello", 2)` = " hello".

`lpad(1, 3)` = " 1"

Tipo di argomento 1	Tipo di argomento 2	Risultato
`String`	`Int`	`String`, il tipo `String` fornito c spaziatura sul lato sinistro, con di spazi corrispondente a `Int`.
`String`	`Decimal`	L'argomento `Decimal` viene arr per difetto al valore `Int` più vici `String` viene aggiunta una spa sinistra, con il numero di spazi s
`String`	`String`	Il secondo argomento viene con un tipo `Decimal`, arrotondato p al valore `Int` più vicino e al tipo viene aggiunta una spaziatura a il numero di spazi specificato. S argomento non può essere conv tipo `Int`, il risultato è `Undefine`
Altro valore	`Int`/`Decimal`/`String`	Il primo valore viene convertito i `String` usando le conversioni s e quindi al tipo `String` viene ap funzione LPAD. Se la conversio possibile, il risultato è `Undefin`

Tipo di argomento 1	Tipo di argomento 2	Risultato
Qualsiasi valore	Altro valore	`Undefined` .

ltrim(String)

Rimuove tutti gli spazi vuoti iniziali (tabulazioni e spazi) dal tipo `String` fornito. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`Ltrim(" h i ")` = "hi ".

Tipo di argomento	Risultato
`Int`	La rappresentazione `String` di `Int` con tutti gli spazi vuoti iniziali rimossi.
`Decimal`	La rappresentazione `String` di `Decimal` con tutti gli spazi vuoti iniziali rimossi.
`Boolean`	La rappresentazione `String` del tipo booleano ("true" o "false") con tutti gli spazi vuoti iniziali rimossi.
`String`	L'argomento con tutti gli spazi vuoti iniziali rimossi.
Array	La rappresentazione `String` del tipo `Array` (usando le regole di conversione standard) con tutti gli spazi vuoti iniziali rimossi.
Oggetto	La rappresentazione `String` del tipo Object (usando le regole di conversione standard) con tutti gli spazi vuoti iniziali rimossi.
Null	`Undefined` .

Tipo di argomento	Risultato
Undefined	`Undefined` .

machinelearning_predict(modelId, roleArn, record)

Usa la `machinelearning_predict` funzione per fare previsioni utilizzando i dati di un messaggio MQTT basato su un modello Amazon SageMaker AI. Supportata da SQL versione 2015-10-08 e versioni successive. Gli argomenti per la funzione `machinelearning_predict` sono i seguenti:

modelId

ID del modello in base a cui effettuare la previsione. L'endpoint in tempo reale del modello deve essere abilitato.

roleArn

Ruolo IAM che dispone di una policy con autorizzazioni `machinelearning:Predict` e `machinelearning:GetMLModel` e permette l'accesso al modello in base a cui viene effettuata la previsione.

record

I dati da passare all' SageMaker API AI Predict. Questo elemento deve essere rappresentato come oggetto JSON a singolo livello. Se il record è un oggetto JSON multilivello, viene appiattito mediante la serializzazione dei rispettivi valori. Ad esempio, l'oggetto JSON seguente:

```
{ "key1": {"innerKey1": "value1"}, "key2": 0}
```

diventerebbe:

```
{ "key1": "{\"innerKey1\": \"value1\"}", "key2": 0}
```

La funzione restituisce un oggetto JSON con i campi seguenti:

predictedLabel

Classificazione dell'input in base al modello.

details

Contiene gli attributi seguenti:

PredictiveModelType

Tipo di modello. I valori validi sono REGRESSION, BINARY, MULTICLASS.

Algoritmo

L'algoritmo utilizzato dall' SageMaker IA per fare previsioni. Il valore deve essere SGD.

predictedScores

Contiene il punteggio di classificazione non elaborato corrispondente a ogni etichetta.

predictedValue

Il valore previsto dall' SageMaker IA.

mod(Decimal, Decimal)

Restituisce il resto della divisione del primo argomento per il secondo argomento. Equivalente a remainder(Decimal, Decimal). È anche possibile usare "%" come operatore infisso per la stessa funzionalità di modulo. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `mod(8, 3)` = 2.

Operando sinistro	Operando destro	Output
`Int`	`Int`	`Int`, il resto della divisione del p argomento per il secondo argom
`Int/Decimal`	`Int/Decimal`	`Decimal`, il resto della divisione argomento per il secondo opera
`String/Int/Decimal`	`String/Int/Decimal`	Se tutte le stringhe vengono cor decimali, il risultato è il resto del del primo argomento per il seco argomento. In caso contrario, U .

Operando sinistro	Operando destro	Output
Altro valore	Altro valore	`Undefined` .

nanvl (,) AnyValue AnyValue

Restituisce il primo argomento, se si tratta di un tipo `Decimal` valido. In caso contrario, viene restituito il secondo argomento. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `Nanvl(8, 3)` = 8.

Tipo di argomento 1	Tipo di argomento 2	Output
Undefined	Qualsiasi valore	Secondo argomento.
Null	Qualsiasi valore	Secondo argomento.
`Decimal` (non un numero)	Qualsiasi valore	Secondo argomento.
`Decimal` (numero)	Qualsiasi valore	Primo argomento.
Altro valore	Qualsiasi valore	Primo argomento.

newuuid()

Restituisce un valore UUID casuale a 16 byte. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `newuuid()` = `123a4567-b89c-12d3-e456-789012345000`

numbytes(String)

Restituisce il numero di byte nella codifica UTF-8 della stringa fornita. Agli argomenti non di tipo `String` si applicano le regole di conversione standard. Supportata da SQL versione 2016-03-23 e versioni successive.

Esempi:

`numbytes("hi")` = 2

`numbytes("€") = 3`

parse_time(String, Long[, String])

Usa la `parse_time` funzione per formattare un timestamp in un formato leggibile dall'uomo. date/time Supportata da SQL versione 2016-03-23 e versioni successive. Per convertire una stringa di timestamp in millisecondi, consulta time_to_epoch(String, String).

La funzione `parse_time` prevede i seguenti argomenti:

pattern

(Stringa) Un date/time modello che segue i formati Joda-Time.

timestamp

(Long) Ora da formattare in millisecondi dall'epoca (Unix epoch). Consulta la funzione timestamp().

timezone

(Stringa) Fuso orario del valore di data/ora formattato. Il valore predefinito è "UTC". La funzione supporta i fusi orari Joda-Time. Questo argomento è facoltativo.

Esempi:

Quando questo messaggio viene pubblicato nell'argomento "A/B", il payload `{"ts": "1970.01.01 AD at 21:46:40 CST"}` viene inviato al bucket S3:

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", 100000000,
 'America/Belize' ) as ts FROM 'A/B'",

        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
```

```
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

Quando questo messaggio viene pubblicato nell'argomento "A/B", un payload simile a `{"ts": "2017.06.09 AD at 17:19:46 UTC"}` (ma con valore di data/ora corrente) viene inviato al bucket S3:

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", timestamp() ) as ts
 FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "ruleDisabled": false,
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

È anche possibile usare `parse_time()` come modello di sostituzione. Quando, ad esempio, questo messaggio viene pubblicato nell'argomento "A/B", il payload viene inviato al bucket S3 con key = "2017":

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT * FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "ruleDisabled": false,
        "actions": [{
```

```
            "s3": {
                "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                "bucketName": "BUCKET_NAME",
                "key": "${parse_time('yyyy', timestamp(), 'UTC')}"
            }
        }],
        "ruleName": "RULE_NAME"
    }
}
```

power(Decimal, Decimal)

Restituisce il primo argomento elevato al secondo argomento. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `power(2, 5)` = 32.0.

Tipo di argomento 1	Tipo di argomento 2	Output
`Int/Decimal`	`Int/Decimal`	Tipo `Decimal` (a precisione dop primo argomento elevato alla po secondo argomento.
`Int/Decimal/String`	`Int/Decimal/String`	Tipo `Decimal` (a precisione dop primo argomento elevato alla po secondo argomento. Le stringhe convertite in decimali. Se il tipo non può essere convertito in De risultato è `Undefined` .
Altro valore	Altro valore	`Undefined` .

principal()

Restituisce l'entità principale utilizzata dal dispositivo per l'autenticazione, in base al modo in cui il messaggio di attivazione è stato pubblicato. La tabella seguente descrive il principale restituito per ogni metodo e protocollo di pubblicazione.

Come viene pubblicato il messaggio	Protocollo	Tipo di credenziali
Client MQTT	MQTT	Certificato del dispositivo X.509
AWS IoT console (client MQTT)	MQTT	Utente o ruolo IAM
AWS CLI	HTTP	Utente o ruolo IAM
AWS IoT SDK del dispositivo	MQTT	Certificato del dispositivo X.509
AWS IoT SDK del dispositivo	MQTT over WebSocket	Utente o ruolo IAM

I seguenti esempi mostrano i diversi tipi di valori che `principal()` può restituire:

- Identificazione personale del certificato X.509: `ba67293af50bf2506f5f93469686da660c7c844e7b3950bfb16813e0d31e9373`
- ID del ruolo IAM e nome della sessione: `ABCD1EFG3HIJK2LMNOP5:my-session-name`
- Restituisce un ID utente: `ABCD1EFG3HIJK2LMNOP5`

rand()

Restituisce un tipo double pseudocasuale, distribuito uniformemente e compreso tra 0,0 e 1,0. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`rand()` = 0.8231909191640703

regexp_matches(String, String)

Restituisce true se la stringa (primo argomento) contiene una corrispondenza dell'espressione regolare (secondo argomento). Se lo usi `|` nell'espressione regolare, usala con `()`.

Esempi:

`regexp_matches("aaaa", "a{2,}")` = true.

`regexp_matches("aaaa", "b")` = false.

`regexp_matches("aaa", "(aaa|bbb)")` = true.

`regexp_matches("bbb", "(aaa|bbb)")` = true.

`regexp_matches("ccc", "(aaa|bbb)")` = false.

Primo argomento:

Tipo di argomento	Risultato
`Int`	Rappresentazione `String` del tipo `Int`.
`Decimal`	Rappresentazione `String` del tipo `Decimal`.
`Boolean`	Rappresentazione `String` del tipo booleano ("true" o "false").
`String`	Tipo `String`.
Array	Rappresentazione `String` del tipo `Array` (usando le regole di conversione standard).
Oggetto	Rappresentazione `String` del tipo Object (usando le regole di conversione standard).
Null	`Undefined` .
Undefined	`Undefined` .

Secondo argomento:

Deve essere un'espressione regex valida. I tipi non String vengono convertiti in tipi `String` usando le regole di conversione standard. A seconda del tipo, la stringa risultante potrebbe non essere un'espressione regolare valida. Se l'argomento (convertito) non è un valore regex valido, il risultato è `Undefined`.

regexp_replace(String, String, String)

Sostituisce tutte le occorrenze del secondo argomento (espressione regolare) nel primo argomento con il terzo argomento. Può fare riferimento a gruppi Capture con "$". Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`regexp_replace("abcd", "bc", "x")` = "axd".

`regexp_replace("abcd", "b(.*)d", "$1")` = "ac".

Primo argomento:

Tipo di argomento	Risultato
`Int`	Rappresentazione `String` del tipo `Int`.
`Decimal`	Rappresentazione `String` del tipo `Decimal`.
`Boolean`	Rappresentazione `String` del tipo booleano ("true" o "false").
`String`	Valore di origine.
Array	Rappresentazione `String` del tipo `Array` (usando le regole di conversione standard).
Oggetto	Rappresentazione `String` del tipo Object (usando le regole di conversione standard).
Null	`Undefined` .
Undefined	`Undefined` .

Secondo argomento:

Deve essere un'espressione regex valida. I tipi non String vengono convertiti in tipi `String` usando le regole di conversione standard. A seconda del tipo, la stringa risultante potrebbe non essere

un'espressione regolare valida. Se l'argomento (convertito) non è un'espressione regex valida, il risultato è `Undefined`.

Terzo argomento:

Deve essere una stringa di sostituzione regex valida. Può fare riferimento a gruppi Capture. I tipi non String vengono convertiti in tipi `String` usando le regole di conversione standard. Se l'argomento (convertito) non è una stringa di sostituzione regex valida, il risultato è `Undefined`.

regexp_substr(String, String)

Trova la prima corrispondenza del secondo parametro (regex) nel primo parametro. Può fare riferimento a gruppi Capture con "$". Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`regexp_substr("hihihello", "hi")` = "hi"

`regexp_substr("hihihello", "(hi)*")` = «hihi»

Primo argomento:

Tipo di argomento	Risultato
`Int`	Rappresentazione `String` del tipo `Int`.
`Decimal`	Rappresentazione `String` del tipo `Decimal`.
`Boolean`	Rappresentazione `String` del tipo booleano ("true" o "false").
`String`	Argomento `String`.
Array	Rappresentazione `String` del tipo `Array` (usando le regole di conversione standard).
Oggetto	Rappresentazione `String` del tipo Object (usando le regole di conversione standard).
Null	`Undefined` .

Tipo di argomento	Risultato
Undefined	`Undefined` .

Secondo argomento:

Deve essere un'espressione regex valida. I tipi non String vengono convertiti in tipi `String` usando le regole di conversione standard. A seconda del tipo, la stringa risultante potrebbe non essere un'espressione regolare valida. Se l'argomento (convertito) non è un'espressione regex valida, il risultato è `Undefined`.

remainder(Decimal, Decimal)

Restituisce il resto della divisione del primo argomento per il secondo argomento. Equivalente a mod(Decimal, Decimal). È anche possibile usare "%" come operatore infisso per la stessa funzionalità di modulo. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `remainder(8, 3)` = 2.

Operando sinistro	Operando destro	Output
`Int`	`Int`	`Int`, il resto della divisione del p argomento per il secondo argon
`Int/Decimal`	`Int/Decimal`	`Decimal`, il resto della divisione argomento per il secondo opera
`String/Int/Decimal`	`String/Int/Decimal`	Se tutte le stringhe vengono cor decimali, il risultato è il resto del del primo argomento per il seco argomento. In caso contrario, U .
Altro valore	Altro valore	`Undefined` .

replace(String, String, String)

Sostituisce tutte le occorrenze del secondo argomento nel primo argomento con il terzo argomento. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`replace("abcd", "bc", "x")` = `"axd"`.

`replace("abcdabcd", "b", "x")` = `"axcdaxcd"`.

Tutti gli argomenti

Tipo di argomento	Risultato
`Int`	Rappresentazione `String` del tipo `Int`.
`Decimal`	Rappresentazione `String` del tipo `Decimal`.
`Boolean`	Rappresentazione `String` del tipo booleano ("true" o "false").
`String`	Valore di origine.
Array	Rappresentazione `String` del tipo `Array` (usando le regole di conversione standard).
Oggetto	Rappresentazione `String` del tipo Object (usando le regole di conversione standard).
Null	`Undefined` .
Undefined	`Undefined` .

rpad(String, Int)

Restituisce l'argomento String con spaziatura sul lato destro, con il numero di spazi specificato nel secondo argomento. L'argomento `Int` deve essere compreso tra 0 e 1.000. Se il valore fornito è al di fuori di questo intervallo valido, l'argomento viene impostato sul valore valido più vicino (0 o 1000). Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`rpad("hello", 2)` = `"hello  "`.

`rpad(1, 3)` = `"1  "`.

Tipo di argomento 1	Tipo di argomento 2	Risultato
`String`	`Int`	Al tipo `String` viene aggiunta una spaziatura sul lato destro, con un numero di spazi corrispondente al valore `Int` fornito.
`String`	`Decimal`	L'argomento `Decimal` viene arrotondato per difetto al valore `Int` più vicino e alla stringa viene aggiunta una spaziatura sul lato destro, con un numero di spazi corrispondente al valore `Int` fornito.
`String`	`String`	Il secondo argomento viene convertito in un tipo `Decimal`, arrotondato per difetto al valore `Int` più vicino. Al tipo `String` viene aggiunta una spaziatura sul lato destro, con un numero di spazi corrispondente al valore `Int` .

Tipo di argomento 1	Tipo di argomento 2	Risultato
Altro valore	`Int`/`Decimal`/`String`	Il primo valore viene convertito in un tipo `String` usando le conversioni standard e quindi al tipo `String` viene applicata la funzione rpad. Se la conversione non è possibile, il risultato è `Undefined` .
Qualsiasi valore	Altro valore	`Undefined` .

round(Decimal)

Arrotonda il tipo `Decimal` specificato al valore `Int` più vicino. Se `Decimal` è equidistante da due valori `Int` (ad esempio, 0,5), il tipo `Decimal` viene arrotondato per eccesso. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `Round(1.2)` = 1.

`Round(1.5)` = 2.

`Round(1.7)` = 2.

`Round(-1.1)` = -1.

`Round(-1.5)` = -2.

Tipo di argomento	Risultato
`Int`	Argomento.
`Decimal`	`Decimal` viene arrotondato per difetto al valore `Int` più vicino.

Tipo di argomento	Risultato
`String`	`Decimal` viene arrotondato per difetto al valore `Int` più vicino. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined` .
Altro valore	`Undefined` .

rtrim(String)

Rimuove tutti gli spazi vuoti finali (tabulazioni e spazi) dal tipo `String` fornito. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`rtrim(" h i ") = " h i"`

Tipo di argomento	Risultato
`Int`	Rappresentazione `String` del tipo `Int`.
`Decimal`	Rappresentazione `String` del tipo `Decimal`.
`Boolean`	Rappresentazione `String` del tipo booleano ("true" o "false").
Array	Rappresentazione `String` del tipo `Array` (usando le regole di conversione standard).
Oggetto	Rappresentazione `String` del tipo Object (usando le regole di conversione standard).
Null	`Undefined` .
Undefined	`Undefined`

sign(Decimal)

Restituisce il segno di un determinato numero. Quando il segno dell'argomento è positivo, viene restituito 1. Quando il segno dell'argomento è negativo, viene restituito -1. Se l'argomento è 0, viene restituito 0. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`sign(-7)` = -1.

`sign(0)` = 0.

`sign(13)` = 1.

Tipo di argomento	Risultato
`Int`	`Int`, il segno del valore `Int`.
`Decimal`	`Int`, il segno del valore `Decimal`.
`String`	`Int`, il segno del valore `Decimal`. La stringa viene convertita in un valore `Decimal` e viene restituito il segno del valore `Decimal`. Se il tipo `String` non può essere convertito in `Decimal`, il risultato è `Undefined` . Supportata da SQL versione 2015-10-08 e versioni successive.
Altro valore	`Undefined` .

sin(Decimal)

Restituisce il seno di un numero in radianti. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `sin(0)` = 0.0

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), il seno dell'argomento.
`Decimal`	`Decimal` (a precisione doppia), il seno dell'argomento.
`Boolean`	`Undefined` .
`String`	`Decimal` (a precisione doppia), il seno dell'argomento. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
`Undefined`	`Undefined` .

sinh(Decimal)

Restituisce il seno iperbolico di un numero. I valori `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Il risultato è un valore `Decimal` a precisione doppia. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `sinh(2.3)` = 4.936961805545957

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), il seno iperbolico dell'argomento.
`Decimal`	`Decimal` (a precisione doppia), il seno iperbolico dell'argomento.

Tipo di argomento	Risultato
`Boolean`	`Undefined` .
`String`	`Decimal` (a precisione doppia), il seno iperbolico dell'argomento. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined` .
Array	`Undefined` .
Oggetto	`Undefined` .
Null	`Undefined` .
Undefined	`Undefined` .

sourceip()

Recupera l'indirizzo IP di un dispositivo o del router a cui si connette. Se il dispositivo è connesso direttamente a Internet, la funzione restituirà l'indirizzo IP di origine del dispositivo. Se il dispositivo è connesso a un router che si connette a Internet, la funzione restituirà l'indirizzo IP di origine del router. Supportato da SQL versione 2016-03-23. `sourceip()` non accetta alcun parametro.

> **Important**
>
> L'indirizzo IP di origine pubblico di un dispositivo è spesso l'indirizzo IP dell'ultimo gateway Network Address Translation (NAT), ad esempio il router o il modem via cavo del provider di servizi Internet.

Esempi:

```
sourceip()="192.158.1.38"
```

```
sourceip()="1.102.103.104"
```

```
sourceip()="2001:db8:ff00::12ab:34cd"
```

Esempio SQL:

```
SELECT *, sourceip() as deviceIp FROM 'some/topic'
```

Esempi di come utilizzare la funzione sourceip () nelle azioni delle AWS IoT Core regole:

Esempio 1

L'esempio seguente mostra come chiamare la funzione () come modello di sostituzione in un'operazione DynamoDB.

```
{
 "topicRulePayload": {
  "sql": "SELECT * AS message FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "dynamoDB": {
     "tableName": "my_ddb_table",
     "hashKeyField": "key",
     "hashKeyValue": "${sourceip()}",
     "rangeKeyField": "timestamp",
     "rangeKeyValue": "${timestamp()}",
     "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB"
    }
   }
  ]
 }
}
```

Esempio 2

L'esempio seguente mostra come aggiungere la funzione sourceip() come proprietà utente MQTT utilizzando i modelli di sostituzione.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "republish": {
```

```
      "topic": "${topic()}/republish",
      "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
      "headers": {
       "payloadFormatIndicator": "UTF8_DATA",
       "contentType": "rule/contentType",
       "correlationData": "cnVsZSBjb3JyZWxhdGlvbiBkYXRh",
       "userProperties": [
        {
         "key": "ruleKey1",
         "value": "ruleValue1"
        },
        {
         "key": "sourceip",
         "value": "${sourceip()}"
        }
       ]
      }
     }
    }
   ]
  }
}
```

È possibile recuperare l'indirizzo IP di origine dai messaggi che passano alle AWS IoT Core regole dai percorsi Message Broker e Basic Ingest. Puoi anche recuperare l'IP di origine per entrambi i messaggi. IPv4 IPv6 L'IP di origine verrà visualizzato come segue:

IPv6: `yyyy:yyyy:yyyy::yyyy:yyyy`

IPv4: `xxx.xxx.xxx.xxx`

> **Note**
>
> L'IP di origine originale non verrà passato tramite l'operazione Republish.

substring(String, Int[, Int])

Richiede un tipo `String` seguito da uno o due valori `Int`. Per un tipo `String` e un singolo argomento `Int`, questa funzione restituisce la sottostringa del tipo `String` fornito dall'indice `Int` specificato (in base 0, incluso) fino alla fine di `String`. Per un tipo `String` e due argomenti `Int`,

questa funzione restituisce la sottostringa del tipo `String` fornito dal primo argomento di indice `Int` specificato (in base 0, incluso) al secondo argomento di indice `Int` (in base 0, escluso). Gli indici inferiori a zero vengono impostati su zero. Gli indici superiori alla lunghezza di `String` vengono impostati sulla lunghezza di `String`. Per la versione con tre argomenti, se il primo indice è maggiore o uguale al secondo indice, il risultato è il tipo `String` vuoto.

Se gli argomenti forniti non sono (*`String`*,*`Int`*) o (,*`String`*,*`Int`*)*`Int`*, le conversioni standard vengono applicate agli argomenti per tentare di convertirli nei tipi corretti. Se i tipi non possono essere convertiti, il risultato della funzione è `Undefined`. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`substring("012345", 0)` = "012345".

`substring("012345", 2)` = "2345".

`substring("012345", 2.745)` = "2345".

`substring(123, 2)` = "3".

`substring("012345", -1)` = "012345".

`substring(true, 1.2)` = "rue".

`substring(false, -2.411E247)` = "false".

`substring("012345", 1, 3)` = "12".

`substring("012345", -50, 50)` = "012345".

`substring("012345", 3, 1)` = "".

sql_version()

Restituisce la versione SQL specificata in questa regola. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`sql_version()` = "2016-03-23"

sqrt(Decimal)

Restituisce la radice quadrata di un numero. Gli argomenti `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `sqrt(9)` = 3.0.

Tipo di argomento	Risultato
`Int`	Radice quadrata dell'argomento.
`Decimal`	Radice quadrata dell'argomento.
`Boolean`	`Undefined .`
`String`	Radice quadrata dell'argomento. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined .`
Array	`Undefined .`
Oggetto	`Undefined .`
Null	`Undefined .`
Undefined	`Undefined .`

startswith(String, String)

Restituisce un tipo `Boolean` che indica se il primo argomento String inizia con il secondo argomento String. Se uno degli argomenti è `Null` oppure `Undefined`, il risultato è `Undefined`. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`startswith("ranger","ran")` = true

Tipo di argomento 1	Tipo di argomento 2	Risultato
`String`	`String`	Indica se il primo argomento Str con il secondo argomento String
Altro valore	Altro valore	Entrambi gli argomenti vengono i in stringhe usando le regole di ne standard. Restituisce true se argomento String inizia con il se argomento String. Se uno degli è `Null` oppure `Undefined` , il `Undefined` .

tan(Decimal)

Restituisce la tangente di un numero in radianti. I valori `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `tan(3)` = -0.1425465430742778

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), la tangente dell'argomento.
`Decimal`	`Decimal` (a precisione doppia), la tangente dell'argomento.
`Boolean`	`Undefined` .
`String`	`Decimal` (a precisione doppia), la tangente dell'argomento. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined` .
Array	`Undefined` .

Tipo di argomento	Risultato
Oggetto	`Undefined .`
Null	`Undefined .`
Undefined	`Undefined .`

tanh(Decimal)

Restituisce la tangente iperbolica di un numero in radianti. I valori `Decimal` vengono arrotondati a un valore a precisione doppia prima dell'applicazione della funzione. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `tanh(2.3)` = 0.9800963962661914

Tipo di argomento	Risultato
`Int`	`Decimal` (a precisione doppia), la tangente iperbolica dell'argomento.
`Decimal`	`Decimal` (a precisione doppia), la tangente iperbolica dell'argomento.
`Boolean`	`Undefined .`
`String`	`Decimal` (a precisione doppia), la tangente iperbolica dell'argomento. Se la stringa non può essere convertita in un tipo `Decimal`, il risultato è `Undefined .`
Array	`Undefined .`
Oggetto	`Undefined .`
Null	`Undefined .`
Undefined	`Undefined .`

time_to_epoch(String, String)

Utilizzo della funzione `time_to_epoch` per convertire una stringa di timestamp in un numero di millisecondi nel tempo dell'epoca Unix. Supportata da SQL versione 2016-03-23 e versioni successive. Per convertire millisecondi in una stringa di timestamp formattata, consulta parse_time(String, Long[, String]).

La funzione `time_to_epoch` accetta i seguenti argomenti:

timestamp

(Stringa) Stringa dell'indicatore temporale da convertire in millisecondi dall'epoca di Unix. Se la stringa del timestamp non specifica un fuso orario, la funzione utilizza il fuso orario UTC.

pattern

(Stringa) Un date/time modello che segue i formati JDK11 temporali.

Esempi:

`time_to_epoch("2020-04-03 09:45:18 UTC+01:00", "yyyy-MM-dd HH:mm:ss VV")` = 1585903518000

`time_to_epoch("18 December 2015", "dd MMMM yyyy")` = 1450396800000

`time_to_epoch("2007-12-03 10:15:30.592 America/Los_Angeles", "yyyy-MM-dd HH:mm:ss.SSS z")` = 1196705730592

timestamp()

Restituisce il timestamp corrente in millisecondi a partire dalle 00:00:00 Coordinated Universal Time (UTC) di giovedì 1 gennaio 1970, come osservato dal motore delle regole. AWS IoT Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio: `timestamp()` = 1481825251155

topic(Decimal)

Restituisce l'argomento a cui è stato inviato il messaggio che ha attivato la regola. Se non viene specificato alcun parametro, viene restituito l'intero argomento. Il parametro `Decimal` viene utilizzato

per specificare un segmento di argomento specifico, con 1 che indica il primo segmento. Per l'argomento `foo/bar/baz`, topic(1) restituisce `foo`, topic(2) restituisce `bar` e così via. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`topic()` = "things/myThings/thingOne"

`topic(1)` = "things"

Quando viene utilizzato Basic Ingest, il prefisso iniziale dell'argomento (`$aws/rules/`*`rule-name`*) non è disponibile per la funzione dell'argomento(). Ad esempio, dato l'argomento:

`$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights`

`topic()` = "Buildings/Building5/Floor2/Room201/Lights"

`topic(3)` = "Floor2"

traceid()

Restituisce l'ID traccia (UUID) del messaggio MQTT oppure `Undefined` se il messaggio non è stato inviato tramite MQTT. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`traceid()` = "12345678-1234-1234-1234-123456789012"

transform(String, Object, Array)

Restituisce una matrice di oggetti contenenti il risultato della trasformazione specificata del parametro `Object` sul parametro `Array`.

Supportata da SQL versione 2016-03-23 e versioni successive.

Stringa

> Modalità di trasformazione da utilizzare. Consulta la tabella seguente per informazioni sulle modalità di trasformazione supportate e sulla modalità di creazione dell'oggetto `Result` da `Object` e dai parametri `Array`.

Oggetto

Un oggetto che contiene gli attributi da applicare a ciascun elemento dell'`Array`.

Array

Matrice di oggetti in cui gli attributi di `Object` vengono applicati.

Ogni oggetto in questa matrice corrisponde a un oggetto nella risposta della funzione. Ogni oggetto nella risposta della funzione contiene gli attributi presenti nell'oggetto originale e gli attributi forniti da `Object` come determinato dalla modalità di trasformazione specificata in `String`.

Parametro **`String`**	Parametro **`Object`**	Parametro **`Array`**	Risultato
`enrichArray`	Oggetto	Matrice di oggetti	Una matrice di oggetti in cui ogni oggetto contiene gli attributi di un elemento del parametro `Array` e gli attributi del parametro `Object`.
Qualsiasi altro valore	Qualsiasi valore	Qualsiasi valore	Undefined

> **Note**
>
> L'array restituito da questa funzione è limitato a 128 KiB.

Esempio 1 di funzione Transform

In questo esempio viene mostrato come la funzione transform() produce una singola matrice di oggetti da un oggetto dati e una matrice.

In questo esempio, il seguente messaggio viene pubblicato nell'argomento MQTT `A/B`.

```
{
```

```
    "attributes": {
        "data1": 1,
        "data2": 2
    },
    "values": [
        {
            "a": 3
        },
        {
            "b": 4
        },
        {
            "c": 5
        }
    ]
}
```

Questa istruzione SQL per un'operazione regola argomento utilizza la funzione transform() con un valore `String` di `enrichArray`. In questo esempio, `Object` è la proprietà `attributes` dal payload del messaggio e `Array` è la matrice `values` che contiene tre oggetti.

```
select value transform("enrichArray", attributes, values) from 'A/B'
```

Dopo aver ricevuto il payload del messaggio, l'istruzione SQL valuta la seguente risposta.

```
[
  {
    "a": 3,
    "data1": 1,
    "data2": 2
  },
  {
    "b": 4,
    "data1": 1,
    "data2": 2
  },
  {
    "c": 5,
    "data1": 1,
    "data2": 2
  }
]
```

Esempio 2 di funzione Transform

In questo esempio viene mostrato come la funzione transform() può utilizzare valori letterali per includere e rinominare singoli attributi dal payload del messaggio.

In questo esempio, il seguente messaggio viene pubblicato nell'argomento MQTT `A/B`. Questo è lo stesso messaggio utilizzato in the section called “Esempio 1 di funzione Transform”.

```
{
    "attributes": {
        "data1": 1,
        "data2": 2
    },
    "values": [
        {
            "a": 3
        },
        {
            "b": 4
        },
        {
            "c": 5
        }
    ]
}
```

Questa istruzione SQL per un'operazione regola argomento utilizza la funzione transform() con un valore `String` di `enrichArray`. L'`Object` nella funzione transform() ha un singolo attributo denominato `key` con il valore di `attributes.data1` nel payload del messaggio e `Array` è la matrice `values`, che contiene gli stessi tre oggetti utilizzati nell'esempio precedente.

```
select value transform("enrichArray", {"key": attributes.data1}, values) from 'A/B'
```

Dopo aver ricevuto il payload del messaggio, questa istruzione SQL valuta la seguente risposta. Nota come la proprietà `data1` è denominata `key` nella risposta.

```
[
  {
    "a": 3,
    "key": 1
  },
  {
```

```
    "b": 4,
    "key": 1
  },
  {
    "c": 5,
    "key": 1
  }
]
```

Esempio 3 di funzione Transform

In questo esempio viene mostrato come la funzione transform() può essere utilizzata in clausole SELECT nidificate per selezionare più attributi e creare nuovi oggetti per l'elaborazione successiva.

In questo esempio, il seguente messaggio viene pubblicato nell'argomento MQTT `A/B`.

```
{
  "data1": "example",
  "data2": {
    "a": "first attribute",
    "b": "second attribute",
    "c": [
      {
        "x": {
          "someInt": 5,
          "someString": "hello"
        },
        "y": true
      },
      {
        "x": {
          "someInt": 10,
          "someString": "world"
        },
        "y": false
      }
    ]
  }
}
```

L'`0bject` per questa funzione di trasformazione è l'oggetto restituito dall'istruzione SELECT, che contiene a ed elementi b del messaggio `data2` oggetto. Il parametro `Array` è costituito dai due oggetti della matrice `data2.c` nel messaggio originale.

```
select value transform('enrichArray', (select a, b from data2), (select value c from
 data2)) from 'A/B'
```

Con il messaggio precedente, l'istruzione SQL valuta la seguente risposta.

```
[
  {
    "x": {
      "someInt": 5,
      "someString": "hello"
    },
    "y": true,
    "a": "first attribute",
    "b": "second attribute"
  },
  {
    "x": {
      "someInt": 10,
      "someString": "world"
    },
    "y": false,
    "a": "first attribute",
    "b": "second attribute"
  }
]
```

La matrice restituita in questa risposta può essere utilizzata con operazioni di regole di argomento che supportano `batchMode`.

trim(String)

Rimuove tutti gli spazi vuoti iniziali e finali dal tipo `String` fornito. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempio:

`Trim(" hi ")` = "hi"

Tipo di argomento	Risultato
`Int`	La rappresentazione `String` di `Int` con tutti gli spazi vuoti iniziali e finali rimossi.
`Decimal`	La rappresentazione `String` di `Decimal` con tutti gli spazi vuoti iniziali e finali rimossi.
`Boolean`	La rappresentazione `String` di `Boolean` ("true" o "false") con tutti gli spazi vuoti iniziali e finali rimossi.
`String`	`String` con tutti gli spazi vuoti iniziali e finali rimossi.
Array	Rappresentazione `String` del tipo `Array` usando le regole di conversione standard.
Oggetto	Rappresentazione `String` del tipo Object usando le regole di conversione standard.
Null	`Undefined` .
Undefined	`Undefined` .

trunc(Decimal, Int)

Tronca il primo argomento al numero di posizioni `Decimal` specificato nel secondo argomento. Se il secondo argomento è inferiore a zero, viene impostato su zero. Se il secondo argomento è superiore a 34, viene impostato su 34. Gli zeri finali vengono eliminati dal risultato. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`trunc(2.3, 0)` = 2.

`trunc(2.3123, 2)` = 2.31.

`trunc(2.888, 2)` = 2.88.

`trunc(2.00, 5)` = 2.

Tipo di argomento 1	Tipo di argomento 2	Risultato
`Int`	`Int`	Valore di origine.
`Int/Decimal`	`Int/Decimal`	Il primo argomento viene tronca lunghezza indicata dal secondo . Il secondo argomento, se non `Int`, viene arrotondato per difet `Int` più vicino.
`Int/Decimal/String`	`Int/Decimal`	Il primo argomento viene tronca lunghezza indicata dal secondo . Il secondo argomento, se non `Int`, viene arrotondato per difet `Int` più vicino. Un tipo `String` convertito in un valore `Decimal` conversione della stringa non ri risultato è `Undefined` .
Altro valore		`Undefined` .

upper(String)

Restituisce la versione con caratteri maiuscoli del tipo `String` specificato. Gli argomenti non di tipo `String` vengono convertiti in `String` usando le regole di conversione standard. Supportata da SQL versione 2015-10-08 e versioni successive.

Esempi:

`upper("hello")` = "HELLO"

`upper(["hello"])` = "[\"HELLO\"]"

Valori letterali

È possibile specificare direttamente oggetti letterali nelle clausole SELECT e WHERE della regola SQL, che possono essere utili per passare informazioni.

> **Note**
>
> I valori letterali sono disponibili solo quando si usa SQL 2016-03-23 o una versione successiva.

Viene usata la sintassi degli oggetti JSON (coppie chiave-valore, separate da virgole, dove le chiavi sono stringhe e i valori sono valori JSON, racchiusi tra parentesi graffe {}). Ad esempio:

Payload in ingresso pubblicato nell'argomento `topic/subtopic`: `{"lat_long": [47.606,-122.332]}`

Istruzione SQL: `SELECT {'latitude': get(lat_long, 0),'longitude':get(lat_long, 1)} as lat_long FROM 'topic/subtopic'`

Il payload in uscita risultante sarebbe: `{"lat_long": {"latitude":47.606,"longitude":-122.332}}`.

È anche possibile specificare direttamente matrici nelle clausole SELECT e WHERE della regola SQL, per poter raggruppare le informazioni. Viene usata la sintassi JSON (elementi separati da virgola racchiusi tra parentesi quadre [] per creare una matrice letterale). Ad esempio:

Payload in ingresso pubblicato nell'argomento `topic/subtopic`: `{"lat": 47.696, "long": -122.332}`

Istruzione SQL: `SELECT [lat,long] as lat_long FROM 'topic/subtopic'`

Il payload in uscita risultante sarebbe: `{"lat_long": [47.606,-122.332]}`.

Istruzioni case

Le istruzioni case possono essere usate per l'esecuzione con diramazioni, come un'istruzione switch.

Sintassi:

```
CASE v WHEN t[1] THEN r[1]
  WHEN t[2] THEN r[2] ...
  WHEN t[n] THEN r[n]
  ELSE r[e] END
```

L'espressione *v* viene valutata e confrontata per verificare la corrispondenza con il valore *t[i]* di ogni espressione `WHEN`. Se viene trovata una corrispondenza, la clausola *r[i]* diventa il risultato

dell'istruzione `CASE`. Le clausole `WHEN` vengono valutate in ordine, cosicché, in caso di più di una clausola di corrispondenza, il risultato della prima clausola di corrispondenza diventi il risultato dell'istruzione `CASE`. Se non ci sono corrispondenze, il risultato è il *r[e]* della clausola `ELSE`. Se non c'è corrispondenza e nessuna clausola `ELSE`, il risultato è `Undefined`.

Le istruzioni `CASE` richiedono almeno una clausola `WHEN`. Una clausola `ELSE` è facoltativa.

Ad esempio:

Payload in ingresso pubblicato nell'argomento `topic/subtopic`:

```
{
    "color":"yellow"
}
```

Istruzione SQL:

```
SELECT CASE color
        WHEN 'green' THEN 'go'
        WHEN 'yellow' THEN 'caution'
        WHEN 'red' THEN 'stop'
        ELSE 'you are not at a stop light' END as instructions
    FROM 'topic/subtopic'
```

Il payload in uscita risultante sarebbe:

```
{
    "instructions":"caution"
}
```

> **Note**
>
> Se *v* è `Undefined`, il risultato dell'istruzione del caso è `Undefined`.

Estensioni JSON

È possibile usare le estensioni seguenti nella sintassi SQL ANSI per semplificare l'uso degli oggetti JSON nidificati.

Operatore "."

Questo operatore accede ai membri degli oggetti e delle funzioni JSON incorporati in modo identico a ANSI SQL e. JavaScript Esempio:

```
SELECT foo.bar AS bar.baz FROM 'topic/subtopic'
```

seleziona il valore della proprietà `bar` nell'oggetto `foo` dal seguente payload del messaggio inviato all'argomento `topic/subtopic`.

```
{
  "foo": {
    "bar": "RED",
    "bar1": "GREEN",
    "bar2": "BLUE"
  }
}
```

Se il nome di una proprietà JSON include caratteri quali il trattino o caratteri numerici, la notazione "punto" non funzionerà. Invece, devi utilizzare get function per estrarre il valore della proprietà.

In questo esempio viene inviato il seguente messaggio all'argomento `iot/rules`.

```
{
  "mydata": {
    "item2": {
      "0": {
        "my-key": "myValue"
      }
    }
  }
}
```

Normalmente, il valore di `my-key` verrebbe identificato come in questa query.

```
SELECT * from iot/rules WHERE mydata.item2.0.my-key= "myValue"
```

Tuttavia, poiché il nome della proprietà `my-key` contiene un trattino e `item2` contiene un carattere numerico, get function deve essere utilizzato come mostra la seguente query.

```
SELECT * from 'iot/rules' WHERE get(get(get(mydata,"item2"),"0"),"my-key") = "myValue"
```

Operatore *

Funziona nello stesso modo del carattere jolly * in SQL ANSI. Viene usato solo nella clausola SELECT e crea un nuovo oggetto JSON contenente i dati del messaggio. Se il payload del messaggio non è in formato JSON, * restituisce l'intero payload del messaggio come byte non elaborati. Ad esempio:

```
SELECT * FROM 'topic/subtopic'
```

Applicazione di un funzione a un valore di attributo

Di seguito è illustrato un esempio di payload JSON che potrebbe essere pubblicato da un dispositivo:

```
{
    "deviceid" : "iot123",
    "temp" : 54.98,
    "humidity" : 32.43,
    "coords" : {
        "latitude" : 47.615694,
        "longitude" : -122.3359976
    }
}
```

L'esempio seguente applica una funzione a un valore di attributo in un payload JSON:

```
SELECT temp, md5(deviceid) AS hashed_id FROM topic/#
```

Il risultato di questa query è l'oggetto JSON seguente:

```
{
   "temp": 54.98,
   "hashed_id": "e37f81fb397e595c4aeb5645b8cbbbd1"
}
```

Modelli di sostituzione

Puoi utilizzare un modello sostitutivo per aumentare i dati JSON restituiti quando una regola viene attivata ed esegue un'azione. AWS IoT La sintassi per un modello sostitutivo è ${ espressione}, dove espressione può essere qualsiasi espressione supportata dalle clausole SELECT, dalle AWS IoT clausole WHERE e. AWS IoT azioni relative alle regole Questa espressione può essere inserita in un campo dell'operazione di una regola, consentendo di configurare dinamicamente un'operazione.

In effetti, questa funzione sostituisce un'informazione in un'operazione. Questo include funzioni, operatori e informazioni presenti nel payload del messaggio originale.

> **Important**
>
> Dal momento che un'espressione in un modello di sostituzione viene valutata separatamente dall'istruzione "SELECT...", non è possibile fare riferimento a un alias creato utilizzando la clausola AS. È possibile fare riferimento solo alle informazioni presenti nel payload originale, oltre alle funzioni e agli operatori.

Per ulteriori informazioni sulle espressioni supportate, consulta AWS IoT Riferimento SQL.

Le seguenti operazioni delle regole supportano i modelli di sostituzione. Ogni operazione supporta campi diversi che possono essere sostituiti.

- Apache Kafka
- CloudWatch allarmi
- CloudWatch Registri
- CloudWatch metriche
- DynamoDB
- Dynamo DBv2
- Elasticsearch
- HTTP
- AWS IoT Events
- AWS IoT SiteWise
- Flussi di dati Kinesis
- Firehose
- Lambda
- Location (Ubicazione)
- OpenSearch
- Ripubblica
- S3
- SNS

- SQS
- Step Functions
- Timestream

I modelli di sostituzione vengono visualizzati nei parametri di azione all'interno di una regola:

```
{
    "sql": "SELECT *, timestamp() AS timestamp FROM 'my/iot/topic'",
    "ruleDisabled": false,
    "actions": [{
        "republish": {
            "topic": "${topic()}/republish",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

Se questa regola viene attivata dal codice JSON seguente pubblicato su `my/iot/topic`:

```
{
    "deviceid": "iot123",
    "temp": 54.98,
    "humidity": 32.43,
    "coords": {
        "latitude": 47.615694,
        "longitude": -122.3359976
    }
}
```

Quindi questa regola pubblica il seguente codice JSON in, che sostituisce da: `my/iot/topic/republish` AWS IoT `${topic()}/republish`

```
{
    "deviceid": "iot123",
    "temp": 54.98,
    "humidity": 32.43,
    "coords": {
        "latitude": 47.615694,
        "longitude": -122.3359976
    },
    "timestamp": 1579637878451
```

```
}
```

Query di oggetti nidificati

Puoi utilizzare le clausole SELECT nidificate per eseguire le query sugli attributi all'interno di matrici e oggetti JSON interni. Supportata da SQL versione 2016-03-23 e versioni successive.

Considera il seguente messaggio MQTT:

```
{
    "e": [
        { "n": "temperature", "u": "Cel", "t": 1234, "v": 22.5 },
        { "n": "light", "u": "lm", "t": 1235, "v": 135 },
        { "n": "acidity", "u": "pH", "t": 1235, "v": 7 }
    ]
}
```

Example

Puoi convertire i valori in una nuova matrice con la seguente regola.

```
SELECT (SELECT VALUE n FROM e) as sensors FROM 'my/topic'
```

La regola genera l'output seguente.

```
{
    "sensors": [
        "temperature",
        "light",
        "acidity"
    ]
}
```

Example

Utilizzando lo stesso messaggio MQTT, puoi anche eseguire le query su un valore specifico all'interno di un oggetto nidificato con la seguente regola.

```
SELECT (SELECT v FROM e WHERE n = 'temperature') as temperature FROM 'my/topic'
```

La regola genera l'output seguente.

```
{
    "temperature": [
        {
            "v": 22.5
        }
    ]
}
```

Example

Puoi anche livellare l'output con una regola più complicata.

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'), 0).v as temperature FROM 'topic'
```

La regola genera l'output seguente.

```
{
    "temperature": 22.5
}
```

Utilizzo di payload binari

Quando il payload del messaggio deve essere gestito come dati binari non elaborati (invece che come oggetto JSON), è possibile usare l'operatore * per farvi riferimento in una clausola SELECT.

In questo argomento:

- Esempi di payload binari
- Decodifica dei payload del messaggio protbuf

Esempi di payload binari

Quando si utilizza * per fare riferimento al payload del messaggio come dati binari non elaborati, puoi aggiungere dati alla regola. Se si dispone di un payload vuoto o in formato JSON, il payload risultante può aggiungere dati utilizzando la regola. Di seguito vengono riportati degli esempi di clausole `SELECT` supportate.

- È possibile utilizzare le seguenti clausole `SELECT` solo con un * per payload binari.
  - ```
    SELECT * FROM 'topic/subtopic'
    ```
    ```

- ```
  SELECT * FROM 'topic/subtopic' WHERE timestamp() % 12 = 0
  ```

- Puoi inoltre aggiungere dati e utilizzare le seguenti clausole SELECT.
  - ```
    SELECT *, principal() as principal, timestamp() as time FROM 'topic/subtopic'
    ```
  - ```
    SELECT encode(*, 'base64') AS data, timestamp() AS ts FROM 'topic/subtopic'
    ```
- È possibile utilizzare anche queste clausole SELECT con payload binari.
  - Quanto segue si riferisce al `device_type` nella clausola WHERE.

    ```
    SELECT * FROM 'topic/subtopic' WHERE device_type = 'thermostat'
    ```
  - Il seguente elemento non è supportato.

    ```
    {
     "sql": "SELECT * FROM 'topic/subtopic'",
     "actions": [
      {
       "republish": {
        "topic": "device/${device_id}"
       }
      }
     ]
    }
    ```

Le seguenti operazioni della regola non supportano i payload binari, quindi è necessario decodificarli.

- Alcune operazioni delle regole non supportano l'input del payload binario, come ad esempio una operazione Lambda, pertanto è necessario decodificare i payload binari. L'operazione della regola Lambda può ricevere dati binari se è codificata in base64 e in un payload JSON. Puoi farlo modificando la regola esistente con la seguente.

  ```
  SELECT encode(*, 'base64') AS data FROM 'my_topic'
  ```

- L'istruzione SQL non supporta la stringa come input. Per convertire un input di stringa in JSON, puoi eseguire il seguente comando.

  ```
  SELECT decode(encode(*, 'base64'), 'base64') AS payload FROM 'topic'
  ```
  ```

Decodifica dei payload del messaggio protbuf

Protocol Buffers (protobuf) è un formato di dati open-source utilizzato per serializzare dati strutturati in una forma binaria, compatta. Viene utilizzato per la trasmissione di dati su reti o l'archiviazione in file. Protobuf consente di inviare dati in pacchetti di piccole dimensioni e a una velocità maggiore rispetto ad altri formati di messaggistica. AWS IoT Core Le regole supportano protobuf fornendo la funzione SQL decode (value, decodingScheme), che consente di decodificare i payload di messaggi con codifica protobuf in formato JSON e di indirizzarli ai servizi downstream. Questa sezione step-by-step descrive in dettaglio il processo per configurare la AWS IoT Core decodifica protobuf in Rules.

In questa sezione:

- Prerequisiti
- Creazione di file descrittori
- Caricamento dei file descrittori nel bucket S3
- Configurazione della decodifica protobuf in Rules
- Limitazioni
- Best practice

Prerequisiti

- Una conoscenza di base di Protocol Buffers (protobuf)
- I file `.proto` che definiscono i tipi di messaggio e le dipendenze correlate
- Installazione di Protobuf Compiler (protoc) nel sistema

Creazione di file descrittori

Se si dispone già di file descrittori, è possibile ignorare questa fase. Un file descrittore (`.desc`) è una versione compilata di un file `.proto`, ovvero un file di testo che definisce le strutture di dati e i tipi di messaggi da utilizzare in una serializzazione protobuf. Per generare un file descrittore, è necessario definire un file `.proto` e utilizzare il compilatore protoc per compilarlo.

1. Crea i file `.proto` che definiscono i tipi di messaggio. Un file `.proto` di esempio può avere l'aspetto seguente:

```
syntax = "proto3";
```

```
message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;
}
```

In questo file `.proto` di esempio, si utilizza la sintassi proto3 e si definisce il tipo di messaggio `Person`. La definizione del messaggio `Person` specifica tre campi (name, id ed email). Per ulteriori informazioni sui formati messaggio del file `.proto`, consulta Guida alle lingue (proto3).

2. Utilizza il compilatore protoc per compilare i file `.proto` e generare un file descrittore. Di seguito è riportato un comando di esempio per creare un file descrittore (`.desc`):

```
protoc --descriptor_set_out=<FILENAME>.desc \
    --proto_path=<PATH_TO_IMPORTS_DIRECTORY> \
    --include_imports \
    <PROTO_FILENAME>.proto
```

Questo comando di esempio genera un file descrittore`<FILENAME>.desc`, che AWS IoT Core Rules può utilizzare per decodificare i payload protobuf conformi alla struttura di dati definita in. `<PROTO_FILENAME>.proto`

- `--descriptor_set_out`

  Specifica il nome del file descrittore (`<FILENAME>.desc`) da generare.

- `--proto_path`

  Specifica le posizioni degli eventuali file `.proto` importati a cui fa riferimento il file in fase di compilazione. È possibile specificare il flag più volte se si dispone di più file `.proto` importati con posizioni differenti.

- `--include_imports`

  Specifica che anche gli eventuali file `.proto` importati devono essere compilati e inclusi nel file descrittore `<FILENAME>.desc`.

- `<PROTO_FILENAME>.proto`

  Specifica il nome del file `.proto` da compilare.

Per ulteriori informazioni sul riferimento protoc, consulta Documentazione di riferimento delle API.

Caricamento dei file descrittori nel bucket S3

Dopo aver creato i file descrittori`<FILENAME>.desc`, carica i file descrittori in un bucket Amazon S3, utilizzando AWS l'API AWS , l'SDK o il. `<FILENAME>.desc` Console di gestione AWS

Considerazioni importanti

- Assicurati di caricare i file descrittori in un bucket Amazon S3 Regione AWS nello stesso spazio in cui intendi Account AWS configurare le regole.
- Assicurati di concedere l' AWS IoT Core accesso per leggere i file da S3. `FileDescriptorSet` Se la crittografia lato server (SSE) è disattivata nel bucket S3 o se il bucket S3 è crittografato tramite chiavi gestite da Amazon S3 (SSE-S3), non sono necessarie configurazioni delle policy aggiuntive. Ciò può essere ottenuto con l'esempio della policy di bucket di esempio:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "Statement1",
   "Effect": "Allow",
   "Principal": {
    "Service": "iot.amazonaws.com"
   },
   "Action": "s3:Get*",
                    "Resource": "arn:aws:s3:::<BUCKET NAME>/<FILENAME>.desc"
  }
 ]
}
```

- Se il tuo bucket S3 è crittografato utilizzando una AWS Key Management Service chiave (SSE-KMS), assicurati di concedere l' AWS IoT Core autorizzazione all'uso della chiave quando accedi al bucket S3. Per farlo, aggiungi questa istruzione alla policy della chiave:

```
{
 "Sid": "Statement1",
 "Effect": "Allow",
```

```
 "Principal": {
  "Service": "iot.amazonaws.com"
 },
 "Action": [
  "kms:Decrypt",
  "kms:GenerateDataKey*",
  "kms:DescribeKey"
 ],
        "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

}
```

Configurazione della decodifica protobuf in Rules

Dopo aver caricato i file descrittori nel bucket Amazon S3, configura una Regola in grado di decodificare il formato di payload dei messaggi protobuf utilizzando la funzione SQL decode(value, decodingScheme). Una firma della funzione dettagliata e un esempio sono disponibili nella funzione SQL decode(value, decodingScheme) della Documentazione di riferimento SQL AWS IoT .

Di seguito è riportata un'espressione SQL di esempio che utilizza la funzione decode(value, decodingScheme):

```
SELECT VALUE decode(*, 'proto', '<BUCKET NAME>', '<FILENAME>.desc', '<PROTO_FILENAME>',
 '<PROTO_MESSAGE_TYPE>') FROM '<MY_TOPIC>'
```

In questa espressione di esempio:

- La funzione SQL decode(value, decodingScheme) viene utilizzata per decodificare il payload del messaggio binario a cui fa riferimento `*`. Può essere un payload con codifica protobuf binario o una stringa JSON che rappresenta un payload protobuf con codifica base64.
- Il payload del messaggio fornito è codificato utilizzando il tipo di messaggio `Person` definito in `PROTO_FILENAME.proto`.
- Il bucket Amazon S3 denominato `BUCKET NAME` contiene il `FILENAME.desc` generato da `PROTO_FILENAME.proto`.

Dopo aver completato la configurazione, pubblica un messaggio AWS IoT Core sull'argomento a cui è sottoscritta la regola.

Limitazioni

AWS IoT Core Le regole supportano protobuf con le seguenti limitazioni:

- La decodifica dei payload dei messaggi protobuf all'interno di modelli di sostituzione non è supportata.
- Quando si decodificano i payload dei messaggi protobuf, è possibile utilizzare la funzione SQL di decodifica all'interno di una singola espressione SQL fino a due volte.
- La dimensione massima del payload in entrata è 128 KiB (1 KiB = 1024 byte), la dimensione massima del payload in uscita è 128 KiB e la dimensione massima per un oggetto `FileDescriptorSet` archiviato in un bucket Amazon S3 è 32 KiB.
- I bucket Amazon S3 crittografati con crittografia SSE-C non sono supportati.

Best practice

Di seguito sono indicate alcune best practice e suggerimenti per la risoluzione dei problemi.

- Eseguire il backup dei file proto nel bucket Amazon S3.

  È buona prassi eseguire il backup dei file proto in caso si verifichino problemi. Ad esempio, se si modificano erroneamente i file proto senza backup durante l'esecuzione di protoc, si possono verificare problemi nello stack di produzione. Esistono diversi modi per eseguire il backup dei file in un bucket Amazon S3. Ad esempio, è possibile utilizzare il controllo delle versioni nei bucket S3. Per ulteriori informazioni su come eseguire il backup dei file nei bucket Amazon S3, consulta la Guida per gli sviluppatori di Amazon S3.
- Configurare la AWS IoT registrazione per visualizzare le voci di registro.

  È buona norma configurare la AWS IoT registrazione in modo da poter controllare AWS IoT i registri del proprio account. CloudWatch Quando una query SQL di una regola richiama una funzione esterna, AWS IoT Core Rules genera una voce di registro con un `eventType` of`FunctionExecution`, che contiene il campo del motivo che consente di risolvere gli errori. I possibili errori includono un oggetto Amazon S3 non trovato o un descrittore del file protobuf non valido. Per ulteriori informazioni su come configurare la registrazione AWS IoT e visualizzare le voci di log, consulta Configurazione della registrazione AWS IoT e Voci di registro del motore delle regole.
- Aggiornare `FileDescriptorSet` utilizzando una nuova chiave oggetto e aggiornare la chiave oggetto nella regola.

È possibile aggiornare `FileDescriptorSet` caricando un file descrittore aggiornato nel bucket Amazon S3. Gli aggiornamenti a `FileDescriptorSet` possono richiedere fino a 15 minuti per essere visualizzati. Per evitare questo ritardo, è buona prassi caricare il `FileDescriptorSet` aggiornato utilizzando una nuova chiave oggetto e aggiornare la chiave oggetto nella regola.

Versioni SQL

Il motore AWS IoT delle regole utilizza una sintassi simile a SQL per selezionare i dati dai messaggi MQTT. Le istruzioni SQL vengono interpretate in base a una versione di SQL specificata con la proprietà `awsIotSqlVersion` in un documento JSON che descrive la regola. Per ulteriori informazioni sulla struttura dei documenti di regole JSON, consulta la pagina relativa alla creazione di una regola. La `awsIotSqlVersion` proprietà consente di specificare quale versione del motore di regole AWS IoT SQL si desidera utilizzare. Quando viene distribuita una nuova versione, puoi continuare a usare una versione precedente o modificare la regola per usare la nuova versione. Le regole correnti continuano a usare la versione con cui sono state create.

L'esempio JSON seguente ti illustra come specificare la versione SQL usando la proprietà `awsIotSqlVersion`.

```
{
    "sql": "expression",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
        "republish": {
            "topic": "my-mqtt-topic",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

AWS IoT attualmente supporta le seguenti versioni SQL:

- `2016-03-23` - versione SQL creata il 23/03/2016 (consigliata).
- `2015-10-08` – versione SQL originale creata il 08/10/2015.
- `beta` – la più recente versione beta di SQL. Questa versione potrebbe comportare modifiche che interrompono il funzionamento delle regole.

Novità della versione del motore di regole SQL 2016-03-23

- Correzioni per la selezione di oggetti JSON nidificati.
- Correzioni per le query su matrici.
- Supporto per query tra oggetti. Per ulteriori informazioni, consulta Query di oggetti nidificati.
- Supporto per l'output di una matrice come oggetto di primo livello.
- Aggiunta della funzione `encode(value, encodingScheme)`, che può essere applicata ai dati in formato JSON e non JSON. Per ulteriori informazioni, consulta la funzione di codifica.

Output di un oggetto **`Array`** come oggetto di primo livello

Questa caratteristica permette a una regola di restituire una matrice come oggetto di primo livello. Ad esempio, partendo dal messaggio MQTT seguente:

```
{
    "a": {"b":"c"},
    "arr":[1,2,3,4]
}
```

E dalla regola seguente:

```
SELECT VALUE arr FROM 'topic'
```

La regola genera l'output seguente.

```
[1,2,3,4]
```

AWS IoT Servizio Device Shadow

Il servizio AWS IoT Device Shadow aggiunge ombre agli oggetti AWS IoT oggetto. Le ombre possono rendere disponibile lo stato di un dispositivo ad app e altri servizi, indipendentemente dal fatto che il dispositivo sia connesso AWS IoT o meno. AWS IoT gli oggetti thing possono avere più ombre denominate in modo che la soluzione IoT abbia più opzioni per connettere i dispositivi ad altre app e servizi.

AWS IoT gli oggetti thing non hanno ombre finché non vengono creati esplicitamente. Le ombre possono essere create, aggiornate ed eliminate utilizzando la console. AWS IoT Dispositivi, altri client web e servizi possono creare, aggiornare ed eliminare copie shadow utilizzando MQTT e gli argomenti MQTT riservati, HTTP utilizzando l'API REST Device Shadow e AWS CLI per AWS IoT. Poiché le ombre vengono archiviate AWS nel cloud, possono raccogliere e segnalare dati sullo stato del dispositivo da app e altri servizi cloud, indipendentemente dal fatto che il dispositivo sia connesso o meno.

Uso delle copie shadow

Le copie shadow forniscono uno store dati affidabile per dispositivi, app e altri servizi cloud per condividere i dati. Consentono a dispositivi, app e altri servizi cloud di connettersi e disconnettersi senza perdere lo stato di un dispositivo.

Mentre i dispositivi, le app e gli altri servizi cloud sono connessi AWS IoT, possono accedere e controllare lo stato corrente di un dispositivo attraverso le sue ombre. Ad esempio, un'app può richiedere una modifica dello stato di un dispositivo aggiornando un'ombra. AWS IoT pubblica un messaggio che indica la modifica apportata al dispositivo. Il dispositivo riceve questo messaggio, aggiorna lo stato in modo che corrisponda e pubblica un messaggio con lo stato aggiornato. Il servizio Device Shadow riflette questo stato aggiornato nella copia shadow corrispondente. L'app può sottoscrivere l'aggiornamento della copia shadow oppure può interrogare la copia shadow per il suo stato corrente.

Quando un dispositivo va offline, un'app può continuare a comunicare con le ombre del dispositivo AWS IoT e con le ombre del dispositivo. Quando il dispositivo si riconnette, riceve lo stato corrente delle sue copie shadow in modo che possa aggiornarne lo stato affinché corrisponda a quello delle sue copie shadow, quindi pubblicare un messaggio con il relativo stato aggiornato. Allo stesso modo, quando un'app non è in linea e lo stato del dispositivo cambia mentre è offline, il dispositivo mantiene

aggiornata la copia shadow in modo che l'app possa interrogare le copie shadow per il suo stato corrente quando si riconnette.

Se i tuoi dispositivi sono spesso offline e desideri configurarli in modo che ricevano messaggi delta dopo la riconnessione, puoi utilizzare la funzione di sessione persistente. Per ulteriori informazioni sul periodo di scadenza della sessione persistente, consulta la sezione Periodo di scadenza della sessione persistente.

Scelta di utilizzare copie shadow con nome o senza nome

Il servizio Device Shadow supporta le copie shadow con nome e senza nome o classiche. Un oggetto può avere più copie shadow con nome e non più di una copia shadow senza nome. L'oggetto può anche avere una copia shadow con nome riservata, che funziona in modo simile a una copia shadow con nome, tranne che non è possibile aggiornare il nome. Per ulteriori informazioni, consulta Copia shadow con nome riservata.

Un oggetto può avere copie shadow con nome e senza nome allo stesso tempo; tuttavia, l'API utilizzata per accedere a ciascuna è leggermente diversa, quindi sarebbe opportuno stabilire quale tipo di copia shadow funzionerebbe meglio per la soluzione e utilizzare solo quel tipo. Per ulteriori informazioni sull'API per accedere alle copie shadow, consulta Argomenti copie shadow.

Con le copie shadow con nome, è possibile creare diverse viste dello stato di un oggetto. Ad esempio, è possibile dividere un oggetto con molte proprietà in copie shadow con gruppi logici di proprietà, ognuno identificato dal nome shadow. È inoltre possibile limitare l'accesso alle proprietà raggruppandole in copie shadow diverse e utilizzando le policy per controllare l'accesso. Per ulteriori informazioni sulle policy da utilizzare con le device shadow, consulta Operazioni, risorse e chiavi di condizione per AWS IoT e Policy AWS IoT Core.

Le copie shadow classiche senza nome sono più semplici, ma un po' più limitate rispetto alle copie shadow con nome. Ogni AWS IoT oggetto può avere solo un'ombra senza nome. Se si prevede che la soluzione IoT abbia una necessità limitata di dati di copie shadow, questo potrebbe essere il modo in cui iniziare a utilizzare le copie shadow. Tuttavia, se si pensa di voler aggiungere altre copie shadow in futuro, prendere in considerazione l'utilizzo di copie shadow con nome fin dall'inizio.

L'indicizzazione del parco istanze supporta copie shadow senza nome e con nome in modo diverso. Per ulteriori informazioni, consulta Gestione dell'indicizzazione del parco istanze.

Accesso alle copie shadow

Ogni copia shadow dispone di un argomento MQTT e di un URL HTTP riservato che supporta le operazioni `get`, `update` e `delete` sulla copia shadow.

Le copie shadow utilizzano i documenti shadow JSON per archiviare e recuperare i dati. Il documento di una copia shadow contiene una proprietà di stato che descrive questi aspetti dello stato del dispositivo:

- `desired`

  Le app specificano gli stati desiderati delle proprietà del dispositivo aggiornando l'oggetto `desired`.
- `reported`

  I dispositivi segnalano il loro stato corrente nell'oggetto `reported`.
- `delta`

  AWS IoT riporta le differenze tra lo stato desiderato e quello riportato nell'`delta`oggetto.

I dati archiviati in una copia shadow sono determinati dalla proprietà stato del corpo del messaggio dell'operazione di aggiornamento. Le operazioni di aggiornamento successive possono modificare i valori di un oggetto dati esistente, nonché aggiungere ed eliminare chiavi e altri elementi nell'oggetto stato della copia shadow. Per ulteriori informazioni sull'accesso alle copie shadow, consulta Utilizzo delle copie shadow nei dispositivi e Utilizzo delle copie shadow in app e servizi.

> **Important**
>
> L'autorizzazione a effettuare richieste di aggiornamento deve essere limitata alle app e ai dispositivi attendibili. Ciò impedisce che la proprietà stato della copia shadow venga modificata in modo imprevisto; in caso contrario, i dispositivi e le app che utilizzano la copia shadow devono essere progettati in modo che le chiavi nella proprietà stato vengano modificate.

Utilizzo delle copie shadow in dispositivi, app e altri servizi cloud

L'utilizzo delle copie shadow in dispositivi, app e altri servizi cloud richiede coerenza e coordinamento. Il servizio AWS IoT Device Shadow memorizza lo stato shadow, invia messaggi quando lo stato shadow cambia e risponde ai messaggi che ne modificano lo stato. I dispositivi, le app e altri servizi cloud nella soluzione IoT devono gestirne lo stato e mantenerlo coerente con lo stato della copia shadow del dispositivo.

I dati dello stato della copia shadow sono dinamici e possono essere modificati da dispositivi, app e altri servizi cloud con autorizzazione per accedere alla copia shadow. Per questo motivo, è importante considerare come ogni dispositivo, app e altro servizio cloud interagiranno con la copia shadow. Ad esempio:

- I dispositivi devono scrivere solo sulla proprietà `reported` dello stato shadow quando comunicano i dati di stato alla copia shadow.
- Le app e gli altri servizi cloud devono scrivere solo nella proprietà `desired` quando comunicano le richieste di modifica dello stato al dispositivo tramite la copia shadow.

> **Important**
>
> I dati contenuti in un oggetto dati shadow sono indipendenti da quelli di altre copie shadow e altre proprietà oggetto, ad esempio gli attributi di un oggetto e il contenuto dei messaggi MQTT che il dispositivo di un oggetto potrebbe pubblicare. Un dispositivo può, tuttavia, segnalare gli stessi dati in diversi argomenti MQTT e copie shadow, se necessario.
> Un dispositivo che supporta più copie shadow deve mantenere la coerenza dei dati segnalati nelle diverse copie shadow.

Ordine dei messaggi

Non è garantito che i messaggi del AWS IoT servizio arrivino al dispositivo in un ordine specifico. Lo scenario seguente mostra ciò che accade in questo caso.

Documento sullo stato iniziale:

```
{
  "state": {
```

```
    "reported": {
      "color": "blue"
    }
  },
  "version": 9,
  "timestamp": 123456776
}
```

Aggiornamento 1:

```
{
  "state": {
    "desired": {
      "color": "RED"
    }
  },
  "version": 10,
  "timestamp": 123456777
}
```

Aggiornamento 2:

```
{
  "state": {
    "desired": {
      "color": "GREEN"
    }
  },
  "version": 11,
  "timestamp": 123456778
}
```

Documento sullo stato finale:

```
{
  "state": {
    "reported": {
      "color": "GREEN"
    }
  },
  "version": 12,
  "timestamp": 123456779
```

```
}
```

Il risultato è costituito da due messaggi delta:

```
{
  "state": {
    "color": "RED"
  },
  "version": 11,
  "timestamp": 123456778
}
```

```
{
  "state": {
    "color": "GREEN"
  },
  "version": 12,
  "timestamp": 123456779
}
```

Il dispositivo potrebbe ricevere questi messaggi non in ordine. Poiché lo stato nei messaggi è cumulativo, un dispositivo può scartare senza problemi qualsiasi messaggio contenente un numero di versione precedente a quello monitorato. Se il dispositivo riceve il delta per la versione 12 prima della versione 11, può scartare senza problemi il messaggio con la versione 11.

Taglio dei messaggi delle copie shadow

Per ridurre le dimensioni dei messaggi delle copie shadow inviati al dispositivo, definisci una regola che seleziona solo i campi necessari per il dispositivo, quindi ripubblica il messaggio in un argomento MQTT in cui il dispositivo è in ascolto.

La regola viene specificata in formato JSON e dovrebbe essere simile a quanto segue:

```
{
  "sql": "SELECT state, version FROM '$aws/things/+/shadow/update/delta'",
  "ruleDisabled": false,
  "actions": [
    {
      "republish": {
        "topic": "${topic(3)}/delta",
```

```
        "roleArn": "arn:aws:iam:123456789012:role/my-iot-role"
      }
    }
  ]
}
```

L'istruzione SELECT determina i campi del messaggio che verranno ripubblicati nell'argomento specificato. Il carattere jolly "+" permette di trovare la corrispondenza con tutti i nomi delle copie shadow. La regola specifica che tutti i messaggi corrispondenti devono essere ripubblicati nell'argomento specificato. In questo caso, la funzione `"topic()"` viene usata per specificare l'argomento su cui ripetere la pubblicazione. `topic(3)` restituisce il nome nell'oggetto nell'argomento originale. Per ulteriori informazioni sulla creazione di regole, consulta Regole per AWS IoT.

Utilizzo delle copie shadow nei dispositivi

Questa sezione descrive le comunicazioni dei dispositivi con le ombre utilizzando i messaggi MQTT, il metodo preferito dai dispositivi per comunicare con il servizio AWS IoT Device Shadow.

Le comunicazioni shadow emulano un request/response modello che utilizza il modello di comunicazione pubblica/sottoscrizione di MQTT. Ogni operazione copia shadow è costituita da un argomento di richiesta, un argomento di risposta riuscita (`accepted`) e un argomento di risposta agli errori (`rejected`).

Se si desidera che app e servizi siano in grado di determinare se un dispositivo è connesso, consulta Rilevamento della connessione di un dispositivo.

> **Important**
>
> Poiché MQTT utilizza un modello di publish/subscribe comunicazione, è necessario sottoscrivere gli argomenti di risposta prima di pubblicare un argomento di richiesta. In caso contrario, non riceverai la risposta alla richiesta pubblicata.
> Se usi un SDK per dispositivi AWS IoTper chiamare il servizio Device Shadow APIs, questo viene gestito automaticamente.

Gli esempi di questa sezione utilizzano una forma abbreviata dell'argomento in cui *`ShadowTopicPrefix`* possono fare riferimento a un'ombra con nome o senza nome, come descritto in questa tabella.

Le copie shadow possono essere con nome o senza nome (classiche). Gli argomenti utilizzati da ciascuno differiscono solo nel prefisso dell'argomento. Questa tabella mostra il prefisso dell'argomento utilizzato da ogni tipo di copia shadow.

ShadowTopicPrefix value	Tipo di copia shadow
`$aws/things/` *`thingName`* `/shadow`	Copia shadow senza nome (classica)
`$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`*	Copia shadow con nome

> **Important**
>
> Assicurati che l'uso delle copie shadow da parte dell'app o del servizio sia coerente e supportato dalle implementazioni corrispondenti nei tuoi dispositivi. Considera ad esempio, come vengono create, aggiornate ed eliminate le copie shadow. Considera inoltre come vengono gestiti gli aggiornamenti nel dispositivo e nelle app o nei servizi che accedono al dispositivo tramite una copia shadow. La progettazione dovrebbe chiarire il modo in cui lo stato del dispositivo viene aggiornato e segnalato e il modo in cui le app e i servizi interagiscono con il dispositivo e le relative copie shadow.

Per creare un argomento completo, selezionare *`ShadowTopicPrefix`* per il tipo di copia shadow a cui si desidera fare riferimento, sostituire *`thingName`* e *`shadowName`* se applicabile, con i relativi valori corrispondenti, quindi aggiungerlo con lo stub dell'argomento, come illustrato nella tabella seguente. Ricorda che gli argomenti prevedono una distinzione tra lettere maiuscole e minuscole.

Consulta Argomenti copie shadow per ulteriori informazioni sugli argomenti riservati per le copie shadow.

Inizializzazione del dispositivo alla prima connessione a AWS IoT

Una volta effettuata la registrazione con AWS IoT, il dispositivo dovrebbe sottoscrivere questi messaggi MQTT per gli shadows che supporta.

Topic	Significato	Operazione che un dispositivo dovrebbe intraprendere quando riceve questo argomento
ShadowTopicPrefix /delete/accepted	La delete richiesta è stata accettata e l'ombra è stata AWS IoT eliminata.	Le operazioni necessarie per gestire la copia shadow eliminata, ad esempio interrompere la pubblicazione degli aggiornamenti.
ShadowTopicPrefix /delete/rejected	La delete richiesta è stata rifiutata AWS IoT e l'ombra non è stata eliminata. Il corpo del messaggio contiene le informazioni sull'errore.	Rispondere al messaggio di errore nel corpo del messaggio.
ShadowTopicPrefix /get/accepted	La get richiesta è stata accettata da AWS IoT e il corpo del messaggio contiene il documento shadow corrente.	Le operazioni necessarie per elaborare il documento di stato nel corpo del messaggio.
ShadowTopicPrefix /get/rejected	La get richiesta è stata rifiutata da AWS IoT e il corpo del messaggio contiene le informazioni sull'errore.	Rispondere al messaggio di errore nel corpo del messaggio.
ShadowTopicPrefix /update/accepted	La update richiesta è stata accettata da AWS IoT e il corpo del messaggio contiene il documento shadow corrente.	Verificare che i dati aggiornati nel corpo del messaggio corrispondano allo stato del dispositivo.
ShadowTopicPrefix /update/rejected	La update richiesta è stata rifiutata da AWS IoT e il corpo del messaggio contiene le informazioni sull'errore.	Rispondere al messaggio di errore nel corpo del messaggio.

Topic	Significato	Operazione che un dispositivo dovrebbe intraprendere quando riceve questo argomento
ShadowTopicPrefix /update/delta	Il documento ombra è stato aggiornato tramite una richiesta a AWS IoT e il corpo del messaggio contiene le modifiche richieste.	Aggiornare lo stato del dispositivo in modo che corrisponda allo stato desiderato nel corpo del messaggio.
ShadowTopicPrefix /update/documents	Recentemente è stato completato un aggiornamento alla copia shadow e il corpo del messaggio contiene il documento shadow corrente.	Verificare che lo stato aggiornato nel corpo del messaggio corrisponda allo stato del dispositivo.

Dopo aver sottoscritto i messaggi nella tabella precedente per ogni copia shadow, il dispositivo deve verificare se le copie shadow supportate sono già state create pubblicando un argomento `/get` per ogni copia shadow. Se viene ricevuto un messaggio `/get/accepted`, il corpo del messaggio contiene il documento shadow, che il dispositivo può utilizzare per inizializzare il suo stato. Se viene ricevuto un messaggio `/get/rejected`, la copia shadow deve essere creata pubblicando un messaggio `/update` con lo stato corrente del dispositivo.

Supponiamo ad esempio che tu abbia un oggetto `My_IoT_Thing` che non ha copie shadow classiche o con nome. Se ora pubblichi una richiesta `/get` sull'argomento riservato `$aws/things/My_IoT_Thing/shadow/get`, restituisce un errore sull'argomento `$aws/things/My_IoT_Thing/shadow/get/rejected` perché l'oggetto non ha copie shadow. Per risolvere questo errore, prima pubblica un messaggio `/update` utilizzando l'argomento `$aws/things/My_IoT_Thing/shadow/update` con lo stato attuale del dispositivo, ad esempio il seguente payload.

```
{
 "state": {
  "reported": {
   "welcome": "aws-iot",
   "color": "yellow"
  }
```

```
 }
}
```

Ora viene creata una copia shadow classica per l'oggetto e il messaggio viene pubblicato sull'argomento `$aws/things/My_IoT_Thing/shadow/update/accepted`. Se pubblichi sull'argomento `$aws/things/My_IoT_Thing/shadow/get`, restituisce una risposta all'argomento `$aws/things/My_IoT_Thing/shadow/get/accepted` con lo stato del dispositivo.

Per le copie shadow con nome, innanzitutto devi creare la copia shadow con nome o pubblicare un aggiornamento con il nome della copia shadow prima di utilizzare la richiesta get. Ad esempio, per creare una copia shadow con nome `namedShadow1`, prima pubblica le informazioni sullo stato del dispositivo sull'argomento `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/update`. Per recuperare le informazioni sullo stato, utilizza la richiesta `/get` per la copia shadow con nome `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/get`.

Elaborazione dei messaggi mentre il dispositivo è connesso a AWS IoT

Mentre un dispositivo è connesso AWS IoT, può ricevere messaggi /update/delta e dovrebbe mantenere lo stato del dispositivo corrispondente ai cambiamenti nelle sue ombre mediante:

1. Lettura di tutti i messaggi /update/delta ricevuti e sincronizzazione dello stato del dispositivo in modo che corrisponda.
2. Pubblicazione di un messaggio /update con un corpo del messaggio `reported` con lo stato corrente del dispositivo, ogni volta che lo stato del dispositivo cambia.

Quando un dispositivo è connesso, dovrebbe pubblicare questi messaggi quando indicato.

Indicazione	Topic	Carico utile
Lo stato del dispositivo è cambiato.	*ShadowTopicPrefix* `/update`	Un documento shadow con la proprietà `reported`.
Il dispositivo potrebbe non essere sincronizzato con la copia shadow.	*ShadowTopicPrefix* `/get`	(vuoto)
Un'operazione sul dispositivo indica che una copia shadow	*ShadowTopicPrefix* `/delete`	(vuoto)

Indicazione	Topic	Carico utile
non sarà più supportata dal dispositivo, ad esempio quando il dispositivo viene rimosso o sostituito.		

Elaborazione dei messaggi quando il dispositivo si riconnette a AWS IoT

Quando un dispositivo con una o più ombre si connette AWS IoT, dovrebbe sincronizzare il suo stato con quello di tutte le ombre supportate mediante:

1. Lettura di tutti i messaggi /update/delta ricevuti e sincronizzazione dello stato del dispositivo in modo che corrisponda.
2. Pubblicazione di un messaggio /update con un corpo del messaggio `reported` con lo stato corrente del dispositivo.

Utilizzo delle copie shadow in app e servizi

Questa sezione descrive come un'app o un servizio interagisce con il servizio AWS IoT Device Shadow. Questo esempio presuppone che l'app o il servizio interagisca solo con la copia shadow e, attraverso la copia shadow, con il dispositivo. Questo esempio non include alcuna operazione di gestione, ad esempio la creazione o l'eliminazione di copie shadow.

Questo esempio utilizza l'API REST del servizio AWS IoT Device Shadow per interagire con le ombre. A differenza dell'esempio utilizzato inUtilizzo delle copie shadow nei dispositivi, che utilizza un modello di publish/subscribe communications model, this example uses the request/response comunicazione dell'API REST. Ciò significa che l'app o il servizio devono effettuare una richiesta prima di poter ricevere una risposta da AWS IoT. Uno svantaggio di questo modello, tuttavia, è che non supporta le notifiche. Se l'app o il servizio richiedono notifiche tempestive sulle modifiche dello stato del dispositivo, prendi in considerazione i protocolli MQTT o MQTT su WSS, che supportano il modello di publish/subscribe comunicazione, come descritto in. Utilizzo delle copie shadow nei dispositivi

> **Important**
>
> Assicurati che l'uso delle copie shadow da parte dell'app o del servizio sia coerente e supportato dalle implementazioni corrispondenti nei tuoi dispositivi. Considera, ad esempio, come vengono create, aggiornate ed eliminate le copie shadow e come vengono gestiti gli aggiornamenti nel dispositivo e nelle app o nei servizi che accedono alla copia shadow.
> Il progetto deve specificare chiaramente come lo stato del dispositivo viene aggiornato e segnalato e come le app e i servizi interagiscono con il dispositivo e le relative copie shadow.

L'URL dell'API REST per una copia shadow denominata è:

```
https://endpoint/things/thingName/shadow?name=shadowName
```

e per una copia shadow senza nome è:

```
https://endpoint/things/thingName/shadow
```

dove:

endpoint

L'endpoint restituito dal comando CLI è:

```
aws iot describe-endpoint --endpoint-type IOT:Data-ATS
```

thingName

Il nome dell'oggetto a cui appartiene la copia shadow è:

shadowName

Il nome della copia shadow con nome. Questo parametro non viene utilizzato con copie shadow senza nome.

Inizializzazione dell'app o del servizio durante la connessione a AWS IoT

Quando l'app si connette per la prima volta AWS IoT, dovrebbe inviare una richiesta HTTP GET alle URLs ombre che utilizza per ottenere lo stato attuale delle ombre che sta utilizzando. Ciò consente di sincronizzare l'app o il servizio con la copia shadow.

Lo stato di elaborazione cambia mentre l'app o il servizio sono connessi a AWS IoT

Mentre l'app o il servizio è connesso AWS IoT, può interrogare periodicamente lo stato corrente inviando una richiesta HTTP GET sulle URLs ombre che utilizza.

Quando un utente finale interagisce con l'app o il servizio per modificare lo stato del dispositivo, l'app o il servizio può inviare una richiesta HTTP POST alle URLs ombre che utilizza per aggiornare lo `desired` stato dell'ombra. Questa richiesta restituisce la modifica accettata, ma potrebbe essere necessario eseguire il polling della copia shadow effettuando richieste HTTP GET fino a quando il dispositivo non ha aggiornato la cop0ia shadow con il suo nuovo stato.

Rilevamento della connessione di un dispositivo

Per stabilire se un dispositivo è attualmente connesso, includere una proprietà `connected` nel documento shadow e utilizzare un messaggio LWT (Last Will and Testament) MQTT per impostare la proprietà `connected` su `false` se un dispositivo viene disconnesso a causa di un errore.

> **Note**
>
> I messaggi MQTT LWT inviati ad argomenti AWS IoT riservati (argomenti che iniziano con $) vengono ignorati dal servizio AWS IoT Device Shadow. Tuttavia, vengono elaborati dai client sottoscritti e dal motore delle AWS IoT regole, pertanto sarà necessario creare un messaggio LWT da inviare a un argomento non riservato e una regola che ripubblichi il messaggio MQTT LWT come messaggio di aggiornamento shadow nell'argomento di aggiornamento riservato dello shadow,. *`ShadowTopicPrefix`*`/update`

Per inviare un messaggio LWT al servizio Device Shadow

1. Creare una regola che ripubblica il messaggio LWT MQTT sull'argomento riservato. L'esempio seguente è una regola che ascolta i messaggi sull'argomento `my/things/myLightBulb/update` e li ripubblica su `$aws/things/myLightBulb/shadow/update`.

```
{
    "rule": {
    "ruleDisabled": false,
    "sql": "SELECT * FROM 'my/things/myLightBulb/update'",
    "description": "Turn my/things/ into $aws/things/",
```

```
    "actions": [
        {
        "republish": {
            "topic": "$$aws/things/myLightBulb/shadow/update",
            "roleArn": "arn:aws:iam:123456789012:role/aws_iot_republish"
            }
        }
     ]
   }
}
```

2. Quando il dispositivo si connette AWS IoT, registra un messaggio LWT su un argomento non riservato affinché la regola di ripubblicazione lo riconosca. In questo esempio, tale argomento è `my/things/myLightBulb/update` e imposta la proprietà connessa su `false`.

```
{
    "state": {
        "reported": {
            "connected":"false"
        }
    }
}
```

3. Dopo la connessione, il dispositivo pubblica un messaggio sul relativo argomento di aggiornamento shadow, `$aws/things/myLightBulb/shadow/update`, per segnalare lo stato corrente, che include l'impostazione della proprietà `connected` su `true`.

```
{
     "state": {
        "reported": {
            "connected":"true"
        }
    }
}
```

4. Prima della disconnessione, il dispositivo pubblica un messaggio sul relativo argomento di aggiornamento shadow, `$aws/things/myLightBulb/shadow/update`, per segnalare lo stato più recente, che include l'impostazione della proprietà `connected` su `false`.

```
{
    "state": {
        "reported": {
```

```
            "connected":"false"
        }
    }
}
```

5. Se il dispositivo si disconnette a causa di un errore, il broker di messaggi pubblica il AWS IoT messaggio LWT del dispositivo per conto del dispositivo. La regola di ripubblicazione rileva questo messaggio e pubblica il messaggio di aggiornamento shadow per aggiornare la proprietà `connected` del Device Shadow.

> **Note**
>
> A causa della natura asincrona dell'elaborazione della disconnessione, non è garantito che i messaggi LWT vengano inviati in ordine durante la riconnessione. Si consiglia di utilizzare gli eventi del ciclo di vita per migliorare l'accuratezza del rilevamento dello stato di connettività, poiché questi eventi forniscono gli attributi per la gestione degli eventi. out-of-order

Simulazione delle comunicazioni del servizio Device Shadow

In questo argomento viene illustrato come il servizio Device Shadow funge da intermediario e consente a dispositivi e app di utilizzare una copia shadow per aggiornare, archiviare e recuperare lo stato di un dispositivo.

Per dimostrare l'interazione descritta in questo argomento e per approfondirla ulteriormente, avrai bisogno di un sistema Account AWS e di un sistema su cui eseguire. AWS CLI Se non si dispone di questi, è ancora possibile vedere l'interazione negli esempi di codice.

In questo esempio, la AWS IoT console rappresenta il dispositivo. AWS CLI Rappresenta l'app o il servizio che accede al dispositivo tramite l'ombra. L' AWS CLI interfaccia è molto simile all'API con cui un'app potrebbe utilizzare per comunicare AWS IoT. Il dispositivo in questo esempio è una lampadina intelligente e l'app visualizza lo stato della lampadina e può cambiare lo stato della lampadina.

Impostazione della simulazione

Queste procedure inizializzano la simulazione aprendo la console AWS IoT, che simula il dispositivo, e la finestra della riga di comando che simula l'app.

Per configurare l'ambiente di simulazione

1. Avrai bisogno Account AWS di un per eseguire autonomamente gli esempi di questo argomento. Se non ne hai uno Account AWS, creane uno, come descritto inConfigurare Account AWS.
2. Aprire la console AWS IoT e, nel menu a sinistra, scegliere Test per aprire il Client MQTT.
3. In un'altra finestra, aprire una finestra di terminale su un sistema su cui sia installata l' AWS CLI .

Dovrebbero essere aperte due finestre: una con la AWS IoT console nella pagina Test e l'altra con il prompt della riga di comando.

Inizializzare il dispositivo

In questa simulazione, lavoreremo con un oggetto chiamato oggetto e la sua ombra denominata SimShadow1. mySimulatedThing

Creazione di un oggetto e della relativa policy IoT

Per creare un oggetto, nella console AWS IoT :

1. Scegli Gestisci, quindi seleziona Oggetti.
2. Fai clic sul pulsante Crea se gli elementi sono elencati, altrimenti fai clic su Registra una singola cosa per creare una singola cosa. AWS IoT
3. Inserisci il nome `mySimulatedThing`, lascia le altre impostazioni predefinite e quindi fai clic su Successivo.
4. Utilizza la creazione di certificati con un solo clic per generare i certificati che autenticheranno la connessione del dispositivo a AWS IoT. Fare clic su Attiva per attivare il certificato.
5. È possibile allegare la policy `My_IoT_Policy` che darebbe al dispositivo l'autorizzazione per pubblicare e sottoscrivere gli argomenti riservati MQTT. Per passaggi più dettagliati su come creare un AWS IoT oggetto e su come creare questa politica, consultaCrea un oggetto.

Creazione di una copia shadow con nome per un oggetto

Puoi creare una copia shadow con nome per un oggetto pubblicando una richiesta di aggiornamento sull'argomento `$aws/things/mySimulatedThing/shadow/name/simShadow1/update` come descritto di seguito.

Oppure, per creare un'ombra con nome:

1. Nella console AWS IoT , scegli l'oggetto dall'elenco degli oggetti visualizzato e quindi scegli Shadows (Copie shadow).
2. Scegli Aggiungi una copia shadow, inserisci il nome `simShadow1`, quindi scegli Crea per aggiungere la copia shadow.

Effettua la sottoscrizione e pubblica gli argomenti riservati MQTT

Nella console, effettua la sottoscrizione a questi argomenti MQTT. Questi argomenti sono le risposte alle operazioni `get`, `update` e `delete`, in modo che il dispositivo sia pronto a ricevere le risposte dopo la pubblicazione di un'operazione.

Per sottoscrivere un argomento MQTT nel client MQTT

1. In MQTT client (Client MQTT), scegli Subscribe to a topic (Sottoscrivi un argomento).
2. Inserisci gli argomenti `get`, `update` e `delete` a cui vuoi effettuare la sottoscrizione. Copia un argomento alla volta dall'elenco seguente, incollalo nel campo Filtro di argomenti, quindi clicca su Sottoscrivi. Verranno visualizzati gli argomenti sotto Sottoscrizioni.

   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/delete/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/delete/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/delta`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/documents`

   A questo punto, il dispositivo simulato è pronto a ricevere gli argomenti pubblicati da AWS IoT.

Per pubblicare in un argomento MQTT nel client MQTT

Dopo l'inizializzazione e la sottoscrizione agli argomenti di risposta, il dispositivo deve interrogare le copie shadow supportate. Questa simulazione supporta solo un'ombra, l'ombra che supporta un oggetto denominato SimShadow1. mySimulatedThing

Per ottenere lo stato shadow corrente dal client MQTT

1. Nel client MQTT, scegliere Pubblica in un argomento.
2. Sotto Pubblica, inserisci l'argomento seguente ed elimina qualsiasi contenuto dalla finestra del corpo del messaggio sottostante in cui è stato inserito l'argomento da ottenere. È quindi possibile scegliere Pubblicazione nell'argomento per pubblicare la richiesta. `$aws/things/mySimulatedThing/shadow/name/simShadow1/get`.

   Se non hai creato un nome shadow `simShadow1`, riceverai un messaggio nell'argomento `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected`, e il `code` sarà `404`, come in questo esempio in cui la copia shadow non è stata creata, quindi verrà creata in seguito.

   ```
   {
     "code": 404,
     "message": "No shadow exists with name: 'simShadow1'"
   }
   ```

Per creare una copia shadow con lo stato corrente del dispositivo

1. Nel Client MQTT, scegli Pubblica in un argomento e inserisci questo argomento:

   ```
   $aws/things/mySimulatedThing/shadow/name/simShadow1/update
   ```

2. Nella finestra del corpo del messaggio qui sotto, dove è stato inserito l'argomento, inserire questo documento shadow per mostrare che il dispositivo sta segnalando il suo ID e il suo colore corrente in valori RGB. Scegli Pubblica per pubblicare la richiesta.

   ```
   {
     "state": {
       "reported": {
         "ID": "SmartLamp21",
         "ColorRGB": [
           128,
           128,
           128
         ]
       }
     },
     "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
   ```

```
}
```

Se viene visualizzato un messaggio nell'argomento:

- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted`: significa che la copia shadow è stata creata e che il corpo del messaggio contiene il documento shadow corrente.
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected`: esamina l'errore nel corpo del messaggio.
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted`: la copia shadow esiste già e il corpo del messaggio ha lo stato shadow corrente, come in questo esempio. Con questo, è possibile impostare il dispositivo o confermare che corrisponda allo stato della copia shadow.

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        }
```

```
      ]
    }
  },
  "version": 3,
  "timestamp": 1591140517,
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

Inviare un aggiornamento dall'app

Questa sezione utilizza il AWS CLI per dimostrare come un'app può interagire con un'ombra.

Per ottenere lo stato attuale dell'ombra, utilizzare AWS CLI

Nella riga di comando, immettere questo comando:

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 /
dev/stdout
```

Sulle piattaforme Windows è possibile usare `con` al posto di `/dev/stdout`.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1
 con
```

Poiché la copia shadow esiste ed è stata inizializzata dal dispositivo per riflettere lo stato corrente, deve restituire il seguente documento shadow.

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
    "reported": {
      "ID": {
```

```
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        }
      ]
    }
  },
  "version": 3,
  "timestamp": 1591141111
}
```

L'app può utilizzare questa risposta per inizializzare la sua rappresentazione dello stato del dispositivo.

Se l'app aggiorna lo stato, ad esempio quando un utente finale cambia il colore della lampadina intelligente in giallo, l'app invierà un comando update-thing-shadow. Questo comando corrisponde all'API REST `UpdateThingShadow`.

Per aggiornare una copia shadow da un'app

Nella riga di comando, immettere questo comando:

AWS CLI v2.x

```
aws iot-data update-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 \
    --cli-binary-format raw-in-base64-out \
    --payload '{"state":{"desired":{"ColorRGB":
[255,255,0]}},"clientToken":"21b21b21-bfd2-4279-8c65-e2f697ff4fab"}' /dev/stdout
```

AWS CLI v1.x

```
aws iot-data update-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 \
```

```
    --payload '{"state":{"desired":{"ColorRGB":
[255,255,0]}},"clientToken":"21b21b21-bfd2-4279-8c65-e2f697ff4fab"}' /dev/stdout
```

In caso di esito positivo, questo comando deve restituire il seguente documento shadow.

```
{
  "state": {
    "desired": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "desired": {
      "ColorRGB": [
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        }
      ]
    }
  },
  "version": 4,
  "timestamp": 1591141596,
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

Rispondere all'aggiornamento nel dispositivo

Tornando al client MQTT nella AWS console, dovreste vedere i messaggi AWS IoT pubblicati in base al comando di aggiornamento emesso nella sezione precedente.

Per visualizzare i messaggi di aggiornamento nel client MQTT

Nel client MQTT, scegli $ aws/things/mySimulatedThing/shadow/name/simShadow1/update/delta nella colonna Abbonamenti. Se il nome dell'argomento viene troncato, è possibile metterlo in pausa per visualizzare l'argomento completo. Nel registro degli argomenti di questo argomento, dovresti vedere un messaggio `/delta` simile a questo.

```
{
  "version": 4,
  "timestamp": 1591141596,
  "state": {
    "ColorRGB": [
      255,
      255,
      0
    ]
  },
  "metadata": {
    "ColorRGB": [
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      }
    ]
  },
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

Il dispositivo elaborerà il contenuto di questo messaggio per impostare lo stato del dispositivo in modo che corrisponda allo stato `desired` del messaggio.

Dopo che il dispositivo ha aggiornato lo stato in modo che corrisponda allo `desired` stato del messaggio, deve inviare nuovamente il nuovo stato segnalato AWS IoT pubblicando un messaggio di aggiornamento. Questa procedura simula questo nel client MQTT.

Per aggiornare la copia shadow dal dispositivo

1. Nel client MQTT, scegliere Pubblica in un argomento.

2. Nella finestra del corpo del messaggio, nel campo dell'argomento sopra la finestra del corpo del messaggio, inserisci l'argomento della copia shadow seguito dall'operazione `/update`: `$aws/things/mySimulatedThing/shadow/name/simShadow1/update` e nel corpo del messaggio, inserisci questo documento shadow aggiornato, che descrive lo stato corrente del dispositivo. Scegli Pubblica per pubblicare lo stato aggiornato del dispositivo.

```
{
  "state": {
    "reported": {
      "ColorRGB": [255,255,0]
      }
  },
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Se il messaggio è stato ricevuto correttamente da AWS IoT, dovresti vedere una nuova risposta nel registro $ aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted message del client MQTT con lo stato attuale dello shadow, come in questo esempio.

```
{
  "state": {
    "reported": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "reported": {
      "ColorRGB": [
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        }
```

```
      ]
    }
  },
  "version": 5,
  "timestamp": 1591142747,
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Un aggiornamento riuscito allo stato segnalato del dispositivo comporta AWS IoT anche l'invio di una descrizione completa dello stato shadow in un messaggio all'`update/documents`argomento, ad esempio questo corpo del messaggio risultante dall'aggiornamento shadow eseguito dal dispositivo nella procedura precedente.

```
{
  "previous": {
    "state": {
      "desired": {
        "ColorRGB": [
          255,
          255,
          0
        ]
      },
      "reported": {
        "ID": "SmartLamp21",
        "ColorRGB": [
          128,
          128,
          128
        ]
      }
    },
    "metadata": {
      "desired": {
        "ColorRGB": [
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          },
          {
```

```
            "timestamp": 1591141596
          }
        ]
      },
      "reported": {
        "ID": {
          "timestamp": 1591140517
        },
        "ColorRGB": [
          {
            "timestamp": 1591140517
          },
          {
            "timestamp": 1591140517
          },
          {
            "timestamp": 1591140517
          }
        ]
      }
    },
    "version": 4
  },
  "current": {
    "state": {
      "desired": {
        "ColorRGB": [
          255,
          255,
          0
        ]
      },
      "reported": {
        "ID": "SmartLamp21",
        "ColorRGB": [
          255,
          255,
          0
        ]
      }
    },
    "metadata": {
      "desired": {
        "ColorRGB": [
```

```
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          }
        ]
      },
      "reported": {
        "ID": {
          "timestamp": 1591140517
        },
        "ColorRGB": [
          {
            "timestamp": 1591142747
          },
          {
            "timestamp": 1591142747
          },
          {
            "timestamp": 1591142747
          }
        ]
      }
    },
    "version": 5
  },
  "timestamp": 1591142747,
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Osservare l'aggiornamento nell'app

L'app può ora interrogare la copia shadow per lo stato corrente come riportato dal dispositivo.

Per ottenere lo stato attuale dell'ombra, utilizzare il AWS CLI

1. Nella riga di comando, immettere questo comando:

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 /dev/stdout
```

Sulle piattaforme Windows è possibile usare `con` al posto di `/dev/stdout`.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 con
```

2. Poiché la copia shadow è stata appena aggiornata dal dispositivo per riflettere lo stato corrente, dovrebbe restituire il seguente documento shadow.


```
{
  "state": {
    "desired": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    },
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "desired": {
      "ColorRGB": [
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        }
```

```
      ]
    },
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        }
      ]
    }
  },
  "version": 5,
  "timestamp": 1591143269
}
```

Andare oltre la simulazione

Prova l'interazione tra AWS CLI (che rappresenta l'app) e la console (che rappresenta il dispositivo) per modellare la soluzione IoT.

Interazione con le copia shadow

In questo argomento vengono descritti i messaggi associati a ciascuno dei tre metodi che AWS IoT offre per utilizzare le copie shadow. Questi metodi includono i seguenti:

`UPDATE`

Crea una copia shadow se non esiste o aggiorna il contenuto di una copia shadow esistente con le informazioni sullo stato fornite nel corpo del messaggio. AWS IoT registra un timestamp con ogni aggiornamento per indicare l'ultimo aggiornamento dello stato. Quando lo stato dell'ombra cambia, AWS IoT invia `/delta` messaggi a tutti gli abbonati MQTT con la differenza tra gli stati `desired` e. `reported` I dispositivi o le app che ricevono un messaggio `/delta` possono

eseguire operazioni in base alla differenza. Un dispositivo, ad esempio, può aggiornare il proprio stato allo stato desiderato oppure un'app può aggiornare la propria interfaccia utente per visualizzare la modifica dello stato del dispositivo.

`GET`

Recupera un documento shadow corrente che contiene lo stato completo della copia shadow, inclusi i metadati.

`DELETE`

Elimina il Device Shadow e il suo contenuto.

Non è possibile ripristinare un documento del Device Shadow eliminato, ma è possibile creare un nuovo Device Shadow con il nome di un documento Device Shadow eliminato. Se si crea un documento del Device Shadow con lo stesso nome di quello eliminato nelle ultime 48 ore, il nuovo documento riporterà il numero di versione seguente rispetto al documento eliminato. Se un documento del Device Shadow è stato eliminato da più di 48 ore, il numero di versione del nuovo documento del Device Shadow, riportante lo stesso nome, sarà 0.

Supporto dei protocolli

AWS IoT supporta MQTT e un'API REST su protocolli HTTPS per interagire con le ombre. AWS IoT fornisce una serie di argomenti riservati di richiesta e risposta per le azioni di pubblicazione e sottoscrizione di MQTT. I dispositivi e le app devono sottoscrivere gli argomenti di risposta prima di pubblicare su un argomento di richiesta per informazioni sulla modalità di AWS IoT gestione della richiesta. Per ulteriori informazioni, consultare Argomenti MQTT di Device Shadow e API REST del servizio Device Shadow.

Stato di richiesta e segnalazione

Quando si progetta una soluzione IoT utilizzando AWS IoT e shadows, è necessario determinare le app o i dispositivi che richiederanno le modifiche e quelli che le implementeranno. In genere, un dispositivo implementa e segnala le modifiche alla copia shadow e le app e i servizi rispondono e richiedono modifiche nella copia shadow. La soluzione potrebbe essere diversa, ma gli esempi in questo argomento presuppongono che l'app client o il servizio richieda modifiche nella copia shadow e che il dispositivo esegua le modifiche e le riporti alla copia shadow.

Aggiornamento di una copia shadow

L'app o il servizio possono aggiornare lo stato di una copia shadow utilizzando l'API UpdateThingShadow o pubblicando sull'argomento /update. Gli aggiornamenti interessano solo i campi specificati nella richiesta.

Aggiornamento di una copia shadow quando un client richiede una modifica di stato

Quando un client richiede una modifica di stato in una copia shadow utilizzando il protocollo MQTT

1. Il client deve disporre di un documento shadow corrente in modo che possa identificare le proprietà da modificare. Vedere l'operazione /get per capire come ottenere il documento shadow corrente.
2. Il client sottoscrive questi argomenti MQTT:
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
   - `$aws/things/thingName/shadow/name/shadowName/update/delta`
   - `$aws/things/thingName/shadow/name/shadowName/update/documents`
3. Il client pubblica un argomento di richiesta `$aws/things/thingName/shadow/name/shadowName/update` con un documento di stato che contiene lo stato desiderato della copia shadow. Solo le proprietà da modificare devono essere incluse nel documento. Questo è un esempio di documento con lo stato desiderato.

   ```
   {
     "state": {
       "desired": {
         "color": {
           "r": 10
         },
         "engine": "ON"
       }
     }
   }
   ```
4. Se la richiesta di aggiornamento è valida, AWS IoT aggiorna lo stato desiderato nell'ombra e pubblica messaggi sui seguenti argomenti:
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`

- `$aws/things/thingName/shadow/name/shadowName/update/delta`

Il messaggio `/update/accepted` contiene un documento shadow /accepted response state document e il messaggio `/update/delta` contiene un documento shadow /delta response state document.

5. Se la richiesta di aggiornamento non è valida, AWS IoT pubblica un messaggio con l'`$aws/things/thingName/shadow/name/shadowName/update/rejected`argomento con un documento Documenti sulla risposta di errore ombra che descrive l'errore.

Quando un client richiede una modifica di stato in una copia shadow utilizzando l'API

1. Il client chiama l'API `UpdateThingShadow` con un documento di stato Documento sullo stato della richiesta come corpo del messaggio.
2. Se la richiesta era valida, AWS IoT restituisce un codice di risposta di successo HTTP e un documento /accepted response state document shadow come corpo del messaggio di risposta.

   AWS IoT pubblicherà anche un messaggio MQTT sull'`$aws/things/thingName/shadow/name/shadowName/update/delta`argomento con un documento /delta response state document shadow per tutti i dispositivi o i client che lo sottoscrivono.
3. Se la richiesta non era valida, AWS IoT restituisce un codice di risposta all'errore HTTP e Documenti sulla risposta di errore come corpo del messaggio di risposta.

Quando il dispositivo riceve lo stato `/desired` sull'argomento `/update/delta`, apporta le modifiche desiderate nel dispositivo. Invia quindi un messaggio all'argomento `/update` per segnalare lo stato corrente alla copia shadow.

Aggiornamento di una copia shadow quando un dispositivo segnala lo stato corrente

Quando un dispositivo segnala lo stato corrente alla copia shadow utilizzando il protocollo MQTT

1. Il dispositivo deve sottoscrivere questi argomenti MQTT prima di aggiornare la copia shadow:

   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
   - `$aws/things/thingName/shadow/name/shadowName/update/delta`
   - `$aws/things/thingName/shadow/name/shadowName/update/documents`

2. Il dispositivo segnala lo stato corrente pubblicando un messaggio nell'argomento `$aws/things/thingName/shadow/name/shadowName/update` che segnala lo stato corrente, come in questo esempio.

```
{
    "state": {
        "reported" : {
            "color" : { "r" : 10 },
            "engine" : "ON"
        }
    }
}
```

3. Se AWS IoT accetta l'aggiornamento, pubblica un messaggio `$aws/things/thingName/shadow/name/shadowName/update/accepted` sugli argomenti con un documento /accepted response state document ombra.
4. Se la richiesta di aggiornamento non è valida, AWS IoT pubblica un messaggio con l'`$aws/things/thingName/shadow/name/shadowName/update/rejected`argomento con un documento Documenti sulla risposta di errore ombra che descrive l'errore.

Quando un dispositivo segnala lo stato corrente alla copia shadow utilizzando l'API

1. Il dispositivo chiama l'API `UpdateThingShadow` con un documento di stato Documento sullo stato della richiesta come corpo del messaggio.
2. Se la richiesta era valida, AWS IoT aggiorna lo shadow e restituisce un codice di risposta di successo HTTP con un documento /accepted response state document shadow come corpo del messaggio di risposta.

   AWS IoT pubblicherà anche un messaggio MQTT sull'`$aws/things/thingName/shadow/name/shadowName/update/delta`argomento con un documento /delta response state document shadow per tutti i dispositivi o i client che lo sottoscrivono.
3. Se la richiesta non era valida, AWS IoT restituisce un codice di risposta all'errore HTTP e Documenti sulla risposta di errore come corpo del messaggio di risposta.

Blocco ottimistico

Puoi usare la versione del documento sullo stato per assicurarti di aggiornare la versione più recente di un documento di una copia shadow di un dispositivo. Quando fornisci una versione con una

richiesta di aggiornamento, il servizio rifiuta la richiesta con un codice di risposta di conflitto HTTP 409 se la versione corrente del documento sullo stato non corrisponde alla versione fornita. Il codice di risposta ai conflitti può comparire anche su qualsiasi API che modifica `ThingShadow`, incluso `DeleteThingShadow`.

Esempio:

Documento iniziale:

```
{
  "state": {
    "desired": {
      "colors": [
        "RED",
        "GREEN",
        "BLUE"
      ]
    }
  },
  "version": 10
}
```

Aggiornamento: (la versione non corrisponde, questa richiesta verrà rifiutata)

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 9
}
```

Risultato:

```
{
  "code": 409,
  "message": "Version conflict",
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
```

```
}
```

Aggiornamento: (la versione corrisponde, la richiesta verrà accettata)

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 10
}
```

Stato finale:

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 11
}
```

Recupero di un documento di una copia shadow

È possibile recuperare un documento shadow utilizzando l'API GetThingShadow o effettuando la sottoscrizione o la pubblicazione sull'argomento /get. In questo modo viene recuperato un documento shadow completo, incluso qualsiasi delta tra gli stati `desired` e `reported`. La procedura per questa attività è la stessa se la richiesta viene effettuata da un dispositivo o da un client.

Per recuperare un documento shadow utilizzando il protocollo MQTT

1. Il dispositivo o il client devono sottoscrivere questi argomenti MQTT prima di aggiornare la copia shadow:

   - `$aws/things/thingName/shadow/name/shadowName/get/accepted`

- `$aws/things/thingName/shadow/name/shadowName/get/rejected`

2. Il dispositivo o il client pubblica un messaggio nell'argomento `$aws/things/thingName/shadow/name/shadowName/get` con un corpo del messaggio vuoto.
3. Se la richiesta ha esito positivo, AWS IoT pubblica un messaggio nell'`$aws/things/thingName/shadow/name/shadowName/get/accepted`argomento con a /accepted response state document nel corpo del messaggio.
4. Se la richiesta non era valida, AWS IoT pubblica un messaggio sull'`$aws/things/thingName/shadow/name/shadowName/get/rejected`argomento con un Documenti sulla risposta di errore nel corpo del messaggio.

Per recuperare un documento shadow utilizzando un'API REST

1. Il dispositivo o il client chiama l'API `GetThingShadow` con un corpo del messaggio vuoto.
2. Se la richiesta è valida, AWS IoT restituisce un codice di risposta di successo HTTP con un documento /accepted response state document ombra come corpo del messaggio di risposta.
3. Se la richiesta non è valida, AWS IoT restituisce un codice di risposta agli errori HTTP e Documenti sulla risposta di errore come corpo del messaggio di risposta.

Eliminazione di dati shadow

Esistono due modi per eliminare i dati shadow: è possibile eliminare proprietà specifiche nel documento shadow o eliminare completamente la copia shadow.

- Per eliminare proprietà specifiche da una copia shadow, aggiornare la copia shadow. Impostare tuttavia il valore delle proprietà che si desidera eliminare su `null`. I campi con un valore pari a `null` vengono rimossi dal documento shadow.
- Per eliminare l'intera copia shadow, utilizzare l'API DeleteThingShadow o pubblicare nell'argomento /delete.

> **Note**
>
> L'eliminazione di un'ombra non comporta il ripristino del relativo numero di versione su zero in una sola volta. Verrà azzerato dopo 48 ore.

Eliminazione di una proprietà da un documento shadow

Per eliminare una proprietà da una copia shadow utilizzando il protocollo MQTT

1. Il dispositivo o il client deve disporre di un documento shadow corrente in modo che possa identificare le proprietà da modificare. Consulta Recupero di un documento di una copia shadow per informazioni su come ottenere il documento shadow corrente.
2. Il dispositivo o il client sottoscrive i seguenti argomenti MQTT:
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
3. Il dispositivo o il client pubblica un argomento di richiesta `$aws/things/thingName/shadow/name/shadowName/update` con un documento di stato che assegna valori `null` alle proprietà della copia shadow da eliminare. Solo le proprietà da modificare devono essere incluse nel documento. Questo è un esempio di documento che elimina la proprietà `engine`.

   ```
   {
     "state": {
       "desired": {
         "engine": null
       }
     }
   }
   ```
4. Se la richiesta di aggiornamento è valida, AWS IoT elimina le proprietà specificate nell'ombra e pubblica un messaggio con l'`$aws/things/thingName/shadow/name/shadowName/update/accepted`argomento con un documento /accepted response state document ombra nel corpo del messaggio.
5. Se la richiesta di aggiornamento non è valida, AWS IoT pubblica un messaggio con l'`$aws/things/thingName/shadow/name/shadowName/update/rejected`argomento con un documento Documenti sulla risposta di errore shadow che descrive l'errore.

Per eliminare una proprietà da una copia shadow utilizzando l'API REST

1. Il dispositivo o il client chiama l'API `UpdateThingShadow` con un Documento sullo stato della richiesta che assegna valori `null` alle proprietà della copia shadow da eliminare. Includere solo le proprietà che si desidera eliminare nel documento. Questo è un esempio di documento che elimina la proprietà `engine`.

```
{
  "state": {
    "desired": {
      "engine": null
    }
  }
}
```

2. Se la richiesta era valida, AWS IoT restituisce un codice di risposta di successo HTTP e un documento /accepted response state document shadow come corpo del messaggio di risposta.
3. Se la richiesta non era valida, AWS IoT restituisce un codice di risposta all'errore HTTP e Documenti sulla risposta di errore come corpo del messaggio di risposta.

Eliminazione di una copia shadow

Di seguito sono riportate alcune considerazioni sull'eliminazione della copia shadow di un dispositivo.

- L'impostazione dello stato shadow del dispositivo su `null` non elimina la copia shadow. La versione shadow verrà incrementata al successivo aggiornamento.
- Eliminando una copia shadow di un dispositivo non si elimina l'oggetto. Eliminando un oggetto non si elimina la copia shadow del dispositivo corrispondente.
- L'eliminazione di un'ombra non comporta il ripristino del relativo numero di versione su zero in una sola volta. Verrà azzerato dopo 48 ore.

Per eliminare una copia shadow utilizzando il protocollo MQTT

1. Il dispositivo o il client sottoscrive i seguenti argomenti MQTT:
   - `$aws/things/thingName/shadow/name/shadowName/delete/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/delete/rejected`
2. Il dispositivo o il client pubblica un `$aws/things/thingName/shadow/name/shadowName/delete` con un buffer di messaggi vuoto.
3. Se la richiesta di eliminazione è valida, AWS IoT elimina l'ombra e pubblica un messaggio con l'`$aws/things/thingName/shadow/name/shadowName/delete/accepted`argomento e un documento /accepted response state document shadow abbreviato nel corpo del messaggio. Questo è un esempio del messaggio di eliminazione accettato:

```
{
  "version": 4,
  "timestamp": 1591057529
}
```

4. Se la richiesta di aggiornamento non è valida, AWS IoT pubblica un messaggio con l'`$aws/things/thingName/shadow/name/shadowName/delete/rejected`argomento con un documento Documenti sulla risposta di errore ombra che descrive l'errore.

Per eliminare una copia shadow utilizzando l'API REST

1. Il dispositivo o il client chiama l'API `DeleteThingShadow` con un buffer di messaggi vuoto.
2. Se la richiesta era valida, AWS IoT restituisce un codice di risposta di successo HTTP / accepted response state document e un documento /accepted response state document shadow abbreviato nel corpo del messaggio. Questo è un esempio del messaggio di eliminazione accettato:

```
{
  "version": 4,
  "timestamp": 1591057529
}
```

3. Se la richiesta non era valida, AWS IoT restituisce un codice di risposta all'errore HTTP e Documenti sulla risposta di errore come corpo del messaggio di risposta.

API REST del servizio Device Shadow

Un copia shadow espone l'URI seguente per aggiornare le informazioni sullo stato:

```
https://account-specific-prefix-ats.iot.region.amazonaws.com/things/thingName/shadow
```

L'endpoint è specifico per il tuo Account AWS. Per trovare l'endpoint, puoi:

- Utilizzare il comando describe-endpoint da AWS CLI.
- Usa le impostazioni della AWS IoT console. Nelle Impostazioni, l'endpoint è elencato sotto Endpoint personalizzato
- Usa la pagina dei dettagli dell'oggetto della AWS IoT console. Nella console:

1. Apri Gestione e sotto Gestione, scegli Oggetti.
2. Nell'elenco degli oggetti, scegli la cosa per la quale vuoi ottenere l'URI dell'endpoint.
3. Seleziona Device Shadows e scegli la tua shadow. È possibile visualizzare l'URI dell'endpoint nella sezione URL Device Shadow della pagina Dettagli Device Shadow.

Il formato dell'endpoint è il seguente:

```
identifier.iot.region.amazonaws.com
```

L'API shadow REST segue le stesse protocols/port mappature HTTPS descritte in. Protocolli di dispositivo di comunicazione

> **Note**
>
> Per utilizzare APIs, è necessario utilizzare `iotdevicegateway` come nome del servizio per l'autenticazione. Per ulteriori informazioni, vedere Io TData Plane.

Operazioni dell'API

- GetThingShadow
- UpdateThingShadow
- DeleteThingShadow
- ListNamedShadowsForThing

Puoi inoltre utilizzare l'API per creare una copia shadow con nome fornendo `name=shadowName` come parte del parametro di query dell'API.

GetThingShadow

Ottiene la copia shadow per l'oggetto specificato.

Il documento sullo stato della risposta include il delta tra gli stati `desired` e `reported`.

Richiesta

La richiesta include le intestazioni HTTP standard più l'URI seguente:

```
HTTP GET https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

Il parametro della query `name` non è necessario per le copie shadow senza nome (classiche).

Risposta

In caso di esito positivo, la risposta include le intestazioni HTTP standard più il codice e il corpo seguenti:

```
HTTP 200
Response Body: response state document
```

Per ulteriori informazioni, consulta il documento di esempio sullo stato della risposta.

Autorizzazione

Per recuperare una copia shadow, è necessaria una policy che permetta all'intermediario di eseguire l'operazione `iot:GetThingShadow`. Il servizio Device Shadow accetta due forme di autenticazione: Signature Version 4 con credenziali IAM o autenticazione reciproca TLS con un certificato client.

Di seguito è riportato un esempio di policy che permette a un intermediario di recuperare una copia shadow di un dispositivo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:GetThingShadow",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:thing/thing"
            ]
        }
    ]
}
```

UpdateThingShadow

Aggiorna la copia shadow per l'oggetto specificato.

Gli aggiornamenti interessano solo i campi specificati nel documento sullo stato della richiesta. Qualsiasi campo con un valore `null` viene rimosso dalla copia shadow del dispositivo.

Richiesta

La richiesta include le intestazioni HTTP standard più l'URI e il corpo seguenti:

```
HTTP POST https://endpoint/things/thingName/shadow?name=shadowName
Request body: request state document
```

Il parametro della query `name` non è necessario per le copie shadow senza nome (classiche).

Per ulteriori informazioni, consulta il documento di esempio sullo stato della richiesta.

Risposta

In caso di esito positivo, la risposta include le intestazioni HTTP standard più il codice e il corpo seguenti:

```
HTTP 200
Response body: response state document
```

Per ulteriori informazioni, consulta il documento di esempio sullo stato della risposta.

Autorizzazione

Per aggiornare una copia shadow, è necessaria una policy che permetta all'intermediario di eseguire l'operazione `iot:UpdateThingShadow`. Il servizio Device Shadow accetta due forme di autenticazione: Signature Version 4 con credenziali IAM o autenticazione reciproca TLS con un certificato client.

Di seguito è riportato un esempio di policy che permette a un intermediario di aggiornare una copia shadow di un dispositivo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:UpdateThingShadow",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:thing/thing"
```

```
            ]
        }
    ]
}
```

DeleteThingShadow

Elimina la copia shadow per l'oggetto specificato.

Richiesta

La richiesta include le intestazioni HTTP standard più l'URI seguente:

```
HTTP DELETE https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

Il parametro della query `name` non è necessario per le copie shadow senza nome (classiche).

Risposta

In caso di esito positivo, la risposta include le intestazioni HTTP standard più il codice e il corpo seguenti:

```
HTTP 200
Response body: Empty response state document
```

Si noti che l'eliminazione di una copia shadow non reimposta il suo numero di versione su 0.

Autorizzazione

Per eliminare una copia shadow di un dispositivo è necessaria una policy che permetta all'intermediario di eseguire l'operazione `iot:DeleteThingShadow`. Il servizio Device Shadow accetta due forme di autenticazione: Signature Version 4 con credenziali IAM o autenticazione reciproca TLS con un certificato client.

Di seguito è riportato un esempio di policy che permette a un intermediario di eliminare una copia shadow di un dispositivo:

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:DeleteThingShadow",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:thing/thing"
            ]
        }
    ]
}
```

ListNamedShadowsForThing

Elenca le copie shadow per l'oggetto specificato.

Richiesta

La richiesta include le intestazioni HTTP standard più l'URI seguente:

```
HTTP GET /api/things/shadow/ListNamedShadowsForThing/thingName?
nextToken=nextToken&pageSize=pageSize
Request body: (none)
```

nextToken

Token usato per recuperare il successivo set di risultati.

Questo valore viene restituito sui risultati di paging e viene utilizzato nella chiamata che restituisce la pagina successiva.

pageSize

Numero di nomi shadow da restituire in ogni chiamata. Consulta anche `nextToken`.

thingName

Il nome dell'oggetto per cui elencare le copie shadow con nome.

Risposta

In caso di esito positivo, la risposta include le intestazioni HTTP standard più il codice di risposta e un Documento di risposta elenco nomi shadow seguenti:

> **Note**
>
> La copia shadow senza nome (classica) non viene visualizzata in questo elenco. La risposta è una lista vuota se hai solo una copia shadow classica o se il `thingName` non esiste.

```
HTTP 200
Response body: Shadow name list document
```

Autorizzazione

Per elencare una copia shadow di un dispositivo è necessaria una policy che permetta all'intermediario di eseguire l'operazione `iot:ListNamedShadowsForThing`. Il servizio Device Shadow accetta due forme di autenticazione: Signature Version 4 con credenziali IAM o autenticazione reciproca TLS con un certificato client.

Di seguito è riportato un esempio di policy che permette a un chiamante di aggiornare una copia shadow con nome di un oggetto:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:ListNamedShadowsForThing",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:thing/thing"
            ]
        }
    ]
}
```

Argomenti MQTT di Device Shadow

Il servizio Device Shadow usa argomenti MQTT riservati per permettere alle applicazioni e ai dispositivi di ottenere, aggiornare o eliminare le informazioni sullo stato per un dispositivo (copia shadow).

Per la pubblicazione e la sottoscrizione negli argomenti delle copie shadow è richiesta l'autorizzazione basata su argomento. AWS IoT si riserva il diritto di aggiungere nuovi argomenti alla struttura di argomenti esistente. Per questo motivo, è consigliabile evitare le sottoscrizioni con caratteri jolly degli argomenti delle copie shadow. Ad esempio, evitate di abbonarvi ai filtri per argomenti, `$aws/things/thingName/shadow/#` perché il numero di argomenti che corrispondono a questo filtro per argomenti potrebbe aumentare man mano che AWS IoT vengono introdotti nuovi argomenti ombra. Per esempi di messaggi pubblicati in questi argomenti, consulta Interazione con le copia shadow.

Le copie shadow possono essere con nome o senza nome (classiche). Gli argomenti utilizzati da ciascuno differiscono solo nel prefisso dell'argomento. Questa tabella mostra il prefisso dell'argomento utilizzato da ogni tipo di copia shadow.

ShadowTopicPrefix value	Tipo di copia shadow
`$aws/things/` *`thingName`* `/shadow`	Copia shadow senza nome (classica)
`$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`*	Copia shadow con nome

Per creare un argomento completo, selezionare *`ShadowTopicPrefix`* per il tipo di copia shadow a cui si desidera fare riferimento, sostituire *`thingName`* e *`shadowName`* se applicabile, con i relativi valori corrispondenti, quindi aggiungerlo con lo stub dell'argomento, come illustrato nelle sezioni seguenti.

Di seguito sono elencati gli argomenti MQTT usati per l'interazione con le copie shadow.

Argomenti

- /get
- /get/accepted
- /get/rejected
- /update
- /update/delta
- /update/accepted
- /update/documents

- /update/rejected
- /delete
- /delete/accepted
- /delete/rejected

/get

Pubblica un messaggio vuoto in questo argomento per ottenere la copia shadow del dispositivo:

```
ShadowTopicPrefix/get
```

AWS IoT risponde pubblicando su uno o. /get/accepted /get/rejected

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/get"
            ]
        }
    ]
}
```

/get/accepted

AWS IoT pubblica un documento shadow di risposta a questo argomento quando restituisce l'ombra del dispositivo:

```
ShadowTopicPrefix/get/accepted
```

Per ulteriori informazioni, consulta Documenti sullo stato della risposta.

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/get/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/get/accepted"
            ]
        }
    ]
}
```

/get/rejected

AWS IoT pubblica un documento di risposta agli errori su questo argomento quando non può restituire l'ombra del dispositivo:

```
ShadowTopicPrefix/get/rejected
```

Per ulteriori informazioni, consulta Documenti sulla risposta di errore.

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/thingName/
shadow/get/rejected"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/shadow/get/
rejected"
      ]
    }
  ]
}
```

/update

Pubblica un documento sullo stato della richiesta in questo argomento per aggiornare la copia shadow del dispositivo:

```
ShadowTopicPrefix/update
```

Il corpo del messaggio contiene un documento di stato della richiesta parziale.

Un client che tenta di aggiornare lo stato di un dispositivo invierebbe un documento di stato della richiesta JSON con la proprietà `desired` come questa:

```
{
  "state": {
    "desired": {
      "color": "red",
      "power": "on"
    }
  }
}
```

Un dispositivo che aggiorna la sua copia shadow invierebbe un documento di stato della richiesta JSON con la proprietà `reported`, ad esempio:

```
{
  "state": {
    "reported": {
      "color": "red",
      "power": "on"
    }
  }
}
```

AWS IoT risponde pubblicando su o/update/accepted. /update/rejected

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update"
```

```
            ]
        }
    ]
}
```

/update/delta

AWS IoT pubblica un documento sullo stato di risposta a questo argomento quando accetta una modifica per l'ombra del dispositivo e il documento sullo stato di risposta contiene valori `desired` e `reported` stati diversi per:

```
ShadowTopicPrefix/update/delta
```

Il buffer dei messaggi contiene un file /delta response state document.

Dettagli corpo del messaggio

- Un messaggio pubblicato in `update/delta` include solo gli attributi desiderati diversi tra le sezioni `desired` e `reported`. Contiene tutti questi attributi, indipendentemente dal fatto che siano inclusi nel messaggio di aggiornamento corrente o siano già archiviati in AWS IoT. Gli attributi che non presentano differenze tra le sezioni `desired` e `reported` non sono inclusi.
- Se un attributo è nella sezione `reported`, ma non ha un equivalente nella sezione `desired`, non viene incluso.
- Se un attributo è nella sezione `desired`, ma non ha un equivalente nella sezione `reported`, viene incluso.
- Se un attributo viene eliminato dalla sezione `reported`, ma è ancora presente nella sezione `desired`, viene incluso.

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/delta"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/delta"
            ]
        }
    ]
}
```

/update/accepted

AWS IoT pubblica un documento sullo stato della risposta a questo argomento quando accetta una modifica per l'ombra del dispositivo:

```
ShadowTopicPrefix/update/accepted
```

Il buffer dei messaggi contiene un file /accepted response state document.

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
```

```
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/accepted"
            ]
        }
    ]
}
```

/update/documents

AWS IoT pubblica un documento di stato su questo argomento ogni volta che un aggiornamento dell'ombra viene eseguito correttamente:

```
ShadowTopicPrefix/update/documents
```

Il corpo del messaggio contiene un /documents response state document.

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/documents"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/documents"
            ]
        }
    ]
}
```

/update/rejected

AWS IoT pubblica un documento di risposta agli errori su questo argomento quando rifiuta una modifica all'ombra del dispositivo:

```
ShadowTopicPrefix/update/rejected
```

Il corpo del messaggio contiene un Documenti sulla risposta di errore.

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/rejected"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/rejected"
            ]
        }
    ]
}
```

/delete

Per eliminare una copia shadow di un dispositivo, pubblica un messaggio vuoto nell'argomento delete:

```
ShadowTopicPrefix/delete
```

Il contenuto del messaggio viene ignorato.

Si noti che l'eliminazione di una copia shadow non reimposta il suo numero di versione su 0.

AWS IoT risponde pubblicando su o. /delete/accepted /delete/rejected

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/delete"
            ]
        }
    ]
}
```

/delete/accepted

AWS IoT pubblica un messaggio su questo argomento quando viene eliminata l'ombra di un dispositivo:

```
ShadowTopicPrefix/delete/accepted
```

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/delete/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/delete/accepted"
            ]
```

```
        }
    ]
}
```

/delete/rejected

AWS IoT pubblica un documento di risposta agli errori su questo argomento quando non può eliminare l'ombra del dispositivo:

```
ShadowTopicPrefix/delete/rejected
```

Il corpo del messaggio contiene un Documenti sulla risposta di errore.

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/delete/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/delete/rejected"
            ]
        }
    ]
```

```
}
```

Documenti del servizio Device Shadow

Il servizio Device Shadow rispetta tutte le regole della specifica JSON. Valori, oggetti e matrici sono archiviati nel documento della copia shadow del dispositivo.

Indice

- Esempi di documenti shadow
- Proprietà del documento
- Stato delta
- Controllo delle versioni documenti shadow
- I token client nei documenti shadow
- Proprietà documento shadow vuote
- Valori di array nei documenti shadow

Esempi di documenti shadow

Il servizio Device Shadow utilizza questi documenti nelle operazioni UPDATE, GET e DELETE utilizzando l'API REST o i Pub/Sub messaggi MQTT.

Esempi

- Documento sullo stato della richiesta
- Documenti sullo stato della risposta
- Documenti sulla risposta di errore
- Documento di risposta elenco nomi shadow

Documento sullo stato della richiesta

Un documento sullo stato della richiesta ha il seguente formato:

```
{
    "state": {
        "desired": {
            "attribute1": integer2,
```

```
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
            "attributeN": boolean1
        }
    },
    "clientToken": "token",
    "version": version
}
```

- `state` – Gli aggiornamenti interessano solo i campi specificati. In genere, si utilizzerà la proprietà `desired` o `reported`, ma non entrambe nella stessa richiesta.
  - `desired` – Le proprietà dello stato e i valori richiesti per essere aggiornati nel dispositivo.
  - `reported` – Le proprietà dello stato e i valori riportati dal dispositivo.
- `clientToken` – Se usato, consente di abbinare la richiesta e la risposta corrispondente dal token client.
- `version` – Se usato, il servizio Device Shadow elabora l'aggiornamento solo se la versione specificata corrisponde alla versione più recente.

Documenti sullo stato della risposta

I documenti relativi allo stato della risposta hanno il seguente formato a seconda del tipo di risposta.

/accepted response state document

```
{
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        }
    },
    "metadata": {
```

```
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

/delta response state document

```
{
    "state": {
        "attribute1": integer2,
        "attribute2": "string2",
        ...
        "attributeN": boolean2
    },
    "metadata": {
        "attribute1": {
            "timestamp": timestamp
        },
        "attribute2": {
            "timestamp": timestamp
        },
        ...
        "attributeN": {
            "timestamp": timestamp
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

/documents response state document

```
{
  "previous" : {
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
            "attributeN": boolean1
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
```

```
    "version": version-1
  },
  "current": {
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "version": version
```

```
  },
  "timestamp": timestamp,
  "clientToken": "token"
}
```

Proprietà del documento dello stato della risposta

- `previous` – Dopo un aggiornamento riuscito, contiene il `state` dell'oggetto prima dell'aggiornamento.
- `current` – Dopo un aggiornamento riuscito, contiene il `state` dell'oggetto dopo l'aggiornamento.
- `state`
  - `reported` – Presente solo se un oggetto ha segnalato dati nella sezione `reported` e contiene solo campi presenti nel documento sullo stato della richiesta.
  - `desired` – Presente solo se un dispositivo ha segnalato dati nella sezione `desired` e contiene solo campi che erano presenti nel documento sullo stato della richiesta.
- `metadata` - Contiene i timestamp per ogni attributo nelle sezioni `desired` e `reported`, per permettere di determinare quando lo stato è stato aggiornato.
- `timestamp`— Data e ora dell'epoca in cui è stata generata la risposta. AWS IoT
- `clientToken` – Presente solo se è stato usato un token client in fase di pubblicazione di contenuto JSON valido nell'argomento `/update`.
- `version` – Versione corrente del documento per la copia shadow del dispositivo condivisa in AWS IoT. Il valore viene aumentato di uno rispetto alla versione precedente del documento.

Documenti sulla risposta di errore

Un documento sulla risposta agli errori ha il seguente formato:

```
{
    "code": error-code,
    "message": "error-message",
    "timestamp": timestamp,
    "clientToken": "token"
}
```

- `code` – Codice di risposta HTTP che indica il tipo di errore.
- `message` – Messaggio di testo che fornisce ulteriori informazioni.

- `timestamp`— La data e l'ora in cui è stata generata AWS IoT la risposta. Questa proprietà non è presente in tutti i documenti di risposta agli errori.
- `clientToken` – Presente solo se è stato usato un token client in un messaggio pubblicato.

Per ulteriori informazioni, consulta Messaggi di errore Device Shadow.

Documento di risposta elenco nomi shadow

Un documento di risposta elenco nomi shadow ha il seguente formato:

```
{
    "results": [
        "shadowName-1",
        "shadowName-2",
        "shadowName-3",
        "shadowName-n"
    ],
    "nextToken": "nextToken",
    "timestamp": timestamp
}
```

- `results` – L'array di nomi shadow.
- `nextToken` – Il valore del token da utilizzare nelle richieste di paging per ottenere la pagina successiva nella sequenza. Questa proprietà non è presente quando non ci sono più nomi shadow da restituire.
- `timestamp`— La data e l'ora in cui è stata generata la risposta AWS IoT.

Proprietà del documento

Un documento di una copia shadow di un dispositivo ha le proprietà seguenti:

`state`

`desired`

Lo stato desiderato del dispositivo. Le applicazioni possono scrivere su questa parte del documento per aggiornare lo stato di un dispositivo senza doversi connettere direttamente a esso.

`reported`

Lo stato segnalato del dispositivo. I dispositivi scrivono in questa parte del documento per segnalare il proprio nuovo stato. Le app leggono questa parte del documento per determinare l'ultimo stato segnalato del dispositivo.

`metadata`

Informazioni sui dati archiviati nella sezione `state` del documento. Le informazioni includono i timestamp, in base all'epoca (Unix epoch), per ogni attributo nella sezione `state`, che permettono di determinare quando gli attributi sono stati aggiornati.

> **Note**
>
> I metadati non contribuiscono alle dimensioni del documento per le restrizioni dei servizi o i prezzi. Per ulteriori informazioni, consulta la pagina relativa alle restrizioni dei servizi AWS IoT.

`timestamp`

Indica da quando è stato inviato il messaggio AWS IoT. Utilizzando il timestamp nel messaggio e i timestamp per i singoli attributi della sezione `desired` o `reported`, un dispositivo può determinare l'età di una proprietà, anche se il dispositivo non dispone di un orologio interno.

`clientToken`

Stringa univoca del dispositivo che permette di associare le risposte alle richieste in un ambiente MQTT.

`version`

Versione del documento. Ogni volta che il documento viene aggiornato, questo numero di versione viene incrementato. Questa proprietà viene usata per garantire che la versione del documento che viene aggiornata sia la più recente.

Per ulteriori informazioni, consulta Esempi di documenti shadow.

Stato delta

Lo stato delta è un tipo di stato virtuale che contiene la differenza tra gli stati `desired` e `reported`. I campi nella sezione `desired` che non sono presenti nella sezione `reported` sono inclusi nel delta.

I campi nella sezione `reported` che non sono presenti nella sezione `desired` non sono inclusi nel delta. Il delta contiene i metadata e i relativi valori corrispondono ai metadata nel campo `desired`. Ad esempio:

```
{
  "state": {
    "desired": {
      "color": "RED",
      "state": "STOP"
    },
    "reported": {
      "color": "GREEN",
      "engine": "ON"
    },
    "delta": {
      "color": "RED",
      "state": "STOP"
    }
  },
  "metadata": {
    "desired": {
      "color": {
        "timestamp": 12345
      },
      "state": {
        "timestamp": 12345
      }
      },
      "reported": {
        "color": {
          "timestamp": 12345
        },
        "engine": {
          "timestamp": 12345
        }
      },
      "delta": {
        "color": {
          "timestamp": 12345
        },
        "state": {
          "timestamp": 12345
        }
```

```
      }
    },
    "version": 17,
    "timestamp": 123456789
  }
}
```

Quando gli oggetti nidificati differiscono, il delta contiene il percorso fino alla root.

```
{
  "state": {
    "desired": {
      "lights": {
        "color": {
          "r": 255,
          "g": 255,
          "b": 255
        }
      }
    },
    "reported": {
      "lights": {
        "color": {
          "r": 255,
          "g": 0,
          "b": 255
        }
      }
    },
    "delta": {
      "lights": {
        "color": {
          "g": 255
        }
      }
    }
  },
  "version": 18,
  "timestamp": 123456789
}
```

Il servizio Device Shadow calcola il delta scorrendo ogni campo con stato `desired` e confrontandolo con lo stato `reported`.

Le matrici vengono trattate come valori. Se una matrice nella sezione `desired` non corrisponde alla matrice nella sezione `reported`, l'intera matrice della sezione desired viene copiata nel delta.

Controllo delle versioni documenti shadow

Il servizio Device Shadow supporta il controllo delle versioni su ogni messaggio di aggiornamento, sia richiesta che risposta. Ciò significa che con ogni aggiornamento di una copia shadow, la versione del documento JSON viene incrementata. Ciò offre due possibilità:

- Un client può ricevere un errore se tenta di sovrascrivere una copia shadow usando un numero di versione precedente. Il client viene informato che è necessaria una nuova sincronizzazione prima dell'aggiornamento di una copia shadow di un dispositivo.
- Un client può decidere di non agire su un messaggio ricevuto se il messaggio ha una versione più bassa rispetto a quella archiviata dal client.

Un client può ignorare la corrispondenza di versione non includendo una versione nel documento shadow.

I token client nei documenti shadow

È possibile usare un token client con la messaggistica basata su MQTT per verificare che lo stesso token client sia incluso in una richiesta e in una risposta a una richiesta. In questo modo la risposta e la richiesta vengono associate.

> **Note**
>
> Il token client non può superare i 64 byte. Un token client di dimensioni superiori a 64 byte causerà una risposta 400 (Richiesta non valida) e un messaggio di errore Token client non valido.

Proprietà documento shadow vuote

Le proprietà `reported` e `desired` in un documento shadow possono essere vuote o omesse quando non si applicano allo stato shadow corrente. Ad esempio, un documento shadow contiene una proprietà `desired` solo se ha uno stato desiderato. Di seguito è riportato un esempio valido di documento di stato senza proprietà `desired`:

```
{
    "reported" : { "temp": 55 }
}
```

La proprietà `reported` può anche essere vuota, ad esempio se la copia shadow non è stata aggiornata dal dispositivo:

```
{
    "desired" : { "color" : "RED" }
}
```

Se un aggiornamento fa sì che le proprietà `desired` o `reported` diventino nulle, viene rimosso dal documento. Di seguito viene illustrato come rimuovere la proprietà `desired` impostandola su `null`. È possibile eseguire questa operazione quando un dispositivo aggiorna il suo stato, ad esempio.

```
{
    "state": {
        "reported": {
            "color": "red"
        },
        "desired": null
    }
}
```

Un documento shadow può inoltre non avere proprietà `desired` o `reported`, rendendo il documento shadow vuoto. Questo è un esempio di documento shadow vuoto ma valido.

```
{
}
```

Valori di array nei documenti shadow

Le copie shadow supportano le matrici, ma le trattano come normali valori, pertanto un aggiornamento di una matrice sostituisce l'intera matrice. Non è possibile aggiornare una parte di una matrice.

Stato iniziale:

```
{
    "desired" : { "colors" : ["RED", "GREEN", "BLUE" ] }
```

```
}
```

Aggiornamento:

```
{
    "desired" : { "colors" : ["RED"] }
}
```

Stato finale:

```
{
    "desired" : { "colors" : ["RED"] }
}
```

Le matrici non possono contenere valori null. La matrice seguente, ad esempio, non è valida e verrà rifiutata.

```
{
    "desired" : {
        "colors" : [ null, "RED", "GREEN" ]
    }
}
```

Messaggi di errore Device Shadow

Il servizio Device Shadow pubblica un messaggio di errore nell'argomento di errore (tramite MQTT) quando un tentativo di modificare il documento sullo stato ha esito negativo. Questo messaggio viene inviato solo in risposta a una richiesta di pubblicazione in uno degli argomenti `$aws` riservati. Se il client aggiorna il documento usando l'API REST, riceve il codice di errore HTTP come parte della risposta e non viene inviato alcun messaggio di errore MQTT.

Codice di errore HTTP	Messaggi di errore
400 (Richiesta non valida)	• JSON non valido • Nodo richiesto mancante: stato • Il nodo di stato deve essere un oggetto • Il nodo desiderato deve essere un oggetto

Codice di errore HTTP	Messaggi di errore
	• Il nodo segnalato deve essere un oggetto • Versione non valida • Token client non valido **Note** Un token client di dimensioni superiori a 64 byte genererà questo tipo di risposta. • JSON contiene troppi livelli di nidificazione; il numero massimo è 6 • Lo stato contiene un nodo non valido
401 (Non autorizzato)	• Non autorizzato
403 (Accesso negato)	• Accesso negato
404 (Non trovato)	• Oggetto non trovato • Non esiste alcuna ombra con nome: *shadowName*
409 (Conflitto)	• Conflitto di versioni
413 (Payload troppo grande)	• Il payload supera le dimensioni massime permesse
415 (Tipo di supporto non supportato)	• Codifica non supportata; la codifica supportata è UTF-8
429 (Troppe richieste)	• Il servizio Device Shadow genera questo messaggio di errore quando ci sono più di 10 richieste in transito su una singola connessione. Una richiesta in-flight è una richiesta in corso che è stata avviata ma non ancora completata.
500 (Errore interno del server)	• Errore interno del servizio

AWS IoT Device Management Catalogo dei pacchetti software

Con AWS IoT Device Management Software Package Catalog, è possibile mantenere un inventario dei pacchetti software e delle relative versioni. È possibile associare le versioni dei pacchetti a singoli elementi e gruppi di oggetti AWS IoT dinamici e distribuirli tramite processi o AWS IoT processi interni.

Un pacchetto software contiene una o più versioni del pacchetto, ovvero una raccolta di file che possono essere distribuiti come una singola unità. Le versioni del pacchetto possono contenere firmware, aggiornamenti del sistema operativo, applicazioni per dispositivi, configurazioni e patch di sicurezza. Con l'evolversi del software nel tempo, puoi creare una nuova versione del pacchetto e distribuirla nel parco istanze.

L'hub del pacchetto AWS IoT software si trova all'interno. AWS IoT Core Puoi utilizzare l'hub per registrare e gestire centralmente l'inventario e i metadati del pacchetto software. Ciò consente di creare un catalogo di pacchetti software e delle relative versioni. Puoi scegliere di raggruppare i dispositivi in base ai pacchetti software e alle versioni dei pacchetti distribuiti sul dispositivo. Questa funzionalità offre la possibilità di mantenere l'inventario dei pacchetti lato dispositivo come una copia shadow con nome, associare e raggruppare i dispositivi in base alle versioni e visualizzare la distribuzione delle versioni del pacchetto nel parco istanze utilizzando parametri del parco istanze.

Se disponi di un sistema di implementazione del software interno, puoi continuare a utilizzare tale processo per distribuire le versioni dei pacchetti. Se non hai definito un processo di implementazione o se lo preferisci, è consigliabile utilizzare i processi AWS IoT per utilizzare le funzionalità del Software Package Catalog. Per ulteriori informazioni, vedere Preparazione dei AWS IoT lavori.

Questo capitolo contiene le sezioni seguenti:

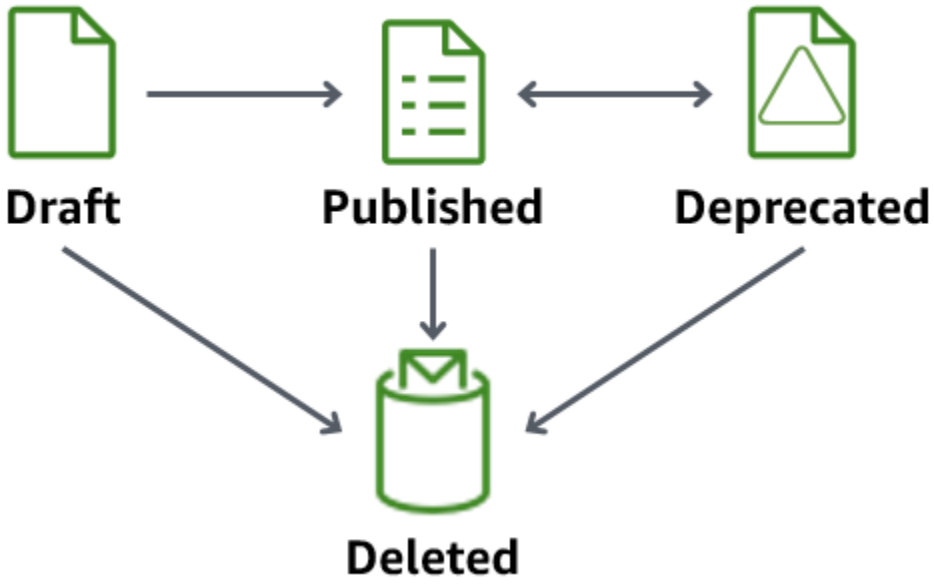
- Preparazione all'uso di Software Package Catalog
- Preparazione della sicurezza
- Preparazione dell'indicizzazione del parco istanze
- Preparazione dei lavori AWS IoT
- Nozioni di base su Software Package Catalog

Preparazione all'uso di Software Package Catalog

La sezione seguente fornisce una panoramica del ciclo di vita delle versioni del pacchetto e informazioni per l'utilizzo del Software AWS IoT Device Management Package Catalog.

Ciclo di vita della versione del pacchetto

Una versione del pacchetto può evolvere attraverso i seguenti stati del ciclo di vita: `draft`, `published` e `deprecated`. Lo stato può anche essere `deleted`.



- Draft

  Quando si crea una versione del pacchetto, questa si trova in uno `draft` stato. Questo stato indica che il pacchetto software è in fase di preparazione o è incompleto.

  Mentre la versione del pacchetto è in questo stato, non è possibile distribuirla. È possibile modificare la descrizione, gli attributi e i tag della versione del pacchetto.

  Puoi trasferire una versione del pacchetto che si trova nello `draft` stato `published` o lo è `deleted` utilizzando la console o emettendo operazioni tramite l'UpdatePackageVersionDeletePackageVersionAPI.

- Published

  Quando la versione del pacchetto è pronta per la distribuzione, trasferisci la versione del pacchetto a uno `published` stato. In questo stato, puoi scegliere di identificare la versione del pacchetto

come versione predefinita modificando il pacchetto software nella console o tramite l'operazione UpdatePackageAPI. In questo stato, puoi modificare solo la descrizione e i tag.

Puoi trasferire una versione del pacchetto che si trova nello `published` stato `deprecated` o lo è `deleted` utilizzando la console o eseguendo operazioni DeletePackageVersiontramite l'UpdatePackageVersionAPI.

- Deprecated

  Se è disponibile una nuova versione del pacchetto, puoi eseguire la transizione delle versioni precedenti del pacchetto a `deprecated`. Puoi comunque distribuire lavori con una versione del pacchetto obsoleta. È inoltre possibile assegnare un nome a una versione obsoleta del pacchetto come versione predefinita e modificare solo la descrizione e i tag.

  Prendi in considerazione la possibilità di passare a `deprecated` una versione del pacchetto una versione obsoleta, ma hai ancora dispositivi sul campo che utilizzano la versione precedente o devi mantenerla a causa della dipendenza in fase di esecuzione.

  Puoi trasferire una versione del pacchetto che si trova nello `deprecated` stato `published` o lo è utilizzando la console oppure eseguendo le operazioni o l'API. `deleted` UpdatePackageVersionDeletePackageVersion

- Eliminato

  Quando non intendi più utilizzare una versione del pacchetto, puoi eliminarla utilizzando la console o eseguendo l'operazione API. DeletePackageVersion

  > **Note**
  >
  > Se elimini una versione del pacchetto mentre sono presenti processi in sospeso che fanno riferimento ad essa, riceverai un messaggio di errore quando il processo termina e tenta di aggiornare la copia shadow con nome.
  > Se la versione del pacchetto software che desideri eliminare è denominata come la versione del pacchetto predefinita, devi aggiornare innanzitutto il pacchetto specificando un'altra versione come predefinita o lasciare il campo senza nome. È possibile farlo utilizzando la console o l'operazione UpdatePackageVersionAPI. (Per rimuovere qualsiasi versione denominata del pacchetto come predefinita, imposta il unsetDefaultVersionparametro su true quando esegui l'operazione UpdatePackageAPI).

Se elimini un pacchetto software tramite la console, vengono eliminate tutte le versioni del pacchetto associate, a meno che una non sia denominata come la versione predefinita.

Convenzioni di denominazione della versione del pacchetto

Quando assegni un nome alle versioni dei pacchetti, è importante pianificare e applicare una strategia di denominazione logica affinché tutti siano in grado di identificare facilmente la versione del pacchetto più recente e la progressione della versione. Durante la creazione della versione del pacchetto, devi fornire un nome della versione, ma la strategia e il formato dipendono in gran parte dal business case.

Come procedura ottimale, consigliamo di utilizzare il formato Semantic Versioning SemVer. Ad esempio, `1.2.3` dove `1` è la versione principale per le modifiche funzionalmente incompatibili, `2` è la versione principale per le modifiche funzionalmente compatibili e `3` è la versione patch (per correzioni di bug). Per ulteriori informazioni, consulta Controllo delle versioni semantico 2.0.0. Per ulteriori informazioni sui requisiti relativi al nome della versione del pacchetto, consulta VersionName nella guida di riferimento dell' AWS IoT API.

Versione predefinita

L'impostazione di una versione come predefinita è facoltativa. Puoi aggiungere o rimuovere versioni dei pacchetti predefinite. Inoltre, puoi distribuire una versione del pacchetto che non è specificata come versione predefinita.

Quando crei una versione del pacchetto, questa assume uno stato `draft` e non può essere denominata come la versione predefinita finché non esegui la transizione della versione del pacchetto a published. Software Package Catalog non seleziona automaticamente una versione come predefinita né aggiorna una versione più recente del pacchetto come quella predefinita. È necessario assegnare intenzionalmente un nome alla versione del pacchetto scelta tramite la console o eseguendo l'operazione API. UpdatePackageVersion

Attributi della versione

Gli attributi della versione e i relativi valori contengono informazioni importanti sulle versioni dei pacchetti. Si consiglia di definire attributi di uso generico per un pacchetto o una versione del pacchetto. Ad esempio, potrebbe essere necessario creare una coppia nome-valore per piattaforma, architettura, sistema operativo, data di rilascio, autore o URL Amazon S3.

Quando crei un AWS IoT lavoro con un documento di lavoro, puoi anche scegliere di utilizzare una variabile di sostituzione (`$parameter`) che si riferisce al valore di un attributo. Per ulteriori informazioni, vedere Preparazione AWS IoT dei lavori.

Gli attributi di versione utilizzati nelle versioni dei pacchetti non verranno aggiunti automaticamente allo shadow denominato riservato e non possono essere indicizzati o interrogati direttamente tramite Fleet Indexing. Per indicizzare o interrogare gli attributi della versione del pacchetto tramite Fleet Indexing, è possibile compilare l'attributo version nell'ombra denominata riservata.

Si consiglia che il parametro dell'attributo version nella cartella riservata named shadow acquisisca le proprietà riportate dal dispositivo, come il sistema operativo e l'ora di installazione. Possono anche essere indicizzati e interrogati tramite Fleet Indexing.

Gli attributi di versione non sono necessari per seguire una convenzione di denominazione specifica. Puoi creare coppie nome-valore per soddisfare le esigenze aziendali. La dimensione combinata di tutti gli attributi di una versione del pacchetto è limitata a 3 KB. Per ulteriori informazioni, vedere Limiti del pacchetto software e delle versioni dei pacchetti del Software Package Catalog.

Utilizzo di tutti gli attributi in un documento di lavoro

È possibile aggiungere automaticamente tutti gli attributi della versione del pacchetto alla distribuzione del processo per dispositivi selezionati. Per utilizzare automaticamente tutti gli attributi della versione del pacchetto a livello di codice in un comando API o CLI, fate riferimento al seguente esempio di documento di lavoro:

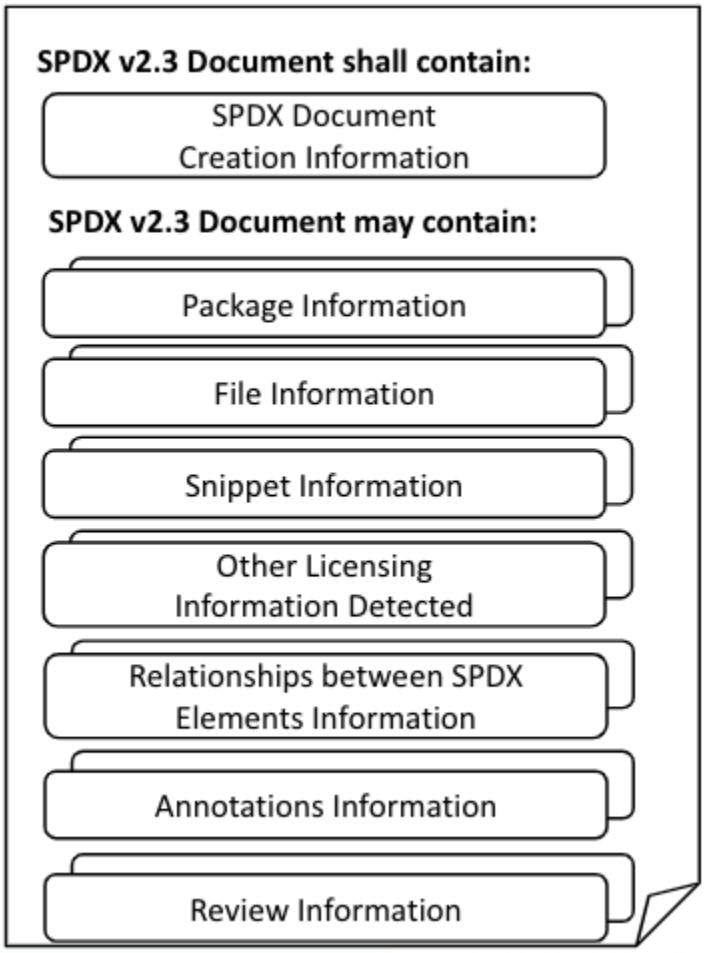
```
"TestPackage": "${aws:iot:package:TestPackage:version:PackageVersion:attributes}"
```

Elenco degli elementi software

La distinta base del software (SBOM) fornisce un archivio centrale per tutti gli aspetti del pacchetto software. Oltre a memorizzare i pacchetti software e le versioni dei pacchetti, è possibile archiviare la distinta base del software (SBOM) associata a ciascuna versione del AWS IoT Device Management pacchetto nel Software Package Catalog. Un pacchetto software contiene una o più versioni del pacchetto e ogni versione del pacchetto è composta da uno o più componenti. Ciascuno di questi componenti che supportano la composizione di una versione specifica del pacchetto può essere descritto e catalogato utilizzando una distinta dei materiali del software. Gli standard di settore per la lista dei materiali software supportati sono SPDX e CyclonedX. Quando un SBOM viene creato per la prima volta, viene sottoposto a convalida rispetto al formato standard di settore SPDX e CyclonedX.

Per ulteriori informazioni su SPDX, vedere System Package Data Exchange. Per ulteriori informazioni su CyclonedX, vedere CyclonedX.

La lista dei materiali del software descrive tutti gli aspetti dei componenti di una versione specifica del pacchetto, come le informazioni sul pacchetto, le informazioni sui file e altri metadati pertinenti. Vedi l'esempio seguente di struttura documentale di una distinta base del software in formato SPDX:



Vantaggi della lista dei materiali del software

Uno dei principali vantaggi dell'aggiunta della distinta base del software per una versione del pacchetto in Software Package Catalog è la gestione delle vulnerabilità.

Gestione delle vulnerabilità

La valutazione e la mitigazione della vulnerabilità agli apparenti rischi di sicurezza dei componenti software rimangono fondamentali per proteggere l'integrità del parco dispositivi. Con l'aggiunta della lista dei materiali software archiviata nel Software Package Catalog per ogni versione del pacchetto, è possibile individuare in modo proattivo le lacune nella sicurezza conoscendo quali dispositivi sono

a rischio in base alla versione del pacchetto e alla SBOM utilizzando la propria soluzione di gestione delle vulnerabilità interna. Puoi implementare correzioni ai dispositivi interessati e proteggere la tua flotta di dispositivi.

Software, distinta dei materiali, archiviazione

La distinta base del software (SBOM) per ogni versione del pacchetto software viene archiviata in un bucket Amazon S3 utilizzando la funzionalità di controllo delle versioni di Amazon S3. Il bucket Amazon S3 che memorizza la SBOM deve trovarsi nella stessa regione in cui è stata creata la versione del pacchetto. Un bucket Amazon S3 che utilizza la funzionalità di controllo delle versioni mantiene più varianti di un oggetto nello stesso bucket. Per ulteriori informazioni sull'utilizzo del controllo delle versioni in un bucket Amazon S3, consulta Uso del controllo delle versioni nei bucket Amazon S3.

> **Note**
>
> A ogni versione del pacchetto software possono essere allegati più file SBOM, ma i file SBOM devono essere archiviati in un unico file di archivio zip.

La chiave Amazon S3 e l'ID di versione specifici del bucket vengono utilizzati per identificare in modo univoco ogni versione di una distinta base software per una versione del pacchetto.

> **Note**
>
> Per una versione del pacchetto con un singolo file SBOM, puoi archiviare quel file SBOM nel tuo bucket Amazon S3 come file di archivio zip.
> Per una versione del pacchetto con più file SBOM, è necessario inserire tutti i file SBOM in un unico file di archivio zip e quindi archiviare tale file di archivio zip nel bucket Amazon S3.
> Tutti i file SBOM archiviati in un unico file di archivio zip in entrambi gli scenari sono formattati come file SPDX o CyclonedX .json.

Politica sulle autorizzazioni

Per AWS IoT agire come principale specificato per accedere ai file di archivio zip SBOM archiviati nel bucket Amazon S3, è necessaria una politica di autorizzazioni basata sulle risorse. Fai riferimento al seguente esempio per la corretta politica di autorizzazioni basata sulle risorse:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "iot.amazonaws.com"
                ]
        },
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::bucketName/*"
        }
    ]
}
```

Per ulteriori informazioni sulle politiche di autorizzazione basate sulle risorse, vedere AWS IoT politiche basate sulle risorse

Aggiornamento della SBOM

È possibile aggiornare la distinta base del software tutte le volte che è necessario per proteggere e potenziare il parco dispositivi. Ogni volta che la distinta base del software viene aggiornata nel tuo bucket Amazon S3, l'ID della versione cambia e devi associare il nuovo URL del bucket Amazon S3 alla versione del pacchetto software appropriata. Vedrai il nuovo ID di versione nella colonna ID della versione di Amazon S3 Object nella pagina della versione del pacchetto nel. Console di gestione AWS Inoltre, puoi utilizzare l'operazione API `GetPackageVersion` o il comando CLI `get-package-version` per visualizzare l'ID della nuova versione.

> **Note**
>
> L'aggiornamento della distinta base del software, che comporterà un nuovo ID di versione, non comporterà la creazione di una nuova versione del pacchetto.

Per ulteriori informazioni sulle chiavi oggetto di Amazon S3, consulta Creazione di nomi di chiavi di oggetto.

Abilitare l'indicizzazione del AWS IoT parco veicoli

L'abilitazione dell'indicizzazione AWS IoT della flotta è un requisito per utilizzare Software AWS IoT Device Management Package Catalog. Per sfruttare l'indicizzazione AWS IoT della flotta con Software Package Catalog, imposta il nome riservato shadow (`$package`) come origine dati per ogni dispositivo su cui desideri indicizzare e raccogli le metriche. Per ulteriori informazioni sulle ombre denominate riservate, consulta. Copia shadow con nome riservata

L'indicizzazione del parco veicoli fornisce un supporto che consente di raggruppare gli AWS IoT elementi tramite gruppi di oggetti dinamici filtrati in base alla versione del pacchetto software. Ad esempio, l'indicizzazione del parco istanze può identificare gli oggetti in cui è installata o non è installata una versione del pacchetto specifica, che non dispongono di versioni dei pacchetti installate o che corrispondono a coppie nome-valore specifiche. Infine, l'indicizzazione della flotta fornisce metriche standard e personalizzate che puoi utilizzare per ottenere informazioni sullo stato del tuo parco dispositivi. Per ulteriori informazioni, consulta Preparazione dell'indicizzazione del parco istanze.

> **Note**
>
> L'abilitazione dell'indicizzazione del parco istanze per Software Package Catalog comporta costi di servizio standard. Per ulteriori informazioni, consulta Prezzi di AWS IoT Device Management.

Copia shadow con nome riservata

La copia shadow con nome riservata, `$package`, riflette lo stato dei pacchetti software installati e delle versioni dei pacchetti del dispositivo. L'indicizzazione del parco istanze utilizza la copia shadow con nome riservata come un'origine dati per creare parametri standard e personalizzati in modo da poter eseguire query sullo stato del parco istanze. Per ulteriori informazioni, consulta Preparazione dell'indicizzazione del parco istanze.

Una copia shadow con nome riservata è simile a una copia shadow con nome, con la differenza che il suo nome è predefinito e non può essere modificato. Inoltre, la copia shadow con nome riservata non viene aggiornata con i metadati e utilizza solo le parole chiave `version` e `attributes`.

Le richieste di aggiornamento che includono altre parole chiave, ad esempio`description`, riceveranno una risposta di errore nell'argomento. `rejected` Per ulteriori informazioni, consulta Messaggi di errore Device Shadow.

Può essere creato quando si crea un AWS IoT oggetto tramite la console, quando un AWS IoT processo completa e aggiorna correttamente lo shadow e se si esegue l'operazione UpdateThingShadowAPI. Per ulteriori informazioni, consulta UpdateThingShadowla guida per gli AWS IoT Core sviluppatori.

> **Note**
>
> L'indicizzazione della copia shadow con nome riservata non conta ai fini del numero di copie shadow con nome che possono essere indicizzate dall'indicizzazione del parco istanze. Per ulteriori informazioni, consulta Limiti e quote per l'indicizzazione del parco istanze di AWS IoT Device Management. Inoltre, se si sceglie di fare in modo che i AWS IoT job aggiornino il named shadow riservato quando un job viene completato con successo, la chiamata API viene conteggiata per le operazioni di Device Shadow e del registro e può comportare un costo. Per ulteriori informazioni, consulta i limiti e le quote dei AWS IoT Device Management lavori e il tipo di dati dell'IndexingFilterAPI.

Struttura della copia shadow **$package**

La copia shadow con nome riservata contiene quanto segue:

```
{
    "state": {
        "reported": {
            "<packageName>": {
                "version": "",
                "attributes": {
                }
            }
        }
    },
    "version" : 1
    "timestamp" : 1672531201
}
```

Le proprietà della copia shadow vengono aggiornate con le seguenti informazioni:

- <packageName>: il nome del pacchetto software installato, che viene aggiornato con il parametro packageName.

- `version`: il nome della versione del pacchetto installato, che viene aggiornato con il parametro packageName.
- `attributes`: metadati opzionali memorizzati dal dispositivo e indicizzati dall'indicizzazione Fleet. Ciò consente ai clienti di interrogare i propri indici in base ai dati memorizzati.
- `version`: il numero di versione della copia shadow. Viene incrementato automaticamente ogni volta che la copia shadow viene aggiornata e inizia da 1.
- `timestamp`: indica quando la copia shadow è stata aggiornata l'ultima volta e viene registrato nell'ora Unix.

Per ulteriori informazioni sul formato e il comportamento di una copia shadow con nome, consultare Ordine dei messaggi Servizio AWS IoT Device Shadow.

Eliminazione di un pacchetto software e delle relative versioni del pacchetto

Prima di eliminare un pacchetto software, effettua le seguenti operazioni:

- Conferma che il pacchetto e le relative versioni non vengano distribuiti attivamente.
- Elimina per prima cosa tutte le versioni associate. Se una delle versioni è designata come la versione predefinita, devi rimuovere la versione predefinita con nome dal pacchetto. Poiché la designazione di una versione predefinita è opzionale, la rimozione non causa alcun conflitto. Per rimuovere la versione predefinita dal pacchetto software, modifica il pacchetto tramite la console o utilizza l'operazione UpdatePackageVersionAPI.

Finché non esiste una versione del pacchetto predefinita con nome, puoi utilizzare la console per eliminare un pacchetto software. Verranno eliminate anche tutte le versioni del pacchetto. Se utilizzi una chiamata API per eliminare i pacchetti software, devi eliminare prima le versioni dei pacchetti e quindi il pacchetto software.

Preparazione della sicurezza

Questa sezione descrive i principali requisiti di sicurezza per AWS IoT Device Management Software Package Catalog.

Autenticazione basata sulle risorse

Software Package Catalog utilizza l'autorizzazione basata sulle risorse per fornire maggiore sicurezza durante l'aggiornamento del software sul parco istanze. Ciò significa che è necessario creare una

policy AWS Identity and Access Management (IAM) che conceda i diritti di esecuzione`create`, `read` `update``delete`, e `list` azioni per i pacchetti software e le versioni dei pacchetti e fare riferimento ai pacchetti software e alle versioni dei pacchetti specifici che si desidera distribuire nella sezione. `Resources` Queste autorizzazioni sono necessarie anche per aggiornare la copia shadow con nome riservata. Puoi fare riferimento ai pacchetti software e alle versioni dei pacchetti includendo un nome della risorsa Amazon (ARN) per ciascuna entità.

> **Note**
>
> Se intendi che la policy conceda i diritti per le chiamate API delle versioni del pacchetto (come CreatePackageVersionUpdatePackageVersion,, DeletePackageVersion), devi includere ARNs nella policy sia il pacchetto software che la versione del pacchetto. Se intendi che la politica conceda diritti per le chiamate API dei pacchetti software (ad esempio CreatePackageUpdatePackage, e DeletePackage), devi includere solo il pacchetto software ARN nella politica.

Strutturate il pacchetto software e la versione del pacchetto ARNs come segue:

- Pacchetto software: `arn:aws:iot:<region>:<accountID>:package/<packageName>/package`
- Versione del pacchetto: `arn:aws:iot:<region>:<accountID>:package/<packageName>/version/<versionName>`

> **Note**
>
> Sono disponibili altri diritti correlati che potrebbe essere necessario includere in questa policy. Ad esempio, potrebbe essere necessario includere un ARN per `job`, `thinggroup` e `jobtemplate`. Per ulteriori informazioni e un elenco completo delle opzioni di policy, consulta Proteggere utenti e dispositivi con AWS IoT Jobs.

Ad esempio, se disponi di un pacchetto software e una versione del pacchetto denominati come segue:

- AWS IoT cosa: `myThing`
- Nome pacchetto: `samplePackage`

- Versione `1.0.0`

L'aspetto della policy potrebbe essere simile all'esempio seguente:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:createPackage",
                "iot:createPackageVersion",
                "iot:updatePackage",
                "iot:updatePackageVersion"
            ],
            "Resource": [
               "arn:aws:iot:us-east-1:111122223333:package/samplePackage",
               "arn:aws:iot:us-east-1:111122223333:package/samplePackage/
version/1.0.0"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow"
            ],
            "Resource": "arn:aws:iot:us-east-1:111122223333:thing/myThing/$package"
        }
    ]
}
```

AWS IoT Job rights per distribuire le versioni dei pacchetti

Per motivi di sicurezza, è importante concedere diritti per distribuire pacchetti e versioni dei pacchetti e assegnare un nome ai pacchetti e alle versioni dei pacchetti specifici che sono autorizzati a distribuire. A questo scopo, puoi creare un ruolo e una policy IAM che concedono l'autorizzazione per distribuire processi con versioni dei pacchetti. La policy deve specificare le versioni dei pacchetti di destinazione come una risorsa.

Policy IAM

La policy IAM concede il diritto per creare un processo che include il pacchetto e la versione denominati nella sezione `Resource`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:CreateJob",
                "iot:CreateJobTemplate"
            ],
            "Resource":[
                "arn:aws:iot:*:111122223333:job/<jobId>",
                "arn:aws:iot:*:111122223333:thing/<thingName>/$package",
                "arn:aws:iot:*:111122223333:thinggroup/<thingGroupName>",
                "arn:aws:iot:*:111122223333:jobtemplate/<jobTemplateName>",
                "arn:aws:iot:*:111122223333:package/<packageName>/
version/<versionName>"
            ]
        }
    ]
}
```

> **Note**
>
> Se si desidera distribuire un processo che disinstalla un pacchetto software e una versione del pacchetto, `$null` è necessario autorizzare un ARN in cui si trova la versione del pacchetto, come nel seguente:

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

AWS IoT Diritti di lavoro per aggiornare l'ombra denominata riservata

Per consentire ai processi di aggiornare la copia shadow con nome riservata quando il processo viene completato correttamente, devi creare un ruolo e una policy IAM. Sono disponibili due modi per

eseguire questa operazione nella console AWS IoT . Il primo è quando crei un pacchetto software nella console. Se viene visualizzata una finestra di dialogo Abilita dipendenze per la gestione dei pacchetti, puoi scegliere di utilizzare un ruolo esistente o crearne uno nuovo. Oppure, nella console AWS IoT , scegli Impostazioni, seleziona Gestisci indicizzazione, quindi Gestisci indicizzazione per pacchetti e versioni dei dispositivi.

> **Note**
>
> Se si sceglie di fare in modo che il servizio AWS IoT Job aggiorni lo shadow denominato riservato quando un processo viene completato correttamente, la chiamata API viene conteggiata per le operazioni di Device Shadow e del registro e può comportare un costo. Per ulteriori informazioni, consultare Prezzi di AWS IoT Core.

Quando utilizzi l'opzione Crea ruolo, il nome del ruolo generato inizia con `aws-iot-role-update-shadows` e contiene le seguenti policy:

Impostazione di un ruolo

Permissions

La policy delle autorizzazioni concede i diritti per eseguire query sulla shadow oggetto e aggiornarla. Il parametro `$package` nell'ARN della risorsa ha come target la copia shadow con nome riservata.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:DescribeEndpoint",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:111122223333:thing/<thingName>/$package"
```

```
            ]
        }
    ]
}
```

Relazione di attendibilità

Oltre alla policy delle autorizzazioni, il ruolo richiede una relazione di attendibilità con AWS IoT Core in modo che l'entità possa assumere il ruolo e aggiornare la copia shadow con nome riservata.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
        },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Impostazione di una policy utente

iam: autorizzazione PassRole

Infine, devi avere l'autorizzazione a passare il ruolo a AWS IoT Core quando chiami l'operazione UpdatePackageConfigurationAPI.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iot:UpdatePackageConfiguration"
            ],
            "Resource": "arn:aws:iam::111122223333:role/<roleName>"
```

```
        }
    ]
}
```

AWS IoT Autorizzazioni per il download di Jobs da Amazon S3

Il documento del processo viene salvato in Amazon S3. Fai riferimento a questo file quando effettui l'invio tramite Jobs. AWS IoT Devi fornire a AWS IoT Jobs i diritti per scaricare il file (`s3:GetObject`). È inoltre necessario impostare una relazione di fiducia tra Amazon S3 e AWS IoT Jobs. Per istruzioni su come creare queste politiche, consulta Presigned URLs in Managing Jobs.

Autorizzazioni per aggiornare la distinta base del software per una versione del pacchetto

Per aggiornare la distinta base del software per una versione del pacchetto negli stati o del `Draft` `Deprecated` ciclo di vita, sono necessari un AWS Identity and Access Management ruolo e delle politiche per individuare la nuova distinta base del software in Amazon S3 e aggiornare la versione del pacchetto in. `Published` AWS IoT Core

Innanzitutto, inserirai la distinta base del software aggiornata nella tua versione del bucket Amazon S3 e richiamerai l'operazione `UpdatePackageVersion` dell'API con `sboms` il parametro incluso. Successivamente, il responsabile autorizzato assumerà il ruolo IAM che hai creato, individuerà la distinta base del software aggiornata in Amazon S3 e aggiornerà la versione del pacchetto in AWS IoT Core Software Package Catalog.

Per eseguire questo aggiornamento sono necessarie le seguenti politiche:

Policy

- Politica di fiducia che stabilisce un rapporto di fiducia con il principale autorizzato che assume il ruolo di IAM in modo che possa individuare la distinta base aggiornata del software dal bucket con versione del bucket in Amazon S3 e aggiornare la versione del pacchetto in. AWS IoT Core
  -

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "s3.amazonaws.com"
            },
                "Action": "sts:AssumeRole"
            }
        ]
}
```

-

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- Politica sulle autorizzazioni: politica per accedere al bucket con versioni di Amazon S3 in cui è archiviata la distinta base del software per una versione del pacchetto e in cui aggiornare la versione del pacchetto. AWS IoT Core

  -

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::awsexamplebucket1"
            ]
        }
    ]
}
```

  -

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdatePackageVersion"
            ],
            "Resource": [
                "arn:aws:iot:*:111122223333:package/<packageName>/
version/<versionName>"
            ]
        }
    ]
}
```

- Passa le autorizzazioni dei ruoli: policy che concede l'autorizzazione a passare il ruolo IAM ad Amazon S3 AWS IoT Core e quando [UpdatePackageVersion] chiami l'operazione API.

  -
  ```
  {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "iam:PassRole",
          "s3:GetObject"
        ],
        "Resource": [
          "arn:aws:s3:::awsexamplebucket1"
        ]
      }
    ]
  }
  ```

  -
  ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
                  "iam:PassRole",
                  "iot:UpdatePackageVersion"
              ],
  ```

```
            "Resource": "arn:aws:iam::111122223333:role/<roleName>"
        }
    ]
}
```

> **Note**
>
> Non è possibile aggiornare la distinta base del software su una versione del pacchetto che è passata allo stato del ciclo di vita. `Deleted`

Per ulteriori informazioni sulla creazione di un ruolo IAM per un AWS servizio, consulta Creazione di un ruolo per delegare l'autorizzazione a un servizio. AWS

Per ulteriori informazioni sulla creazione di un bucket Amazon S3 e sul caricamento di oggetti in esso, consulta Creazione di un bucket e Caricamento di oggetti.

Preparazione dell'indicizzazione del parco istanze

Con l'indicizzazione AWS IoT della flotta, puoi cercare e aggregare i dati utilizzando il nome riservato shadow (). `$package` Puoi anche raggruppare gli AWS IoT oggetti interrogando i gruppi di oggetti Copia shadow con nome riservata dinamici. Ad esempio, è possibile trovare informazioni su quali AWS IoT elementi utilizzano una versione specifica del pacchetto, su quali non è installata una versione specifica del pacchetto o su quali non è installata alcuna versione del pacchetto. Puoi ottenere ulteriori informazioni dettagliate combinando gli attributi. Ad esempio, identificare gli elementi che hanno una versione specifica e sono di un tipo di oggetto specifico (come la versione 1.0.0 e il tipo di oggetto di pump_sensor). Per ulteriori informazioni, consulta Indicizzazione del parco istanze.

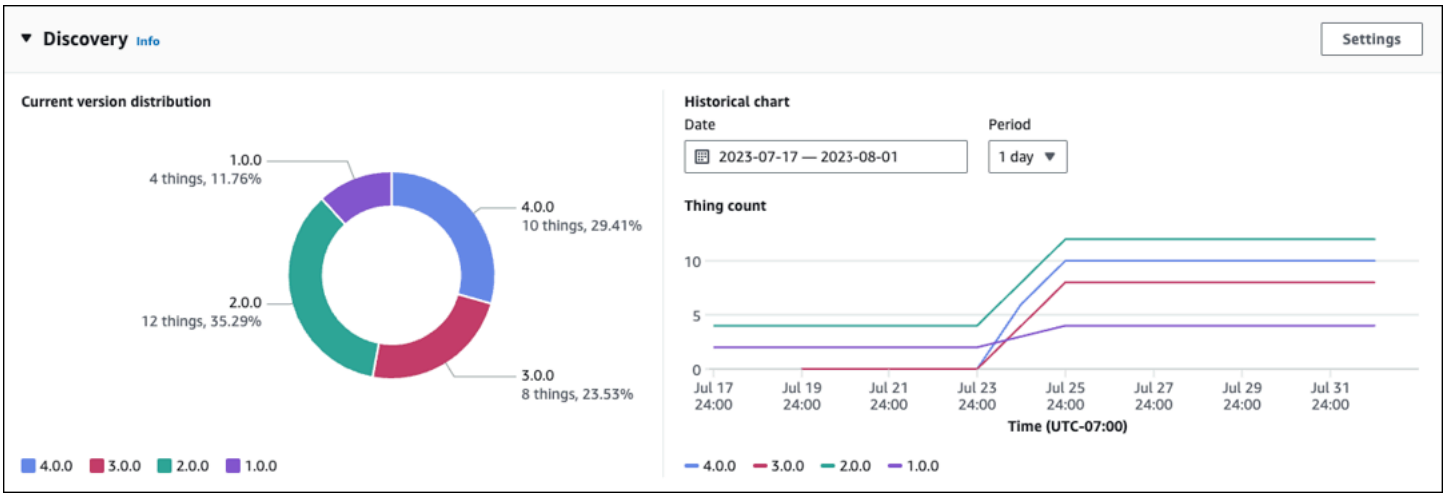
Impostazione della copia shadow **$package** come un'origine dati

Per utilizzare l'indicizzazione del parco istanze con Software Package Catalog, devi abilitare l'indicizzazione del parco istanze, impostare la copia shadow con nome come l'origine dati e definire `$package` come il filtro della copia shadow con nome. Se non hai abilitato l'indicizzazione del parco istanze, puoi abilitarla all'interno di questo processo. Da AWS IoT Core nella console, apri Impostazioni, scegli Gestisci indicizzazione, quindi seleziona Aggiungi copie shadow denominate, Aggiunta di pacchetti software del dispositivo e versioni e Aggiorna. Per ulteriori informazioni, consulta Gestione dell'indicizzazione degli oggetti.

In alternativa, puoi abilitare l'indicizzazione del parco istanze quando crei il primo pacchetto. Quando viene visualizzata la finestra di dialogo Abilita dipendenze per la gestione dei pacchetti, scegli l'opzione per aggiungere pacchetti e versioni software del dispositivo come origini dati all'indicizzazione del parco istanze. Selezionando questa opzione, abiliti anche l'indicizzazione del parco istanze.

> **Note**
>
> L'abilitazione dell'indicizzazione del parco istanze per Software Package Catalog comporta costi di servizio standard. Per ulteriori informazioni, consulta Prezzi di AWS IoT Device Management.

Parametri visualizzati nella console



Nella pagina dei dettagli del pacchetto software della AWS IoT console, il pannello Discovery mostra le metriche standard acquisite dall'ombra. `$package`

- Il grafico di distribuzione della versione corrente mostra il numero di dispositivi e la percentuale per le 10 versioni del pacchetto più recenti associate a un AWS IoT elemento tra tutti i dispositivi associati a questo pacchetto software. Nota: se il pacchetto software dispone di più versioni del pacchetto di quelle etichettate nel grafico, puoi trovarle raggruppate in Altro.
- Nel Grafico storico viene visualizzato il numero di dispositivi associati alle versioni dei pacchetti selezionate in un periodo di tempo specificato. Il grafico è inizialmente vuoto finché non si seleziona un massimo di 5 versioni del pacchetto e si definisce l'intervallo di date e l'intervallo di tempo. Per selezionare i parametri del grafico, scegli Impostazioni. I dati visualizzati nel Grafico storico potrebbero essere diversi da quelli del grafico Distribuzione della versione corrente a causa

della differenza nel numero di versioni del pacchetto visualizzate e anche perché puoi scegliere quali versioni del pacchetto analizzare nel Grafico storico. Nota: quando selezioni una versione del pacchetto da visualizzare, questa viene conteggiata ai fini del numero massimo di limiti dei parametri del parco istanze. Per ulteriori informazioni, consulta Limiti e quote per l'indicizzazione del parco istanze.

Per un altro metodo per ottenere informazioni dettagliate sulla raccolta della distribuzione delle versioni dei pacchetti, consulta Raccolta della distribuzione delle versioni dei pacchetti tramite `getBucketsAggregation`.

Modelli di query

L'indicizzazione del parco veicoli con Software Package Catalog utilizza la maggior parte delle funzionalità standard supportate (come termini, frasi e campi di ricerca). Tuttavia, gli operatori di confronto (ad esempio, minore di < e maggiore di>) e `range` le query non sono disponibili per la chiave riservata denominata shadow (). `$package version` Queste interrogazioni sono disponibili per la `attributes` chiave. Per ulteriori informazioni, consulta Sintassi delle query.

Dati di esempio

Nota: per informazioni sulla copia shadow con nome riservata e la struttura relativa, consulta Copia shadow denominata riservata.

In questo esempio, un primo dispositivo viene denominato `AnyThing` e dispone dei seguenti pacchetti installati:

- Pacchetto software: `SamplePackage`

  Versione del pacchetto: `1.0.0`

  ID pacchetto: `1111`

L'aspetto della copia shadow è il seguente:

```
{
    "state": {
        "reported": {
            "SamplePackage": {
                "version": "1.0.0",
                "attributes": {
```

```
                    "s3UrlForSamplePackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
                    "packageID": "1111"
                    }
            }
        }
    }
}
```

Un secondo dispositivo viene denominato `AnotherThing` e dispone dei seguenti pacchetti installati:

- Pacchetto software: `SamplePackage`

  Versione del pacchetto: `1.0.0`

  ID pacchetto: `1111`
- Pacchetto software: `OtherPackage`

  Versione del pacchetto: `1.2.5`

  ID pacchetto: `2222`

L'aspetto della copia shadow è il seguente:

```
{
    "state": {
        "reported": {
            "SamplePackage": {
                "version": "1.0.0",
                "attributes": {
                    "s3UrlForSamplePackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
                    "packageID": "1111"
                }
            },
            "OtherPackage": {
                "version": "1.2.5",
                "attributes": {
                    "s3UrlForOtherPackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile2",
                    "packageID": "2222"
                    }
```

```
            },
        }
    }
}
```

Query di esempio

Nella tabella seguente vengono elencate query campione basate sulle device shadow di esempio per `AnyThing` e `AnotherThing`. Per ulteriori informazioni, consulta Esempio di query per oggetti.

Versione più recente di AWS IoT Device Tester for FreerTOS

Informazioni richieste	Query	Risultato
Oggetti con installat a una versione del pacchetto specifica	`shadow.name.$package.reported.SamplePackage.version:1.0.0`	`AnyThing`, `OtherThing`
Oggetti che non hanno una versione del pacchetto specifica installata	`NOT shadow.name.$package.reported.OtherPackage.version:1.2.5`	`AnyThing`
Qualsiasi dispositivo che utilizza una versione del pacchetto il cui ID pacchetto è maggiore di 1500	`shadow.name.$package.reported.*.attributes.packageID>1500"`	`OtherThing`
Oggetti con installato un pacchetto specifico e con installati più pacchetti	`shadow.name.$package.reported.SamplePackage.version:1.0.0 AND shadow.name.$package.reported.totalCount:2`	`OtherThing`

Raccolta della distribuzione delle versioni dei pacchetti tramite `getBucketsAggregation`

Oltre al pannello Discovery all'interno della AWS IoT console, puoi anche ottenere informazioni sulla distribuzione della versione del pacchetto utilizzando l'operazione `GetBucketsAggregation`API. Per ottenere informazioni sulla distribuzione delle versioni dei pacchetti, devi eseguire le seguenti operazioni:

- Definisci un campo personalizzato all'interno dell'indicizzazione del parco istanze per ogni pacchetto software. Nota: la creazione di campi personalizzati conta ai fini delle Service Quotas di indicizzazione del parco istanze AWS IoT.
- Formatta il campo personalizzato come segue:

  `shadow.name.$package.reported.<packageName>.version`

Per ulteriori informazioni, consulta la sezione Campi personalizzati nell'indicizzazione del AWS IoT parco veicoli.

Preparazione dei lavori AWS IoT

AWS IoT Device Management Software Package Catalog estende AWS IoT Jobs tramite parametri di sostituzione e integrazione con l'indicizzazione AWS IoT della flotta, i gruppi di oggetti dinamici e il nome shadow riservato dell' AWS IoT oggetto.

> **Note**
>
> Per utilizzare tutte le funzionalità offerte da Software Package Catalog, è necessario creare i seguenti ruoli e politiche AWS Identity and Access Management (IAM): i diritti AWS IoT Jobs per distribuire le versioni dei pacchetti e i diritti AWS IoT Jobs per aggiornare lo shadow denominato riservato. Per ulteriori informazioni, consulta Preparazione della sicurezza.

Parametri di sostituzione per i lavori AWS IoT

È possibile utilizzare i parametri di sostituzione come segnaposto all'interno del documento di lavoro. AWS IoT Quando il servizio del processo rileva un parametro di sostituzione, indirizza il processo a un attributo della versione software con nome per il valore del parametro. Puoi utilizzare questo

processo per creare un singolo documento del processo e passare i metadati al processo tramite attributi di uso generico. Ad esempio, potrebbe essere necessario passare un URL di Amazon Simple Storage Service (Amazon S3), un nome della risorsa Amazon (ARN) del pacchetto software o una firma nel documento del processo tramite gli attributi della versione del pacchetto.

I parametri di sostituzione devono essere formattati nel documento di lavoro come segue:

- Nome e versione del pacchetto software
  - La stringa vuota intermedia `package::version` rappresenta il parametro di sostituzione del nome del pacchetto software. La stringa vuota tra i due `version::attribute` rappresenta il parametro di sostituzione della versione del pacchetto software.
    Fate riferimento al seguente esempio per utilizzare i parametri di sostituzione del nome del pacchetto e della versione del pacchetto in un documento di lavoro:.
    `${aws:iot:package::version::attributes:<attributekey>}`
  - Il documento di lavoro compilerà automaticamente questi parametri di sostituzione utilizzando l'ARN della versione dai dettagli della versione del pacchetto. Se stai creando un processo o un modello di processo per una distribuzione a pacchetto singolo utilizzando un comando API o CLI, l'ARN della versione per una versione del pacchetto è rappresentato dal `destinationPackageVersions` parametro in and. `CreateJob` `DescribeJob`
- Tutti gli attributi di una versione del pacchetto software
  - Fate riferimento al seguente esempio per utilizzare tutti gli attributi di un parametro di sostituzione della versione di un pacchetto software in un documento di lavoro:
    `${aws:iot:package:<packageName>:version:<versionName>:attributes}`

> **Note**
>
> Il nome del pacchetto, la versione del pacchetto e i parametri di sostituzione di tutti gli attributi possono essere utilizzati insieme. Fate riferimento al seguente esempio per utilizzare tutti e tre i parametri di sostituzione in un documento di lavoro:
> `${aws:iot:package::version::attributes}`

Nell'esempio seguente, esiste un pacchetto software denominato `samplePackage` e ha una versione del pacchetto denominata `2.1.5` con i seguenti attributi:

- nome: `s3URL`, valore: `https://EXAMPIEBUCKET.s3.us-west-2.amazonaws.com/exampleCodeFile`

  - Questo attributo identifica la posizione del file di codice archiviato in Amazon S3.
- nome: `signature`, valore: `aaaaabbbbbccccddddeeeeefffffgggghhhhhiiiiijjjj`
  - Questo attributo fornisce un valore di firma del codice richiesto dal dispositivo come misura di sicurezza. Per ulteriori informazioni, consulta Firma del codice per i processi. Nota: questo attributo è un esempio e non è richiesto come parte del Software Package Catalog o dei processi.

Per `s3URL`, il parametro del documento del processo è scritto come segue:

```
{
"samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
}
```

Per `signature`, il parametro del documento del processo è scritto come segue:

```
{
"samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
}
```

Il documento del processo completo è scritto come segue:

```
{
  ...
  "Steps": {
    "uninstall": ["samplePackage"],
    "download": [
      {
        "samplePackage":
 "${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
      },
    ],
    "signature": [
      "samplePackage" :
 "${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
    ]
  }
}
```

Dopo aver effettuato la sostituzione, nei dispositivi viene distribuito il seguente documento del processo:

```
{
  ...
  "Steps": {
    "uninstall": ["samplePackage"],
    "download": [
      {
        "samplePackage": "https://EXAMPIEBUCKET.s3.us-west-2.amazonaws.com/
exampleCodeFile"
      },
    ],
    "signature": [
      "samplePackage" : "aaaaabbbbbcccccdddddeeeeeffffffgggggghhhhhiiiiijjjj"
    ]
  }
}
```

Parametri di sostituzione (visualizzazione prima e dopo)

I parametri di sostituzione semplificano la creazione di un documento di lavoro utilizzando vari flag, ad esempio per la versione predefinita del pacchetto. `$default` Ciò elimina la necessità di inserire manualmente i metadati della versione specifica del pacchetto per ogni distribuzione del processo, poiché tali flag vengono riempiti automaticamente con i metadati di riferimento nella versione specifica del pacchetto. Per ulteriori informazioni sugli attributi della versione del pacchetto, ad esempio `$default` per la versione predefinita del pacchetto, vedere. Preparazione del documento del processo e della versione del pacchetto per la distribuzione

In Console di gestione AWS, attiva il pulsante Anteprima della sostituzione nella finestra dell'editor del file di istruzioni di distribuzione durante la distribuzione di un lavoro per una versione del pacchetto per visualizzare il documento del lavoro con e senza i parametri di sostituzione.

Utilizzando il parametro «before-substitution» in and, è possibile visualizzare la risposta dell'API prima `DescribeJob` e `GetJobDocument` APIs dopo la rimozione dei parametri di sostituzione. Fate riferimento ai seguenti esempi con e: `DescribeJob` `GetJobDocument` APIs

- `DescribeJob`
  - Visualizzazione predefinita

    ```
    {
    ```

```
    "jobId": "<jobId>",
    "description": "<description>",
    "destinationPackageVersions": ["arn:aws:iot:us-west-2:123456789012:package/
TestPackage/version/1.0.2"]
}
```

- Prima della visualizzazione sostitutiva

```
{
    "jobId": "<jobId>",
    "description": "<description>",
    "destinationPackageVersions": ["arn:aws:iot:us-west-2:123456789012:package/
TestPackage/version/$default"]
}
```

- `GetJobDocument`
  - Visualizzazione predefinita

```
{
    "attributes": {
        "location": "prod-artifacts.s3.us-east-1.amazonaws.com/mqtt-core",
        "signature": "IQoJb3JpZ2luX2VjEIrwEaCXVzLWVhc3QtMSJHMEUCIAofPNPpZ9cI",
        "streamName": "mqtt-core",
        "fileId": "0"
    },
}
```

  - Prima della visualizzazione sostitutiva

```
{
    "attributes": "${aws:iot:package:TestPackage:version:$default:attributes}",
}
```

Per ulteriori informazioni sui AWS IoT lavori, sulla creazione di documenti di lavoro e sulla distribuzione dei lavori, consulta Jobs.

Preparazione del documento del processo e della versione del pacchetto per la distribuzione

Quando viene creata una versione del pacchetto, è in uno `draft` stato tale da indicare che è in fase di preparazione per la distribuzione. Per preparare la versione del pacchetto per la distribuzione, è necessario creare un documento di lavoro, salvare il documento in una posizione accessibile al processo (ad esempio Amazon S3) e confermare che la versione del pacchetto abbia i valori degli attributi che si desidera utilizzare nel documento di lavoro. (Nota: puoi aggiornare gli attributi per una versione del pacchetto solo mentre è nello `draft` stato.)

Quando si crea un AWS IoT Job o un modello di Job per una distribuzione a pacchetto singolo, sono disponibili le seguenti opzioni per personalizzare il documento di lavoro:

File di istruzioni di distribuzione () **`recipe`**

- Il file di istruzioni di distribuzione per una versione del pacchetto contiene le istruzioni di distribuzione, incluso un documento di lavoro in linea, per la distribuzione di una versione del pacchetto su più dispositivi. Il file associa istruzioni di distribuzione specifiche a una versione del pacchetto per una distribuzione del lavoro rapida ed efficiente.

  In Console di gestione AWS, è possibile creare il file nella finestra di anteprima del file delle istruzioni di distribuzione nella scheda Configurazioni di distribuzione delle versioni del flusso di lavoro per la creazione di un nuovo pacchetto. Puoi sfruttarlo AWS IoT per generare automaticamente un file di istruzioni dagli attributi della versione del pacchetto utilizzando Start from AWS IoT recommended file o utilizzare il documento di lavoro esistente archiviato in un bucket Amazon S3 utilizzando Usa il tuo file di istruzioni di distribuzione.

  > **Note**
  >
  > Se utilizzi il tuo documento di lavoro, puoi aggiornarlo direttamente nella finestra di anteprima del file delle istruzioni di distribuzione, ma non aggiornerà automaticamente il documento di lavoro originale archiviato nel tuo bucket Amazon S3.

  Quando si utilizza AWS CLI o un comando API come`CreatePackageVersion`, or `GetPackageVersionUpdatePackageVersion`, `recipe` rappresenta il file di istruzioni di distribuzione, che include un documento di lavoro in linea.

Per ulteriori informazioni su cos'è un documento di lavoro, vedereConcetti di base.

Fate riferimento al seguente esempio per il file di istruzioni di distribuzione rappresentato da`recipe`:

```
{
    "packageName": "sample-package-name",
    "versionName": "sample-package-version",
    ...
    "recipe": "{...}"
}
```

> **Note**
>
> Il file di istruzioni di distribuzione rappresentato da `recipe` può essere aggiornato quando la versione di un pacchetto si trova nello `published` stato in cui è separato dai metadati della versione del pacchetto. Diventa immutabile durante l'implementazione del lavoro.

`Artifact`attributo di versione

- Utilizzando l'attributo version `artifact` nella versione del pacchetto software, puoi aggiungere la posizione Amazon S3 per gli artefatti della versione del pacchetto. Quando viene attivata una distribuzione di lavoro per la versione del pacchetto utilizzando AWS IoT Jobs, il segnaposto URL predefinito `${aws:iot:package:<packageName>:version:<versionName>:artifact-location:s3-presigned-url}` nel documento del processo verrà aggiornato utilizzando il bucket Amazon S3, la chiave del bucket e la versione del file archiviato nel bucket Amazon S3. Il bucket Amazon S3 che archivia gli artefatti della versione del pacchetto deve trovarsi nella stessa regione in cui è stata creata la versione del pacchetto.

  > **Note**
  >
  > Per archiviare più versioni di oggetti dello stesso file nel tuo bucket Amazon S3, devi abilitare il controllo delle versioni sul bucket. Per ulteriori informazioni, consulta Abilitare il controllo delle versioni sui bucket.

Per accedere agli elementi della versione del pacchetto nel bucket Amazon S3 quando si utilizza `CreatePackageVersion` l'operazione `UpdatePackageVersion` o l'API, è necessario disporre delle seguenti autorizzazioni:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::bucket-name/key-name"
        }
    ]
}
```

Per ulteriori informazioni sull'attributo version `artifact` nelle operazioni `CreatePackageVersion` e `UpdatePackageVersion` API, consulta e. CreatePackageVersionUpdatePackageVersion

Fai riferimento al seguente esempio che mostra l'attributo version che `artifact` supporta la posizione dell'artefatto in Amazon S3 durante la creazione di una nuova versione del pacchetto:

```
{
    "packageName": "sample package name",
    "versionName": "1.0",
    "artifact": {
        "s3Location": {
            "bucket": "firmware",
            "key": "image.bin",
            "version": "12345"
        }
    }
}
```

> **Note**
>
> Quando una versione del pacchetto viene aggiornata da uno stato di `draft` status a uno stato di `published` status, gli attributi della versione del pacchetto e la posizione

dell'artificat diventano immutabili. Per aggiornare queste informazioni, è necessario creare una nuova versione del pacchetto ed eseguire tali aggiornamenti mentre si trova nello stato. `draft`

Versione del pacchetto

- Una versione del pacchetto software predefinita può essere indicata nelle versioni disponibili del pacchetto software che forniscono una versione del pacchetto sicura e stabile. Questa serve come versione di base del pacchetto software quando si distribuisce la versione del pacchetto predefinita nel parco dispositivi utilizzando Jobs. AWS IoT Quando si crea un lavoro per distribuire la versione del `$default` pacchetto per un pacchetto software, la versione del pacchetto nel documento del lavoro e nella nuova distribuzione del lavoro deve corrispondere a. `$default` La versione del pacchetto nella distribuzione del processo è rappresentata dai comandi `destinationPackageVersions` for API e CLI e `VersionARN` in. Console di gestione AWS La versione del pacchetto nel documento di lavoro è rappresentata dal seguente segnaposto del documento di lavoro mostrato di seguito:

  ```
  arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$default
  ```

  Per creare un lavoro o un modello di lavoro utilizzando la versione del pacchetto predefinita, utilizzate il `$default` flag nel comando `CreateJob` o `CreateJobTemplate` API come illustrato di seguito:

  ```
  "$ aws iot create-job \
      --destination-package-versions "arn:aws:iot:us-west-2:123456789012:package/
  TestPackage/version/$default"
      --document file://jobdoc.json
  ```

  > **Note**
  >
  > L'attributo della versione del `$default` pacchetto che fa riferimento alla versione predefinita è un attributo facoltativo richiesto solo quando si fa riferimento alla versione predefinita del pacchetto per una distribuzione di lavoro tramite AWS IoT Jobs.

Quando sei soddisfatto della versione del pacchetto, pubblicala tramite la pagina dei dettagli del pacchetto software nella AWS IoT console o eseguendo l'UpdatePackageVersionoperazione API. È quindi possibile fare riferimento alla versione del pacchetto quando si crea il processo tramite la AWS IoT console o emettendo l'operazione CreateJobAPI.

Assegnazione di un nome ai pacchetti e alle versioni durante la distribuzione

Per distribuire la versione di un pacchetto software su un dispositivo, verificate che il pacchetto software e la versione del pacchetto a cui si fa riferimento nel documento di lavoro corrispondano al pacchetto software e alla versione del pacchetto indicati nel `destinationPackageVersions` parametro dell'operazione API. `CreateJob` Se non corrispondono, riceverai un messaggio di errore che ti chiederà di far corrispondere entrambi i riferimenti. Per ulteriori informazioni sui messaggi di errore del Software Package Catalog, vedereMessaggi di errore generali sulla risoluzione dei problemi.

Oltre ai pacchetti software e alle versioni dei pacchetti a cui si fa riferimento nel documento di lavoro, è possibile includere pacchetti software e versioni di pacchetto aggiuntivi nel `destinationPackageVersions` parametro dell'operazione `CreateJob` API non referenziata nel documento del lavoro. Assicuratevi che nel documento di lavoro siano incluse le informazioni di installazione necessarie affinché i dispositivi installino correttamente le versioni del pacchetto software aggiuntivo. Per ulteriori informazioni sul funzionamento dell'`CreateJob` API, consulta CreateJob.

Indirizzazione dei lavori tramite gruppi di AWS IoT oggetti dinamici

Software Package Catalog funziona con l'indicizzazione del parco istanze, i processi di AWS IoT e i gruppi di oggetti dinamici AWS IoT per filtrare e definire i dispositivi all'interno del parco istanze per selezionare la versione del pacchetto da distribuire nei dispositivi. Puoi eseguire una query di indicizzazione del parco veicoli in base alle informazioni sulla confezione corrente del tuo dispositivo e indirizzare tali elementi per un determinato lavoro. AWS IoT Puoi anche rilasciare aggiornamenti software, ma solo per i dispositivi di destinazione idonei. Ad esempio, puoi specificare che desideri distribuire una configurazione solo sui dispositivi che attualmente eseguono `iot-device-client 1.5.09`. Per ulteriori informazioni, consulta Creazione di un gruppo di oggetti dinamico.

Copia shadow con nome riservata e versioni dei pacchetti

Se configurato, AWS IoT Jobs può aggiornare il nome riservato di un oggetto denominato shadow (`$package`) quando il processo viene completato correttamente. In tal caso, non occorre associare manualmente una versione del pacchetto alla copia shadow con nome riservata di un oggetto.

Puoi scegliere di associare o aggiornare manualmente una versione del pacchetto alla copia shadow con nome riservata dell'oggetto nelle seguenti situazioni:

- Si registra un oggetto AWS IoT Core senza associare la versione del pacchetto installato.
- AWS IoT Jobs non è configurato per aggiornare il nome shadow riservato dell'oggetto.
- Utilizzi un processo interno per inviare le versioni dei pacchetti alla tua flotta e tale processo non AWS IoT Core si aggiorna una volta completato.

> **Note**
>
> Ti consigliamo di utilizzare AWS IoT Jobs per aggiornare la versione del pacchetto nella cartella riservata denominata shadow (). `$package` L'aggiornamento del parametro version nello `$package` shadow tramite altri processi (ad esempio chiamate API manuali o programmatiche) quando anche AWS IoT Jobs è configurato per aggiornare lo shadow, può causare incongruenze tra la versione effettiva sul dispositivo e la versione riportata allo shadow denominato riservato.

Puoi aggiungere o aggiornare una versione del pacchetto a una copia con nome riservata dell'oggetto (`$package`) tramite la console o l'operazione API `UpdateThingShadow`. Per ulteriori informazioni, vedete Associare una versione del pacchetto a un oggetto. AWS IoT

> **Note**
>
> L'associazione di una versione del pacchetto a un AWS IoT elemento non aggiorna direttamente il software del dispositivo. Devi distribuire la versione del pacchetto al dispositivo per aggiornare il software del dispositivo.

Disinstallazione di un pacchetto software e della relativa versione del pacchetto

`$null`è un segnaposto riservato che richiede al servizio AWS IoT Jobs di rimuovere il pacchetto software e la versione del pacchetto esistenti dall'ombra denominata riservata del dispositivo. `$package` Per ulteriori informazioni, consulta Copia shadow con nome riservata.

Per utilizzare questa funzionalità, sostituisci il nome della versione alla fine di destinationPackageVersionAmazon Resource Name (ARN) con. `$null` Successivamente, devi indicare al servizio di rimuovere il software dal dispositivo.

L'ARN autorizzato utilizza il seguente formato:

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

Ad esempio,

```
$ aws iot create-job \
    ... \
    --destinationPackageVersions ["arn:aws:iot:us-east-1:111122223333:package/
samplePackage/version/$null"]
```

Nozioni di base su Software Package Catalog

È possibile creare e gestire il AWS IoT Device Management Software Package Catalog tramite le Console di gestione AWS operazioni AWS IoT Core API e AWS Command Line Interface (AWS CLI).

> **Note**
>
> L'abilitazione dell'indicizzazione AWS IoT della flotta è un requisito per utilizzare Software Package Catalog. Le operazioni di base, come la creazione di una versione del pacchetto software nel comando API Console di gestione AWS e l'utilizzo del comando `CreatePackage` API, avranno esito negativo senza l'indicizzazione del AWS IoT parco veicoli abilitata.
> Per ulteriori informazioni sull'utilizzo dell'indicizzazione del AWS IoT parco veicoli con Software Package Catalog, vedere. Preparazione dell'indicizzazione del parco istanze

Utilizzo della console

Per utilizzare il Console di gestione AWS, accedi al tuo AWS account e vai a. AWS IoT Core Nel riquadro di navigazione, scegli Pacchetti software. Quindi, puoi creare e gestire i pacchetti e le relative versioni da questa sezione.

Utilizzo delle operazioni API o CLI

È possibile utilizzare le operazioni AWS IoT Core API per creare e gestire le funzionalità del Software Package Catalog. Per ulteriori informazioni, consulta AWS IoT API Reference AWS SDKs and Toolkits. I AWS CLI comandi gestiscono anche il tuo catalogo. Per ulteriori informazioni, consulta la pagina di Riferimento ai comandi CLI AWS IoT.

Questo capitolo contiene le sezioni seguenti:

- Creazione di un pacchetto software e di una versione del pacchetto
- Distribuzione di una versione del pacchetto tramite job AWS IoT
- Associare una versione del pacchetto a un oggetto AWS IoT

Creazione di un pacchetto software e di una versione del pacchetto

Puoi utilizzare i seguenti passaggi per creare un pacchetto e un oggetto della versione iniziale tramite la Console di gestione AWS.

Per creare un pacchetto software

1. Accedi al tuo AWS account e accedi alla AWS IoT console.
2. Nel riquadro di navigazione, scegli Pacchetti software.
3. Nella pagina del pacchetto software AWS IoT , scegli Crea pacchetto. Viene visualizzata la finestra di dialogo Abilita dipendenze per la gestione dei pacchetti.
4. In Indicizzazione del parco istanze, seleziona Aggiungi pacchetti software del dispositivo e versione. Ciò è richiesto per Software Package Catalog e fornisce l'indicizzazione del parco istanze e i parametri relativi al parco istanze.
5. [Facoltativo] Se desideri che i AWS IoT job aggiornino lo shadow denominato riservato quando i lavori vengono completati correttamente, seleziona Aggiorna automaticamente le ombre dai lavori. Se non desideri che i AWS IoT lavori effettuino questo aggiornamento, lascia questa casella di controllo deselezionata.

6. [Facoltativo] Per concedere ai AWS IoT lavori il diritto di aggiornare l'ombra denominata riservata, in Seleziona ruolo, scegli Crea ruolo. Se non desideri che i AWS IoT lavori effettuino questo aggiornamento, questo ruolo non è obbligatorio.
7. Crea o seleziona un ruolo.
   a. Se non disponi di un ruolo per questo scopo: quando viene visualizzata la finestra di dialogo Crea ruolo, inserisci un Nome di ruolo, quindi scegli Crea.
   b. Se disponi di un ruolo per questo scopo: in Seleziona ruolo, scegli il ruolo, quindi assicurati che la casella di controllo Collega policy al ruolo IAM sia selezionata.
8. Scegli Conferma. Viene visualizzata la pagina Crea nuovo pacchetto.
9. In Dettagli del pacchetto, inserisci un Nome del pacchetto.
10. In Descrizione pacchetto, inserisci le informazioni per facilitare l'identificazione e la gestione di questo pacchetto.
11. [Facoltativo] Puoi usare i tag per facilitare la classificazione e la gestione di questo pacchetto. Per aggiungere tag, espandi Tag, scegli Aggiungi tag e inserisci una coppia chiave-valore. Puoi inserire fino a 50 tag. Per ulteriori informazioni, consulta Taggare le AWS IoT risorse.

Per aggiungere una versione del pacchetto durante la creazione di un nuovo pacchetto

1. In Versione iniziale, inserisci un nome per la versione.

   Ti consigliamo di utilizzare il SemVer formato (ad esempio,`1.0.0.0`) per identificare in modo univoco la versione del pacchetto. Puoi anche utilizzare una strategia di formattazione diversa più appropriata per il tuo caso d'uso. Per ulteriori informazioni, consulta Ciclo di vita della versione del pacchetto.
2. In Descrizione versione, inserisci le informazioni per facilitare l'identificazione e la gestione di questa versione del pacchetto.

> **Note**
>
> La casella di controllo Versione predefinita è disattivata perché le versioni dei pacchetti vengono create in uno stato `draft`. È possibile assegnare un nome alla versione predefinita dopo aver creato la versione del pacchetto e quando si modifica lo stato in`published`. Per ulteriori informazioni, consulta Ciclo di vita della versione del pacchetto.

3. [Facoltativo] Per semplificare la gestione di questa versione o per comunicare informazioni ai dispositivi, inserisci una o più coppie nome-valore per Attributi di versione. Scegli Aggiungi attributo per ciascuna coppia nome-valore inserita. Per ulteriori informazioni, consulta Attributi della versione.
4. [Facoltativo] Puoi usare i tag per facilitare la classificazione e la gestione di questo pacchetto. Per aggiungere tag, espandi Tag, scegli Aggiungi tag e inserisci una coppia chiave-valore. Puoi inserire fino a 50 tag. Per ulteriori informazioni, consulta Taggare le AWS IoT risorse.
5. Scegli Next (Successivo).

Associare la distinta base del software a una versione del pacchetto (opzionale)

1. Al Passaggio 3: Versione SBOMs (opzionale) nella finestra delle configurazioni SBOM, scegliete il formato di file SBOM predefinito e la modalità di convalida utilizzati per convalidare la distinta base del software prima che venga associata alla versione del pacchetto.
2. Nella finestra Aggiungi file SBOM, inserisci l'Amazon Resource Name (ARN) che rappresenta la tua versione del bucket Amazon S3 e il formato di file SBOM preferito se il tipo predefinito non funziona.

> **Note**
>
> Puoi aggiungere un singolo file SBOM o un singolo file zip contenente più SBOMs di una distinta software per la versione del pacchetto.

3. Nella finestra Aggiunto file SBOM, è possibile visualizzare il file SBOM aggiunto per la versione del pacchetto.
4. Scegli Crea pacchetto e versione. Viene visualizzata la pagina della versione del pacchetto e puoi vedere lo stato di convalida del tuo file SBOM nella finestra Aggiungi file SBOM. Lo stato iniziale sarà quello in cui il file SBOM viene `In progress` sottoposto a convalida.

> **Note**
>
> Gli stati di convalida del file SBOM sono`Invalid file`,,,`Not started`, `In progress` e i motivi dell'errore di convalida. `Validated (SPDX)` `Validated (CycloneDX)`

Distribuzione di una versione del pacchetto tramite job AWS IoT

Puoi utilizzare la procedura seguente per distribuire una versione del pacchetto tramite la Console di gestione AWS.

Prerequisiti:

Prima di iniziare, esegui queste attività:

- Registra AWS IoT le cose con AWS IoT Core. Per indicazioni su come aggiungere i dispositivi AWS IoT Core, consulta Creare un oggetto.
- [Facoltativo] Crea un AWS IoT gruppo di oggetti o un gruppo di oggetti dinamico per indirizzare i dispositivi su cui distribuirai la versione del pacchetto. Per istruzioni su come creare un gruppo di oggetti, consulta Creazione di un gruppo di oggetti statico. Per istruzioni su come creare un gruppo di oggetti dinamico, consulta Creazione di un gruppo di oggetti dinamico.
- Crea un pacchetto software e una versione del pacchetto. Per ulteriori informazioni, consulta Creazione di un pacchetto software e di una versione del pacchetto.
- Crea un documento del processo. Per ulteriori informazioni, consulta Preparazione del documento del processo e della versione del pacchetto per l'implementazione.

Per distribuire un lavoro AWS IoT

1. Sulla console AWS IoT, scegli Pacchetti software.
2. Scegli il pacchetto software che desideri distribuire. Viene visualizzata la pagina dei dettagli del pacchetto software.
3. Scegli la versione del pacchetto che desideri distribuire, in Versioni, e scegli Distribuisci versione del processo.
4. Se è la prima volta che distribuisci un processo tramite questo portale, viene visualizzata una finestra di dialogo che descrive i requisiti. Rivedi le informazioni e scegli Accetta.
5. Inserisci un nome per l'implementazione o lascia il nome generato automaticamente nel campo Nome.
6. [Facoltativo] Nel campo Descrizione, inserisci una descrizione che identifica lo scopo o i contenuti dell'implementazione o lascia le informazioni generate automaticamente.

   Nota: si consiglia di non utilizzare informazioni personali di identificazione nel nome del processo e nei campi di descrizione.

7. [Facoltativo] Aggiungi eventuali tag da associare a questo processo.
8. Scegli Next (Successivo).
9. In Destinazioni del processo, scegli gli oggetti o i gruppi di oggetti che devono ricevere il processo.
10. Nel campo File del processo, specifica il file JSON del documento del processo.
11. Apri Integrazione dei processi con il servizio Package Catalog.
12. Seleziona i pacchetti e le versioni specificati all'interno del documento del processo.

> **Note**
>
> Ti viene richiesto di scegliere gli stessi pacchetti e le stesse versioni dei pacchetti specificati nel documento del processo. Puoi includerne altri, ma il processo fornirà istruzioni solo per i pacchetti e le versioni inclusi nel documento del processo.
> Per ulteriori informazioni, consulta Denominazione di pacchetti e versioni durante l'implementazione.

13. Scegli Next (Successivo).
14. Nella pagina Configurazione del processo, seleziona uno dei seguenti tipi di processo nella finestra di dialogo Configurazione del processo:

    - Snapshot job (Processo snapshot): un processo snapshot viene completato quando termina la sua esecuzione su dispositivi e gruppi di destinazione.
    - Continuous job (Processo continuo): un processo continuo si applica ai gruppi di oggetti e viene eseguito su qualsiasi dispositivo aggiunto successivamente a un gruppo di destinazione specificato.

15. Nella finestra di dialogo Configurazioni aggiuntive - opzionale, esamina le seguenti configurazioni del processo opzionali ed esegui le selezioni di conseguenza. Per ulteriori informazioni, consulta Configurazioni di rollout, pianificazione e interruzione dei processi e Configurazioni di timeout e di nuovo tentativo di esecuzione del processo.

    - Rollout configuration (Configurazione rollout)
    - Scheduling configuration (Configurazione della pianificazione)
    - Job executions timeout configuration (Configurazione del timeout di esecuzioni dei processi)
    - Configurazione dei tentativi delle esecuzioni dei processi
    - Configurazione dell’interruzione

16. Esamina le selezioni del processo e scegli Invia.

Dopo aver creato il processo, la console genera una firma JSON e la inserisce nel documento di processo. È possibile utilizzare la AWS IoT console per visualizzare lo stato di un lavoro oppure annullarlo o eliminarlo. Per gestire i processi, vai all'Hub di processo della console.

Associare una versione del pacchetto a un oggetto AWS IoT

Dopo aver installato il software sul dispositivo, è possibile associare una versione del pacchetto a un AWS IoT oggetto denominato shadow riservato. Se i AWS IoT job sono stati configurati per aggiornare il named shadow riservato dell'oggetto dopo che il job è stato distribuito e completato con successo, non è necessario completare questa procedura. Per ulteriori informazioni, consulta Copia shadow con nome riservata.

Prerequisiti:

Prima di iniziare, esegui queste attività:

- Crea una o AWS IoT più cose e stabilisci la telemetria tramite. AWS IoT Core Per ulteriori informazioni, consulta Guida introduttiva. AWS IoT Core
- Crea un pacchetto software e una versione del pacchetto. Per ulteriori informazioni, consulta Creazione di un pacchetto software e di una versione del pacchetto.
- Installa il software della versione del pacchetto sul dispositivo.

> **Note**
>
> L'associazione di una versione del pacchetto a un AWS IoT elemento non comporta l'aggiornamento o l'installazione del software sul dispositivo fisico. La versione del pacchetto deve essere distribuita sul dispositivo.

Associare una versione del pacchetto a un oggetto AWS IoT

1. Nel riquadro di navigazione della console AWS IoT, espandi il menu Tutti i dispositivi e scegli Oggetti.
2. Identifica l' AWS IoT elemento che desideri aggiornare dall'elenco e scegli il nome dell'oggetto per visualizzarne la pagina dei dettagli.

3. Nella sezione Dettagli, scegli Pacchetti e versioni.
4. Scegli Aggiungi al pacchetto e alla versione.
5. In Scegli un pacchetto di dispositivi, seleziona il pacchetto software desiderato.
6. In Scegli una versione, seleziona la versione del software desiderata.
7. Scegli Aggiungi pacchetto di dispositivi.

   Il pacchetto e la versione vengono visualizzati nell'elenco Pacchetti e versioni selezionate.
8. Ripeti questi passaggi per ogni pacchetto e versione che desideri associare a questo oggetto.
9. Al termine, scegli Aggiungi pacchetto e dettagli della versione. Viene visualizzata la pagina Dettagli dell'oggetto in cui è possibile visualizzare il nuovo pacchetto e la nuova versione nell'elenco.

AWS IoT Lavori

Usa AWS IoT Jobs per definire una serie di operazioni remote che possono essere inviate ed eseguite su uno o più dispositivi a cui sono collegati AWS IoT. Ad esempio, è possibile definire un processo che indichi a un insieme di dispositivi di scaricare e installare applicazioni, eseguire aggiornamenti, eseguire il riavvio, ruotare i certificati o eseguire operazioni di risoluzione dei problemi in remoto.

Accedere ai AWS IoT lavori

Puoi iniziare a usare AWS IoT Jobs utilizzando la console o l' AWS IoT Core API.

Utilizzo della console

Accedi a Console di gestione AWS e vai alla AWS IoT console. Nel riquadro di navigazione, scegliere Manage (Gestisci), quindi Jobs (Processi). Puoi creare e gestire processi da questa sezione. Se vuoi creare e gestire modelli di processo, nel pannello di navigazione scegli Job Templates (Modelli di processo). Per ulteriori informazioni, consulta Crea e gestisci lavori utilizzando Console di gestione AWS.

Utilizzo dell'API o dell'interfaccia a riga di comando

Puoi iniziare utilizzando le operazioni AWS IoT Core API. Per ulteriori informazioni, consulta la Documentazione di riferimento delle API AWS IoT. L' AWS IoT Core API su cui si basa AWS IoT Jobs è supportata dall' AWS SDK. Per ulteriori informazioni, consulta AWS SDKs and Toolkits.

È possibile utilizzare i comandi AWS CLI per eseguire comandi per creare e gestire lavori e modelli di lavoro. Per ulteriori informazioni, consulta la documentazione di riferimento dell'interfaccia della riga di comando AWS IoT.

AWS IoT Offerte di lavoro, regioni ed endpoint.

AWS IoT Jobs supporta gli endpoint API del piano di controllo e del piano dati specifici del tuo. Regione AWS Gli endpoint dell'API del piano dati sono specifici del tuo Account AWS and. Regione AWSPer ulteriori informazioni sugli endpoint AWS IoT Jobs, consulta AWS IoT Device Management - jobs data endpoints nel General AWS Reference.

Cos'è un'operazione remota?

> **Note**
>
> Over-the-air Gli aggiornamenti (OTA) non sono disponibili nelle seguenti regioni:
>
> - Europa (Spagna)
> - Asia Pacifico (Malesia)

Un'operazione remota è qualsiasi aggiornamento o azione che è possibile eseguire su un dispositivo fisico, un dispositivo virtuale o un endpoint e che può essere eseguita in remoto senza la presenza fisica di un operatore o di un tecnico. L'operazione remota viene eseguita utilizzando un aggiornamento over-the-air (OTA) in modo che i dispositivi non debbano essere fisicamente presenti. La gestione del parco dispositivi in the Cloud AWS consente di eseguire operazioni remote sui dispositivi quando sono registrati AWS IoT Core.

AWS IoT Device Management Jobs offre un approccio scalabile per eseguire azioni remote sui dispositivi registrati con AWS IoT Core. Un lavoro viene creato in Cloud AWS e inviato a tutti i dispositivi di destinazione utilizzando un aggiornamento OTA tramite il protocollo MQTT o HTTP.

AWS IoT Device Management Jobs offre la possibilità di eseguire operazioni remote come il ripristino delle impostazioni di fabbrica, il riavvio dei dispositivi e gli aggiornamenti OTA del software in modo sicuro, scalabile e più conveniente.

Per ulteriori informazioni su, vedere. AWS IoT CoreChe cos'è AWS IoT?

Per ulteriori informazioni sui AWS IoT Device Management lavori, vedereChe cos'è Jobs AWS IoT ?.

Vantaggi dell'utilizzo di AWS IoT Device Management Jobs per operazioni remote

L'utilizzo AWS IoT Device Management di Jobs per eseguire operazioni remote semplifica la gestione del parco dispositivi. L'elenco seguente evidenzia alcuni dei principali vantaggi dell'utilizzo di AWS IoT Device Management Jobs per eseguire operazioni remote:

- Perfetta integrazione con altri Servizi AWS
  - AWS IoT Device Management Jobs si integra strettamente con il valore aggiunto e le funzionalità Servizi AWS seguenti:

    - Amazon S3: archivia le istruzioni operative remote in un bucket Amazon S3 sicuro dove puoi controllare le autorizzazioni di accesso per quel contenuto. L'uso di un bucket Amazon S3 offre una soluzione di storage scalabile e durevole che si integra nativamente con AWS IoT Device Management Software Package Catalog, AWS IoT Device Management permettendo a Jobs di fare riferimento e sostituire le istruzioni di aggiornamento. Per ulteriori informazioni, consulta Cos'è Amazon S3? .
    - Amazon CloudWatch: monitora e registra lo stato di implementazione dell'operazione remota dell'esecuzione del lavoro per ogni dispositivo, oltre all'attività degli altri dispositivi, per tracciare e analizzare le prestazioni lavorative complessive in AWS IoT Device Management Jobs. Per ulteriori informazioni, consulta What is Amazon CloudWatch? Monitoraggio dei registri dei lavori e acquisizione di dati storici per la risoluzione dei problemi. Come funziona con Jobs.
    - Servizio AWS IoT Device Shadow: Mantieni una rappresentazione digitale dei tuoi AWS IoT contenuti tramite un'ombra del dispositivo utilizzando AWS IoT Device Management Jobs in modo che lo stato del dispositivo sia disponibile per applicazioni e altri servizi indipendentemente dalla connettività del dispositivo. Per ulteriori informazioni, consulta Servizio AWS IoT Device Shadow.
- Procedure ottimali di sicurezza
    - Controllo delle autorizzazioni: controlla le autorizzazioni di accesso alle istruzioni operative remote utilizzando Amazon S3 e determina quali utenti IAM possono distribuire le istruzioni operative remote alla tua flotta di dispositivi AWS IoT utilizzando policy e ruoli utente IAM.
        - Per ulteriori informazioni sulle AWS IoT politiche, consulta. Crea una AWS IoT politica
        - Per ulteriori informazioni sui ruoli utente IAM, consultaGestione delle identità e degli accessi per AWS IoT.
- Scalabilità
    - Implementazione mirata del lavoro: controlla quali dispositivi ricevono il documento di lavoro da un lavoro con una distribuzione mirata del lavoro utilizzando criteri di raggruppamento dei dispositivi specifici inseriti nel documento di lavoro al momento della creazione del lavoro. La creazione di un AWS IoT elemento per ogni dispositivo e l'archiviazione di tali informazioni nel AWS IoT registro consentono di eseguire ricerche mirate utilizzando l'indicizzazione del parco veicoli. È possibile creare gruppi personalizzati in base ai risultati della ricerca di indicizzazione del parco veicoli per supportare l'implementazione del lavoro desiderata. Per ulteriori informazioni, consulta Gestione dei dispositivi con AWS IoT. Usa i lavori per eseguire istantanee anziché lavori continui.

  - Stato del lavoro: monitora lo stato dell'implementazione del documento di lavoro nel tuo parco dispositivi e lo stato generale del lavoro a livello di flotta di dispositivi, oltre allo stato di implementazione individuale del documento di lavoro su ciascun dispositivo. Per ulteriori informazioni, consulta Processi e stati di esecuzione dei processi.
  - Nuova scalabilità dei dispositivi: distribuisci facilmente il tuo documento di lavoro su un nuovo dispositivo aggiungendolo a un gruppo personalizzato esistente creato utilizzando l'indicizzazione del parco veicoli tramite un processo continuo. In questo modo risparmierai tempo e non dovrai più distribuire il documento di lavoro su ogni nuovo dispositivo separatamente. In alternativa, è possibile utilizzare un approccio più mirato con un'istantanea distribuendo un documento di lavoro su un gruppo predeterminato di dispositivi una volta e poi il lavoro è completato.
- Flessibilità
  - Configurazioni del lavoro: personalizza il lavoro e il documento di lavoro con le configurazioni di lavoro opzionali rollout, scheduling, abortisci, timeout e riprova per soddisfare le tue esigenze specifiche. Per ulteriori informazioni, consulta Configurazioni diprocesso.
- Conveniente
  - Introduci una struttura dei costi più efficiente per la manutenzione del parco dispositivi sfruttando AWS IoT Device Management Jobs per distribuire aggiornamenti critici ed eseguire attività di manutenzione ordinaria. Una soluzione do-it-yourself (fai-da-te) per la manutenzione del parco dispositivi include costi ricorrenti e variabili, come l'infrastruttura necessaria per ospitare e gestire la soluzione fai-da-te, i costi di manodopera per lo sviluppo, la manutenzione e la scalabilità della soluzione fai-da-te e i costi di trasmissione dei dati. Sfruttando la struttura trasparente e fissa dei costi di AWS IoT Device Management Jobs, sai esattamente quanto costerà l'esecuzione di ogni processo per un dispositivo oltre ai costi di trasmissione dei dati necessari per facilitare l'implementazione dei documenti di lavoro nel tuo parco dispositivi e il monitoraggio dello stato di esecuzione del lavoro per ciascun dispositivo. Per ulteriori informazioni, consultare Prezzi di AWS IoT Core.

Che cos'è Jobs AWS IoT ?

Utilizza AWS IoT Jobs per definire una serie di operazioni remote che possono essere inviate ed eseguite su uno o più dispositivi a cui è possibile connettersi AWS IoT.

Per creare i processi, definisci innanzitutto un documento di processo che contiene un elenco di istruzioni che descrivono le operazioni che il dispositivo deve eseguire in remoto. Per eseguire

queste operazioni, specificare un elenco di obiettivi, che sono oggetti individuali, gruppi di oggetti, o entrambi. Il documento di processo e gli obiettivi costituiscono un'implementazione.

Ciascuna implementazione può avere configurazioni aggiuntive:

- Rollout: Questa configurazione definisce il numero di dispositivi che ricevono il documento di processo ogni minuto.
- Abort (interruzione): se un certo numero di dispositivi non riceve la notifica di processo, utilizza questa configurazione per annullare il processo. In questo modo, eviti l'invio di un aggiornamento errato a un intero parco istanze.
- Timeout: se entro una certa durata gli obiettivi di processo non ricevono una risposta, il processo potrebbe non riuscire. Puoi tenere traccia del processo in esecuzione in questi dispositivi.
- Riprova: se un dispositivo segnala un guasto o un processo scade, puoi utilizzare AWS IoT Jobs per inviare nuovamente il documento di lavoro al dispositivo automaticamente.
- Scheduling (Pianificazione): questa configurazione consente di pianificare un processo per una data e un'ora future. Consente inoltre di creare finestre di manutenzione ricorrenti che aggiornano i dispositivi durante periodi predefiniti con traffico ridotto.

AWS IoT Jobs invia un messaggio per informare gli obiettivi che un lavoro è disponibile. L'obiettivo avvia l'esecuzione del lavoro scaricando il documento del lavoro, eseguendo le operazioni specificate e segnalandone lo stato di AWS IoT avanzamento. È possibile tenere traccia dell'avanzamento di un lavoro per un obiettivo specifico o per tutti gli obiettivi eseguendo i comandi forniti da AWS IoT Jobs. Quando viene avviato, un processo ha lo stato In progress (In corso). I dispositivi segnalano quindi gli aggiornamenti incrementali durante la visualizzazione di questo stato fino alla corretta esecuzione, a un errore o al timeout del processo.

Gli argomenti seguenti descrivono alcuni concetti chiave relativi ai processi, al ciclo di vita dei processi e alle esecuzioni dei processi.

Argomenti

- Concetti chiave dei processi
- Processi e stati di esecuzione dei processi

Concetti chiave dei processi

I concetti seguenti forniscono dettagli sui AWS IoT lavori e su come creare e distribuire lavori per eseguire operazioni remote sui dispositivi.

Concetti di base

Di seguito sono riportati i concetti di base che è necessario conoscere quando si utilizza AWS IoT Jobs.

Processo

Un processo è un'operazione remota che viene inviata a uno o più dispositivi connessi ad AWS IoT ed eseguita su di essi. Puoi ad esempio definire un processo che indichi a un insieme di dispositivi di scaricare e installare un'applicazione o eseguire aggiornamenti del firmware, eseguire il riavvio, ruotare i certificati o eseguire operazioni di risoluzione dei problemi in remoto.

Documento del processo

Per creare un processo, devi prima creare un documento del processo che sia una descrizione delle operazioni remote che i dispositivi dovranno eseguire.

I documenti dei processi sono documenti JSON con codifica UTF-8 che contengono le informazioni necessarie ai dispositivi per eseguire un processo. Un documento di lavoro contiene uno o più documenti URLs in cui il dispositivo può scaricare un aggiornamento o altri dati. Il documento del processo può essere archiviato in un bucket di Amazon S3 oppure essere incluso inline con il comando che crea il processo.

> **Note**
>
> Nelle risposte MQTT, il `jobDocument` campo è un oggetto JSON. Nelle risposte HTTP, è una rappresentazione in formato stringa dell'oggetto JSON.

Target

Quando crei un processo, devi specificare un elenco di target che sono i dispositivi che devono eseguire le operazioni. I target possono essere oggetti, gruppi di oggetti o entrambi. Il servizio AWS IoT Jobs invia un messaggio a ciascuna destinazione per informarla della disponibilità di un lavoro.

Distribuzione

Dopo avere creato un processo fornendo il documento di processo e specificato l'elenco di destinazioni, il documento di processo viene quindi distribuito sui dispositivi di destinazione remoti per i quali desideri eseguire l'aggiornamento. Per i processi snapshot, il processo viene completato dopo l'implementazione nei dispositivi di destinazione. Per i processi continui, il processo viene implementato su un gruppo di dispositivi quando questi vengono aggiunti ai gruppi.

Esecuzione del processo

L'esecuzione di un processo è un'istanza di un processo su un dispositivo target. Il target avvia l'esecuzione di un processo scaricando il documento del processo. Quindi esegue le operazioni specificate nel documento e ne segnala lo stato di avanzamento a AWS IoT. Un numero di esecuzione è un identificatore univoco dell'esecuzione di un processo su un determinato target. Il servizio AWS IoT Jobs fornisce comandi per tenere traccia dell'avanzamento dell'esecuzione di un lavoro su un obiettivo e dell'avanzamento di un processo su tutte le destinazioni.

Concetti relativi ai tipi di processo

I seguenti concetti possono aiutarti a comprendere meglio i diversi tipi di lavori che puoi creare con AWS IoT Jobs.

Processo di snapshot

Per impostazione predefinita, un processo viene inviato a tutti i target specificati al momento della creazione del processo. Dopo che i target completano il processo (o segnalano l'impossibilità di farlo), il processo è completato.

Processo continuo

Un processo continuo viene inviato a tutti i target specificati al momento della creazione del processo. Viene eseguito in modo continuo e inviato a tutti i nuovi dispositivi (oggetti) che vengono aggiunti al gruppo target. Ad esempio, un processo continuo può essere usato per eseguire l'onboarding o l'aggiornamento dei dispositivi quando vengono aggiunti a un gruppo. Puoi rendere un processo continuo impostando un parametro opzionale quando crei il processo.

> **Note**
>
> Quando il tuo parco istanze IoT di destinazione è definito utilizzando gruppi di oggetti dinamici, ti consigliamo di usare processi continui anziché i processi di snapshot.

Utilizzando i processi continui, i dispositivi che si uniscono al gruppo ricevono l'esecuzione del processo anche dopo la creazione del processo.

Predefinito URLs

Per un accesso sicuro e limitato nel tempo ai dati non inclusi nel documento di lavoro, puoi utilizzare Amazon S3 prefirmato. URLs Puoi inserire i dati in un bucket Amazon S3 e aggiungere un collegamento segnaposto ai dati nel documento del processo. Quando AWS IoT Jobs riceve una richiesta per il documento di lavoro, analizza il documento di lavoro cercando i link segnaposto, quindi sostituisce i link con Amazon S3 prefirmato. URLs

Il collegamento segnaposto ha il formato seguente:

```
${aws:iot:s3-presigned-url:https://s3.amazonaws.com/bucket/key}
```

*bucket*dov'è il nome del bucket e *key* l'oggetto nel bucket a cui ti stai collegando.

Nelle regioni di Pechino e Ningxia, i predefiniti URLs funzionano solo se il proprietario della risorsa dispone di una licenza ICP (Internet Content Provider). Per ulteriori informazioni, consulta Amazon Simple Storage Service nella documentazione Getting Started with AWS Services in Cina.

Concetti relativi alla configurazione dei processi

I concetti seguenti sono utili per comprendere come configurare processi.

Rollouts

Puoi specificare con che velocità vengono inviate ai target le notifiche relative all'esecuzione di un processo in sospeso. In questo modo, è possibile creare un'implementazione per fasi, per gestire meglio aggiornamenti, riavvii e altre operazioni. È possibile creare una configurazione di rollout utilizzando una velocità di rollout statica o esponenziale. Per specificare il numero massimo di obiettivi di processo a cui inviare notifiche ogni minuto, utilizza una velocità di rollout costante.

Per esempi di impostazione delle frequenze di implementazione e ulteriori informazioni sull'implementazione del processo di configurazione, consulta Configurazioni di rollout, pianificazione e interruzione dei processi.

Pianificazione

La pianificazione del processo consente di pianificare il periodo di tempo di implementazione di un documento del processo in tutti i dispositivi del gruppo di destinazione per processi continui e snapshot. Inoltre, è possibile creare una finestra di manutenzione opzionale contenente date e ore specifiche in cui un processo avvierà la distribuzione del documento del processo in tutti i dispositivi del gruppo di destinazione. Una finestra di manutenzione è un'istanza ricorrente con una frequenza di date e ore giornaliera, settimanale, mensile o personalizzata selezionata durante la creazione del processo iniziale o del modello di processo. È possibile pianificare solo processi continui per eseguire una distribuzione durante una finestra di manutenzione.

La pianificazione dei processo è specifica del processo. Le singole esecuzioni del processo non possono essere pianificate. Per ulteriori informazioni, consulta Configurazioni di rollout, pianificazione e interruzione dei processi.

Interruzione

È possibile creare un insieme di condizioni per interrompere i rollout quando sono soddisfatti i criteri specificati. Per ulteriori informazioni, consulta Configurazioni di rollout, pianificazione e interruzione dei processi.

Timeout

I timeout del processo ti informano ogni volta che l'implementazione di un processo si blocca in stato `IN_PROGRESS` per un periodo di tempo inaspettatamente lungo. Sono disponibili due tipi di timer: in corso e della fase. Quando il processo è `IN_PROGRESS`, è possibile monitorare e tracciare il progresso dell'implementazione del processo.

Le configurazioni di rollout e interruzione sono specifiche del processo, mentre la configurazione di timeout è specifica dell'implementazione del processo. Per ulteriori informazioni, consulta Configurazioni di timeout e di nuovo tentativo di esecuzione del processo.

Tentativi

Le ripetizioni di tentativi per i processi permettono di riprovare l'esecuzione del processo in caso di errore e/o timeout del processo. Puoi definire fino a 10 nuovi tentativi di esecuzione del processo. È possibile monitorare e tracciare l'avanzamento del nuovo tentativo e verificare se l'esecuzione del processo è riuscita.

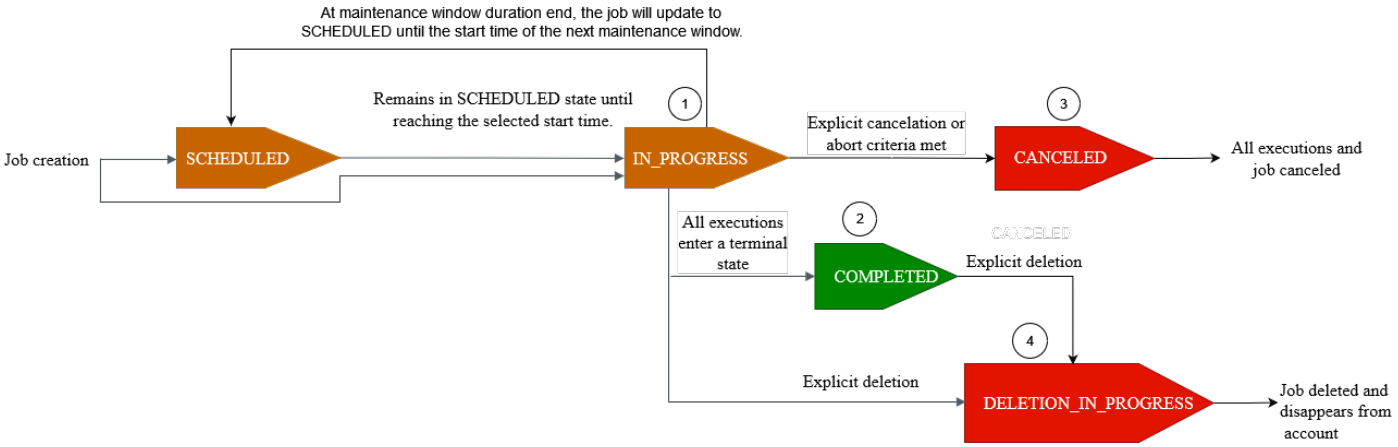
Le configurazioni di rollout e interruzione sono specifiche del processo, mentre le configurazioni di timeout e ripetizione dei tentativi sono specifiche dell'esecuzione del processo. Per ulteriori informazioni, consulta Configurazioni di timeout e di nuovo tentativo di esecuzione del processo.

Processi e stati di esecuzione dei processi

Le seguenti sezioni descrivono il ciclo di vita di un AWS IoT job e il ciclo di vita dell'esecuzione di un job.

Job stati

Il diagramma seguente mostra i diversi stati di un AWS IoT lavoro.



Un lavoro creato utilizzando AWS IoT Jobs può trovarsi in uno dei seguenti stati:

- SCHEDULED

  Durante la creazione iniziale del lavoro o del modello di lavoro utilizzando la AWS IoT console, CreateJobl'CreateJobTemplateAPI o l'API, puoi selezionare la configurazione di pianificazione opzionale nella AWS IoT console o `SchedulingConfig` nell'CreateJobAPI o nell'CreateJobTemplateAPI. Quando si avvia un processo pianificato contenente parametri `startTime`, `endTime` e `endBehavior` specifici, lo stato del processo viene aggiornato a `SCHEDULED`. Quando il processo raggiunge l'`startTime` selezionata o l'`startTime` della finestra di manutenzione successiva (se hai selezionato l'implementazione del processo durante una finestra di manutenzione), lo stato verrà aggiornato da `SCHEDULED` a `IN_PROGRESS` e inizierà il rollout del documento del processo a tutti i dispositivi del gruppo di destinazione.
- IN_PROGRESS

  Quando si crea un lavoro utilizzando la AWS IoT console o l'CreateJobAPI, lo stato del processo viene aggiornato a`IN_PROGRESS`. Durante la creazione di un processo, AWS IoT Jobs inizia a implementare le esecuzioni del processo nei dispositivi del gruppo di destinazione. Una volta implementate tutte le esecuzioni del processo, AWS IoT Jobs attende che i dispositivi completino l'operazione remota.

Per informazioni sulla simultaneità e sui limiti applicabili ai processi in corso, consulta AWS IoT Limiti dei lavori.

> **Note**
>
> Quando un processo `IN_PROGRESS` raggiunge la fine della finestra di manutenzione corrente, la distribuzione del documento del processo si interrompe. Il processo verrà aggiornato a `SCHEDULED` fino alla `startTime` della finestra di manutenzione successiva.

- COMPLETED

  Un processo continuo viene gestito in uno dei seguenti modi:

  - Un processo continuo senza la configurazione della pianificazione opzionale selezionata è sempre in corso e continua a essere eseguito per qualsiasi nuovo dispositivo che viene aggiunto al gruppo di destinazione. Non raggiungerà mai uno stato di `COMPLETED`.
  - Per un processo continuo con la configurazione della pianificazione opzionale selezionata, vale quanto segue:
    - Se è stata specificata una `endTime`, un processo continuo raggiungerà lo stato `COMPLETED` quando `endTime` è passata e tutte le esecuzioni del processo hanno raggiunto uno stato terminale.
    - Se non è stata specificata una `endTime` nella configurazione della pianificazione opzionale, il processo continuo continuerà a eseguire la distribuzione del documento del processo.

  Per un processo snapshot, lo stato del processo diventa `COMPLETED` quando tutte le esecuzioni del processo raggiungono uno stato terminale, come `SUCCEEDED`, `FAILED`, `TIMED_OUT`, `REMOVED` o `CANCELED`.

- CANCELED (ANNULLATO)

  Quando annulli un lavoro utilizzando la AWS IoT console, l'CancelJobAPI o ilConfigurazione dell'interruzione del processo, lo stato del lavoro cambia in`CANCELED`. Durante l'annullamento del lavoro, AWS IoT Jobs inizia ad annullare le esecuzioni di lavori create in precedenza.

  Per informazioni sulla simultaneità e sui limiti applicabili ai processi che sono in fase di cancellazione, consulta AWS IoT Limiti dei lavori.

- DELETION_IN_PROGRESS

Quando elimini un lavoro utilizzando la AWS IoT console o l'DeleteJobAPI, lo stato del lavoro cambia in. `DELETION_IN_PROGRESS` Durante l'eliminazione dei AWS IoT job, Jobs inizia a eliminare le esecuzioni di job create in precedenza. Dopo che tutte le esecuzioni di job sono state eliminate, il job scompare dal tuo account. AWS

Stati di esecuzione del processo

La tabella seguente mostra i diversi stati di esecuzione di un AWS IoT job e se la modifica dello stato viene avviata dal dispositivo o da Jobs. AWS IoT

Origine e stati di esecuzione del processo

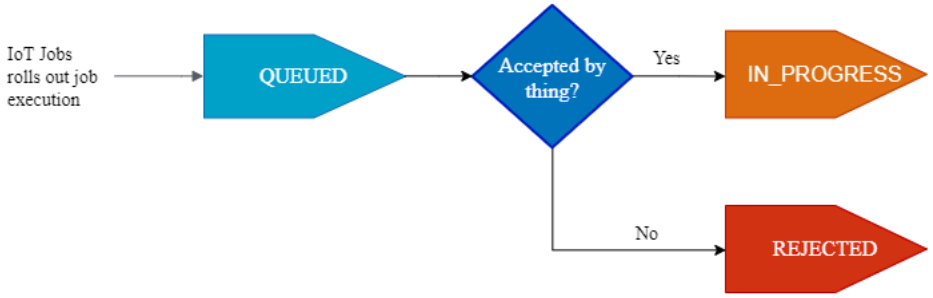
Stato di esecuzione del processo	Iniziato dal dispositivo?	Avviato da Jobs AWS IoT ?	Stato del terminale?	Può essere riprovato?
`QUEUED`	No	Sì	No	Non applicabile
`IN_PROGRESS`	Sì	No	No	Non applicabile
`SUCCEEDED`	Sì	No	Sì	Non applicabile
`FAILED`	Sì	No	Sì	Sì
`TIMED_OUT`	No	Sì	Sì	Sì
`REJECTED`	Sì	No	Sì	No
`REMOVED`	No	Sì	Sì	No
`CANCELED`	No	Sì	Sì	No

La sezione seguente descrive ulteriori informazioni sugli stati di esecuzione di un lavoro che viene implementato quando si crea un lavoro con AWS IoT Jobs.

- IN CODA

Quando AWS IoT Jobs implementa un'esecuzione di lavoro per un dispositivo di destinazione, lo stato di esecuzione del lavoro è impostato su`QUEUED`. L'esecuzione del processo resta nello stato `QUEUED` finché non si verifica quanto segue:

- Il dispositivo riceve l'esecuzione del processo, richiama le operazioni API Jobs e segnala lo stato come `IN_PROGRESS`.
- Quando annulli correttamente il processo o l'esecuzione del processo o vengono soddisfatti i criteri che hai specificato usando una configurazione di interruzione, lo stato diventa `CANCELED`.
- Il dispositivo viene rimosso dal gruppo di destinazione e lo stato diventa `REMOVED`.



- IN_PROGRESS

  Se il dispositivo IoT si abbona alla rete riservata Argomenti di processo `$notify` e `$notify-next` il dispositivo richiama l'`StartNextPendingJobExecution`API o l'`UpdateJobExecution`API con uno stato di`IN_PROGRESS`, AWS IoT Jobs imposterà lo stato di esecuzione del processo su. `IN_PROGRESS`

  L'API `UpdateJobExecution` può essere richiamata più volte con lo stato `IN_PROGRESS`. Puoi specificare maggiori dettagli sulle fasi di esecuzione utilizzando l'oggetto `statusDetails`.

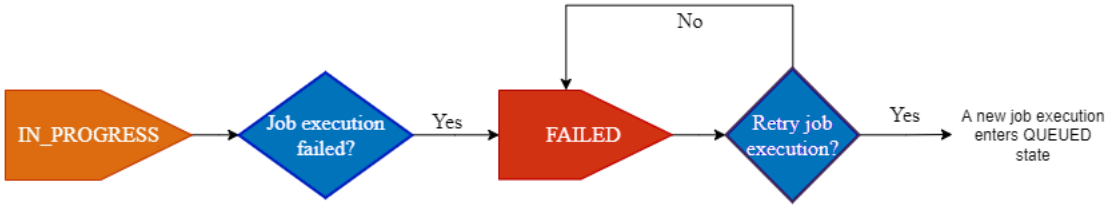
  > **Note**
  >
  > Se crei più lavori per ogni dispositivo, AWS IoT Jobs e il protocollo MQTT non garantiscono l'ordine di consegna.

- RIUSCITA

  Quando il dispositivo completa correttamente l'operazione remota, deve richiamare l'`UpdateJobExecution`API con uno stato pari `SUCCEEDED` a per indicare che l'esecuzione del processo è riuscita. AWS IoT Jobs quindi aggiorna e restituisce lo stato di esecuzione del lavoro come. `SUCCEEDED`

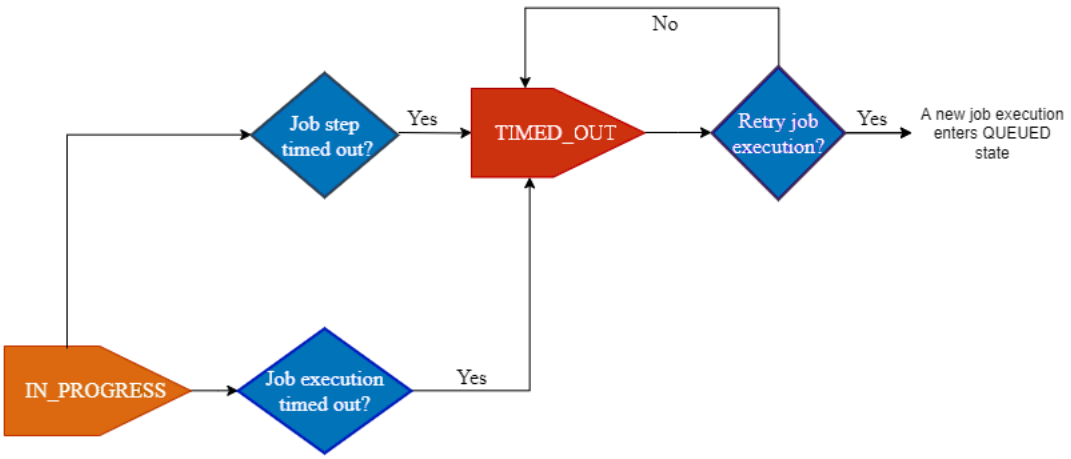
- NON RIUSCITO

  Quando il dispositivo non riesce a completare l'operazione remota, deve richiamare l'`UpdateJobExecution`API con uno stato pari `Failed` a per indicare che l'esecuzione del processo non è riuscita. AWS IoT Jobs quindi aggiorna e restituisce lo stato di esecuzione del lavoro come`Failed`. Puoi riprovare l'esecuzione del processo per il dispositivo usando la Configurazione del nuovo tentativo di esecuzione del processo.



- TIMED_OUT

  Quando il dispositivo non riesce a completare una fase di lavoro quando lo stato è`IN_PROGRESS`, o quando non riesce a completare l'operazione remota entro la durata del timeout del timer in corso, AWS IoT Jobs imposta lo stato di esecuzione del lavoro su. `TIMED_OUT` Hai anche un timer per ogni fase di un processo in corso che si applica solo all'esecuzione del processo. La durata del timer in corso viene specificata usando la proprietà `inProgressTimeoutInMinutes` della Configurazione del timeout di esecuzione del processo. Puoi riprovare l'esecuzione del processo per il dispositivo usando la Configurazione del nuovo tentativo di esecuzione del processo.



- REJECTED

  Quando il dispositivo riceve una richiesta non valida o incompatibile, deve richiamare l'`UpdateJobExecution`API con uno stato di. `REJECTED` AWS IoT Jobs quindi aggiorna e restituisce lo stato di esecuzione del processo come. `REJECTED`
- REMOVED

  Quando il dispositivo non è più una destinazione valida per l'esecuzione del processo, ad esempio quando è scollegato da un gruppo di oggetti dinamico, AWS IoT Jobs imposta lo stato di esecuzione del processo su `REMOVED`. Puoi ricollegare l'oggetto al gruppo di destinazione e riavviare l'esecuzione del processo per il dispositivo.
- CANCELED (ANNULLATO)

  Quando si annulla un lavoro o si annulla l'esecuzione di un lavoro utilizzando la console `CancelJob` o l'`CancelJobExecution`API o o quando vengono soddisfatti i criteri di interruzione specificati utilizzando ilConfigurazione dell'interruzione del processo, AWS IoT Jobs annulla il lavoro e imposta lo stato di esecuzione del lavoro su. `CANCELED`

Gestione dei processi

Utilizza i processi per inviare notifica ai dispositivi riguardo un aggiornamento del software o del firmware. È possibile utilizzare la AWS IoT console, il Operazioni API di gestione e controllo dei processi AWS Command Line Interface, o il AWS SDKsper creare e gestire i lavori.

Firma del codice per i processi

Quando si invia un codice ai dispositivi affinché questi rilevino se il codice è stato modificato durante il transito, si consiglia di firmare il file di codice utilizzando la AWS CLI. Per istruzioni, consulta Creazione e gestione di processi utilizzando la AWS CLI.

Per ulteriori informazioni, vedi A cosa serve la firma del codice AWS IoT? .

Documento di processo

Prima di creare un processo, è necessario creare un documento del processo. Se utilizzi la firma del codice per AWS IoT, devi caricare il documento di lavoro in un bucket Amazon S3 con versione. Per informazioni sulla creazione e il caricamento di un file in un bucket Amazon S3, consulta la sezione Nozioni di base su Amazon Simple Storage Service nella Guida alle nozioni di base su Amazon S3.

> **Tip**
>
> Per esempi di documenti di lavoro, consulta l'esempio jobs-agent.js nell'SDK per. AWS IoT JavaScript

Predefinito URLs

Il documento di processo può contenere un URL Amazon s3 prefirmato che punta al file del codice (o ad altri file). Le Amazon URLs S3 prefirmate sono valide solo per un periodo di tempo limitato e vengono generate quando un dispositivo richiede un documento di lavoro. Poiché l'URL prefirmato non è stato creato durate la creazione del documento di processo, è necessario inserire un URL segnaposto nel documento di processo. Un URL segnaposto ha il seguente aspetto:

```
${aws:iot:s3-presigned-url-v2:https://
s3.region.amazonaws.com/<bucket>/<code file>}
```

dove:

- *`bucket`*è il bucket Amazon S3 che contiene il file di codice.
- *`code file`*è la chiave Amazon S3 del file di codice.

Quando un dispositivo richiede il documento di lavoro, AWS IoT genera l'URL predefinito e sostituisce l'URL segnaposto con l'URL prefirmato. Il documento di processo viene quindi inviato al dispositivo.

Ruolo IAM per concedere l'autorizzazione a scaricare file da S3

Quando crei un lavoro che utilizza Amazon URLs S3 predefinito, devi fornire un ruolo IAM. Il ruolo deve concedere l'autorizzazione per scaricare file dal bucket Amazon S3 in cui vengono archiviati i dati o gli aggiornamenti del processo. Il ruolo deve anche concedere ad AWS IoT l'autorizzazione per assumere il ruolo.

È possibile specificare un timeout opzionale per l'URL prefirmato. Per ulteriori informazioni, consulta CreateJob.

Concedi AWS IoT a Jobs il permesso di assumere il tuo ruolo

1. Vai a Hub dei ruoli della console IAM e scegli il ruolo.

2. Nella scheda Relazioni di trust, seleziona Modifica relazione di trust e sostituisci il documento della policy con il JSON seguente. Scegli Update Trust Policy (Aggiorna policy di trust).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iot.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. Per prevenire il problema "confused deputy", aggiungi le chiavi di contesto di condizione globali `aws:SourceArn` e `aws:SourceAccount` alla policy.

> **Important**
>
> `aws:SourceArn` deve essere conforme al formato: `arn:aws:iot:`*`region`*`:`*`account-id`*`:*`. Assicurati che *`region`* corrisponda alla tua AWS IoT regione e *`account-id`* all'ID del tuo account cliente. Per ulteriori informazioni consulta la pagina relativa alla prevenzione del problema "confused deputy" tra servizi.

```
{
  "Effect": "Allow",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service":
          "iot.amazonaws.com"
       },
      "Action": "sts:AssumeRole",
```

```
      "Condition": {
         "StringEquals": {
            "aws:SourceAccount": "123456789012"
         },
         "ArnLike": {
              "aws:SourceArn": "arn:aws:iot:*:123456789012:job/*"
         }
       }
     }
   ]
}
```

4. Se il processo utilizza un documento del processo che è un oggetto Amazon S3, seleziona Permissions (Autorizzazioni) e utilizza il JSON seguente. In questo modo viene aggiunta una policy che concede l'autorizzazione per scaricare file dal bucket Amazon S3:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::your_S3_bucket/*"
        }
    ]
}
```

URL predefinito per il caricamento dei file

Se i tuoi dispositivi devono caricare file su un bucket Amazon S3 durante l'implementazione di un processo, puoi includere il seguente segnaposto URL predefinito nel documento di lavoro:

```
${aws:iot:s3-presigned-url-upload:https://s3.region.amazonaws.com/<bucket>/<key>}
```

Puoi utilizzare un massimo di due parole chiave per ciascuna di `${thingName}` esse e `${executionNumber}` come parole chiave riservate all'interno dell'`key`attributo nell'URL segnaposto per il caricamento del file che si trova nel documento di lavoro. `${jobId}` Il segnaposto locale che rappresenta le parole chiave riservate nell'`key`attributo verrà analizzato e sostituito al momento della creazione dell'esecuzione del lavoro. L'utilizzo di un segnaposto locale con parole

chiave riservate specifiche per ogni dispositivo garantisce che ogni file caricato da un dispositivo sia specifico per quel dispositivo e non venga sovrascritto da un file simile caricato da un altro dispositivo destinato allo stesso processo di implementazione. Per informazioni sulla risoluzione dei problemi relativi ai segnaposto locali all'interno di un segnaposto URL predefinito per il caricamento di file durante l'implementazione di un processo, consulta. Messaggi di errore generali sulla risoluzione dei problemi

> **Note**
>
> Il nome del bucket Amazon S3 non può contenere il segnaposto locale che rappresenta le parole chiave riservate per il file caricato. Il segnaposto locale deve trovarsi nell'attributo. `key`

Questo segnaposto URL predefinito verrà convertito in un URL di caricamento predefinito di Amazon S3 nel documento di lavoro quando un dispositivo lo riceve. I tuoi dispositivi lo useranno per caricare file su un bucket Amazon S3 di destinazione.

> **Note**
>
> Quando il bucket e la chiave Amazon S3 non sono forniti nell'URL segnaposto sopra riportato, AWS IoT Jobs genererà automaticamente una chiave per ogni dispositivo utilizzando un massimo di due di, e. `${thingName}` `${jobId}` `${executionNumber}`

URL predefinito utilizzando il controllo delle versioni di Amazon S3

La salvaguardia dell'integrità di un file archiviato in un bucket Amazon S3 è fondamentale per garantire implementazioni sicure che utilizzino quel file nella tua flotta di dispositivi. Con l'uso del controllo delle versioni di Amazon S3, puoi aggiungere un identificatore di versione per ogni variante del file archiviata nel tuo bucket Amazon S3 per tracciare ogni versione del file. Ciò fornisce informazioni sulla versione del file distribuita nel tuo parco dispositivi utilizzando Jobs. AWS IoT Per ulteriori informazioni sui bucket Amazon S3 che utilizzano il controllo delle versioni, consulta Utilizzo del controllo delle versioni nei bucket Amazon S3.

Se il file è archiviato in Amazon S3 e il documento di lavoro contiene un segnaposto URL predefinito, AWS IoT Jobs genererà un URL predefinito nel documento di lavoro utilizzando il bucket Amazon S3, la chiave del bucket e la versione del file archiviata nel bucket Amazon S3. Questo URL predefinito generato nel documento di lavoro sostituirà il segnaposto URL predefinito

originariamente presente nel documento di lavoro. Se aggiorni il file archiviato nel tuo bucket Amazon S3, `versionId` verranno create una nuova versione del file e una versione successiva per segnalare gli aggiornamenti effettuati e fornire la possibilità di indirizzare quel file specifico nelle future implementazioni di lavoro.

Fai riferimento ai seguenti esempi per vedere prima e durante l'utilizzo di Amazon S3 prefirmato URLs nel documento di lavoro utilizzando: `versionId`

Segnaposto URL predefinito Amazon S3 (prima del Job Deployment)

```
//Virtual-hosted style URL
${aws:iot:s3-presigned-url-v2:https://bucket-name.s3.region-code.amazonaws.com/key-name%3FversionId%3Dversion-id}

//Path-style URL
${aws:iot:s3-presigned-url-v2:https://s3.region-code.amazonaws.com/bucket-name/key-name%3FversionId%3Dversion-id}
```

URL predefinito Amazon S3 (durante la distribuzione di Job)

```
//Virtual-hosted style URL
${aws:iot:s3-presigned-url-v2:https://sample-bucket-name.s3.us-west-2.amazonaws.com/sample-code-file.png%3FversionId%3Dversion1}

//Path-style
${aws:iot:s3-presigned-url-v2:https://s3.us-west-2.amazonaws.com/sample-bucket-name/sample-code-file.png%3FversionId%3Dversion1}
```

Per ulteriori informazioni sugli oggetti ospitati virtualmente e in stile path di Amazon S3 URLs, consulta richieste e richieste in stile Path. Virtual-hosted-style

> **Note**
>
> Se desideri aggiungere un URL predefinito `versionId` di Amazon S3, deve essere conforme al supporto della codifica URL. AWS SDK for Java 2.x Per ulteriori informazioni, consulta Modifiche nell'analisi di Amazon URIs S3 dalla versione 1 alla versione 2.

Differenze tra le versioni dei segnaposto URL predefiniti di Amazon S3

L'elenco seguente descrive le differenze tra i `${aws:iot:s3-presigned-url-v1` segnaposto URL predefiniti di Amazon S3 (versione 1) e (versione 2)`${aws:iot:s3-presigned-url-v2`:

- Il segnaposto `${aws:iot:s3-presigned-url-v1` URL predefinito di Amazon S3 non è supportato. `version-id`
- Il segnaposto URL predefinito di Amazon S3 riceve l'URL di `${aws:iot:s3-presigned-url-v1` Amazon S3 come non codificato. Il segnaposto Amazon S3 Presigned URL richiede che l'URL di `${aws:iot:s3-presigned-url-v2` Amazon S3 sia codificato in modo che sia conforme allo standard Amazon S3 SDK.

Argomenti

- Crea e gestisci lavori utilizzando Console di gestione AWS
- Crea e gestisci i lavori utilizzando il AWS CLI

Crea e gestisci lavori utilizzando Console di gestione AWS

Questa sezione descrive come creare e gestire i lavori dalla AWS IoT console. Dopo aver creato un lavoro, puoi visualizzare le informazioni sul lavoro nella pagina dei dettagli e gestirlo.

> **Note**
>
> Se desideri eseguire la firma del codice per i AWS IoT lavori, usa il AWS CLI. Per ulteriori informazioni, vedere Creare e gestire i lavori utilizzando AWS CLI.

Argomenti

- Crea e gestisci lavori utilizzando il Console di gestione AWS
- Visualizza e gestisci i lavori utilizzando il Console di gestione AWS

Crea e gestisci lavori utilizzando il Console di gestione AWS

Per creare un lavoro, accedi alla AWS IoT console e vai all'hub Jobs nella sezione Azioni remote. Quindi, esegui i seguenti passaggi.

1. Nella pagina Jobs, nella finestra di dialogo Jobs, scegli Crea lavoro.

2. A seconda del dispositivo che stai utilizzando, puoi creare un lavoro personalizzato, un processo di aggiornamento OTA FreerTOS o un lavoro. AWS IoT Greengrass Per questo esempio, scegli Crea un processo personalizzato. Scegli Next (Successivo).
3. Nella pagina Custom job properties (Proprietà del processo personalizzato), nella finestra di dialogo Job properties (Proprietà del processo), inserisci le informazioni per i seguenti campi:

   - Name (Nome): inserisci un nome alfanumerico univoco per il processo.
   - Description - optional (Descrizione - opzionale): inserisci una descrizione opzionale del processo.
   - Tags - optional (Tag - opzionale):

   > **Note**
   >
   > Ti consigliamo di non utilizzare informazioni di identificazione personale nel lavoro IDs e nella descrizione.

   Scegli Next (Successivo).
4. Nella pagina File configuration (Configurazione dei file) nella finestra di dialogo Job targets (Destinazioni del processo), seleziona Things (Oggetti) o Thing groups (Gruppi di oggetti) per i quali si desidera eseguire questo processo.

   Nella finestra di dialogo Job document (Documento del processo), seleziona una delle seguenti opzioni:

   - From file (Dal file): un file dell'attività JSON caricato in precedenza in un bucket Amazon S3
     - Firma del codice

       Nel documento del processo che si trova nell'URL Amazon S3, `${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}` è obbligatorio come segnaposto finché non viene sostituito dal percorso del file di codice firmato utilizzando il Profilo di firma del codice. Il nuovo file di codice firmato verrà inizialmente visualizzato in una cartella `SignedImages` nel bucket di origine di Amazon S3. Verrà creato un nuovo documento del processo contenente un prefisso `Codesigned_` con il percorso del file di codice firmato che sostituisce il segnaposto di firma del codice e inserito nell'URL Amazon S3 per creare un nuovo processo.

- Risorsa pre-firmata URLs

  Nel menu a discesa Ruolo di pre-firma, scegli il ruolo IAM che hai creato in Presigned. URLs L'utilizzo `${aws:iot:s3-presigned-url:` di to presign URLs per oggetti che si trovano in Amazon S3 è una best practice di sicurezza per i dispositivi che scaricano oggetti da Amazon S3.

  Se desideri utilizzare presigned come segnaposto URLs per la firma del codice, usa il seguente modello di esempio:

  ```
  ${aws:iot:s3-presigned-url:${aws:iot:code-sign-signature:<S3 URL>}
  ```

- From template (Dal modello): un modello di processo contenente un documento del processo e le configurazioni del processo. Il modello di lavoro può essere un modello di lavoro personalizzato creato dall'utente o un AWS modello gestito.

  Se stai creando un lavoro per eseguire azioni remote utilizzate di frequente, come il riavvio del dispositivo, puoi utilizzare un modello AWS gestito. Questi modelli sono già stati preconfigurati per l'uso. Per ulteriori informazioni, consultare Crea di un modello di processo personalizzato e Creazione di modelli di processo personalizzati partendo da modelli gestiti.

5. Nella pagina Job configuration (Configurazione del processo) nella finestra di dialogo Job configuration (Configurazione del processo), seleziona uno dei seguenti tipi di processo:

   - Snapshot job (Processo snapshot): un processo snapshot viene completato quando termina la sua esecuzione su dispositivi e gruppi di destinazione.
   - Continuous job (Processo continuo): un processo continuo si applica ai gruppi di oggetti e viene eseguito su qualsiasi dispositivo aggiunto successivamente a un gruppo di destinazione specificato.

6. Nella finestra di dialogo Additional configurations - optional (Configurazioni aggiuntive - opzionale), esamina le seguenti configurazioni del processo opzionali ed esegui le selezioni di conseguenza:

   - Rollout configuration (Configurazione rollout)
   - Scheduling configuration (Configurazione della pianificazione)
   - Job executions timeout configuration (Configurazione del timeout di esecuzioni dei processi)
   - Job executions retry configuration - new (Configurazione dei tentativi delle esecuzioni dei processi - nuovo)

- Abort configuration (Configurazione dell'interruzione)

Per ulteriori informazioni sulle configurazioni del processo, fai riferimento alle seguenti sezioni:

- Configurazioni di rollout, pianificazione e interruzione dei processi
- Configurazioni di timeout e di nuovo tentativo di esecuzione del processo

Esamina tutte le selezioni del processo e quindi scegli Submit (Invia) per creare il processo.

Visualizza e gestisci i lavori utilizzando il Console di gestione AWS

Dopo aver creato il processo, la console genera una firma JSON e la inserisce nel documento di processo. Puoi utilizzare la console AWS IoT per visualizzare lo stato di un processo, annullare o eliminare il processo.

Se scegli il lavoro che hai creato, puoi trovare:

- Dettagli generali del lavoro, come il nome, la descrizione, il tipo, l'ora in cui è stato creato, l'ultimo aggiornamento e l'ora di inizio prevista.
- Tutte le configurazioni di lavoro specificate e il relativo stato.
- Documento del processo.
- Le esecuzioni dei lavori e gli eventuali tag facoltativi specificati.

Per gestire i lavori, vai al Job hub della console e scegli se modificare, eliminare o annullare il lavoro.

Crea e gestisci i lavori utilizzando il AWS CLI

Questa sezione illustra come creare e gestire i processi.

Creazione di processi

Per creare un AWS IoT lavoro, utilizzare il CreateJob comando. Il processo viene accodato per l'esecuzione nei target (oggetti o gruppi di oggetti) specificati. Per creare un AWS IoT lavoro, è necessario un documento di lavoro che può essere incluso nel corpo della richiesta o come collegamento a un documento Amazon S3. Se il processo include il download di file utilizzando Amazon URLs S3 predefinito, è necessario un ruolo IAM Amazon Resource Name (ARN) che

disponga dell'autorizzazione per scaricare il file e conceda l'autorizzazione al servizio Jobs per assumere AWS IoT il ruolo.

Per ulteriori informazioni sulla sintassi da utilizzare per l'immissione di data e ora utilizzando un comando API o il, consulta Timestamp. AWS CLI

Firma del codice con i processi

Se utilizzi la firma del codice per AWS IoT, devi avviare un processo di firma del codice e includere l'output nel documento di lavoro. Questo sostituirà il segnaposto di firma del codice nel documento del processo, che è richiesto come segnaposto finché non viene sostituito da percorso del file di codice firmato utilizzando il profilo di firma del codice. Il segnaposto di firma del codice sarà simile a quanto segue:

```
    ${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}
```

Usa il start-signing-jobcomando per creare un processo di firma del codice. `start-signing-job`brestituisce un ID di lavoro. Utilizza il comando describe-signing-job per ottenere la posizione di Amazon S3 in cui è archiviata la firma. È possibile eseguire il download della firma da Amazon S3. Per ulteriori informazioni sui processi di firma del codice, consulta Firma del codice per AWS IoT.

Il documento del processo deve contenere un segnaposto URL prefirmato per il file del codice e l'output di firma JSON inserito in un bucket Amazon S3 utilizzando il comando start-signing-job:

```
{
    "presign": "${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/bucket/
image}",
}
```

Crea un processo con un documento del processo

Il comando seguente mostra come creare un lavoro utilizzando un documento di lavoro (*job-document.json*) archiviato in un bucket Amazon S3 (*jobBucket*) e un ruolo con autorizzazione a scaricare file da Amazon S3 (). *S3DownloadRole*

```
aws iot create-job  \
      --job-id 010  \
      --targets arn:aws:iot:us-east-1:123456789012:thing/thingOne  \
      --document-source https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json
 \
```

```
      --timeout-config inProgressTimeoutInMinutes=100 \
      --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute
\": 50, \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings
\": 1000, \"numberOfSucceededThings\": 1000}}, \"maximumPerMinute\": 1000}" \
      --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
      --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

Il processo viene eseguito. *thingOne*

Il parametro `timeout-config` opzionale specifica l'intervallo di tempo a disposizione di ciascun dispositivo per terminare l'esecuzione del processo. Il timer viene avviato quando imposti lo stato di esecuzione del processo su `IN_PROGRESS`. Se lo stato di esecuzione del processo non è impostato su un altro stato terminale prima della scadenza del tempo a disposizione, viene impostato su `TIMED_OUT`.

Il timer in corso non può essere aggiornato e viene applicato a tutte le esecuzioni del processo. Ogni volta che l'esecuzione di un job rimane nello `IN_PROGRESS` stato per un periodo superiore a questo intervallo, fallisce e passa allo stato del terminale`TIMED_OUT`. AWS IoT pubblica anche una notifica MQTT.

Per ulteriori informazioni sulla creazione delle configurazioni dei rollout e delle interruzioni di processo, consulta la sezione relativa alla Configurazione dei rollout e delle interruzioni di processo.

> **Note**
>
> I documenti dei processi specificati come file Amazon S3 vengono recuperati al momento della creazione del processo. Se modifichi i contenuti del file Amazon S3 usato come origine del documento del processo dopo la creazione del processo, ciò che viene inviato ai destinatari del processo non cambia.

Aggiornamento di un processo

Utilizza il comando UpdateJob per aggiornare un processo. È possibile aggiornare i campi `description`, `presignedUrlConfig`, `jobExecutionsRolloutConfig`, `abortConfig` e `timeoutConfig` di un processo.

```
aws iot update-job  \
  --job-id 010  \
  --description "updated description" \
  --timeout-config inProgressTimeoutInMinutes=100 \
  --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\": 50,
 \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\": 1000,
 \"numberOfSucceededThings\": 1000}, \"maximumPerMinute\": 1000}}" \
  --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
  --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

Per ulteriori informazioni, consulta Configurazione dei rollout e delle interruzioni di processo.

Annullamento di un processo

Utilizza il comando CancelJob per annullare un processo. L'annullamento di un lavoro AWS IoT impedisce l'implementazione di nuove esecuzioni di lavoro per il lavoro. Annulla anche tutte le esecuzioni di lavoro che si verificano in uno stato. QUEUED AWS IoT mantiene inalterate le esecuzioni di lavoro in uno stato terminale perché il dispositivo ha già completato il lavoro. Se lo stato di esecuzione di un processo è IN_PROGRESS, non verrà modificato, a meno che non si utilizzi il parametro opzionale `--force`.

Il comando seguente mostra come annullare un processo con ID 010.

```
aws iot cancel-job --job-id 010
```

Il comando visualizza il seguente output:

```
{
    "jobArn": "string",
    "jobId": "string",
    "description": "string"
}
```

Quando si annulla un processo, le esecuzioni con stato QUEUED vengono annullate. Le esecuzioni del processo con stato IN_PROGRESS saranno annullate se si specifica il parametro facoltativo `--force`. Le esecuzioni del processo con stato terminale non vengono annullate.

> **Warning**
>
> L'annullamento di un processo con stato `IN_PROGRESS` (impostando il parametro `--force`) annulla tutte le esecuzioni dei processi in corso e impedisce al dispositivo che esegue il processo di aggiornarne lo stato di esecuzione. Prestare attenzione e verificare che tutti i dispositivi in cui è in esecuzione un processo annullato possano effettuare il ripristino a uno stato valido.

Lo stato di un lavoro annullato o di una delle relative esecuzioni di lavoro alla fine è costante. AWS IoT interrompe la pianificazione di nuove esecuzioni di lavori ed esecuzioni di lavori per quel `QUEUED` lavoro sui dispositivi il prima possibile. La modifica dello stato dell'esecuzione di un processo in `CANCELED` potrebbe tuttavia richiedere un po' di tempo, a seconda del numero di dispositivi e di altri fattori.

Se un processo viene annullato perché soddisfa i criteri definiti da un oggetto `AbortConfig`, il servizio aggiunge valori popolati automaticamente per i campi `comment` e `reasonCode`. Puoi creare valori personalizzati per `reasonCode` quando l'annullamento del processo è basato sull'utente.

Annullamento dell'esecuzione di un processo

Utilizza il comando CancelJobExecution per annullare l'esecuzione di un processo su un dispositivo. Questo comando annulla un processo che si trova in uno stato `QUEUED`. Per annullare l'esecuzione di un processo in corso, è necessario utilizzare il parametro `--force`.

Il comando seguente mostra come annullare l'esecuzione del processo 010 su `myThing`.

```
aws iot cancel-job-execution --job-id 010 --thing-name myThing
```

Il comando non visualizza alcun output.

Annulla l'esecuzione di un processo che si trova in uno stato `QUEUED`. L'esecuzione di un processo con stato `IN_PROGRESS` viene annullata se si specifica il parametro opzionale `--force`. Le esecuzioni del processo con stato terminale non possono essere annullate.

> **Warning**
>
> Quando si annulla l'esecuzione di un processo con uno stato `IN_PROGRESS`, il dispositivo non può aggiornarne lo stato di esecuzione del processo. Prestare attenzione e verificare che il dispositivo possa effettuare il ripristino a uno stato valido.

Se l'esecuzione del processo è in uno stato terminale oppure se è nello stato `IN_PROGRESS` e il parametro `--force` non è impostato su `true`, questo comando genera un'eccezione `InvalidStateTransitionException`.

Lo stato dell'esecuzione di un processo annullato è consistente finale. La modifica dello stato dell'esecuzione di un processo in `CANCELED` potrebbe tuttavia richiedere un po' di tempo, a seconda di vari fattori.

Eliminare un processo

Puoi usare il comando DeleteJob per eliminare un processo e le relative esecuzioni. Per impostazione predefinita, è possibile eliminare solo un processo in stato terminale (`SUCCEEDED` o `CANCELED`). In caso contrario, viene generata un'eccezione. È possibile eliminare un processo nello stato `IN_PROGRESS` solo se il parametro `force` è impostato su `true`.

Per eliminare un processo, eseguire il seguente comando:

```
aws iot delete-job --job-id 010 --force|--no-force
```

Il comando non visualizza alcun output.

> **Warning**
>
> Quando si elimina un processo con uno stato `IN_PROGRESS`, il dispositivo che sta eseguendo il processo non è in grado di accedere alle informazioni sul processo o di aggiornarne lo stato di esecuzione. Procedi con cautela e verifica che tutti i dispositivi in cui è in esecuzione un processo annullato possano effettuare il ripristino a uno stato valido.

L'eliminazione di un processo potrebbe richiedere del tempo, a seconda del numero di esecuzioni create per il processo e di altri fattori. Mentre il processo viene eliminato, il suo stato viene indicato

come DELETION_IN_PROGRESS. Il tentativo di eliminare o annullare un processo il cui stato è già DELETION_IN_PROGRESSrestituisce un errore.

Solo 10 processi possono essere nello stato DELETION_IN_PROGRESS nello stesso momento. In caso contrario, si verifica un'eccezione LimitExceededException.

Recupero di un documento del processo

Usa il comando GetJobDocument per recuperare un documento per un processo. Un documento del processo è una descrizione delle operazioni remote che i dispositivi dovranno eseguire.

Esegui il comando seguente per ottenere un documento di processo:

```
aws iot get-job-document --job-id 010
```

Il comando restituisce il documento per il processo specificato:

```
{
    "document": "{\n\t\"operation\":\"install\",\n\t\"url\":\"http://amazon.com/
firmWareUpate-01\",\n\t\"data\":\"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/
amzn-s3-demo-bucket/datafile}\"\n}"
}
```

> **Note**
>
> Quando usi questo comando per recuperare un documento di lavoro, i segnaposto URLs non vengono sostituiti da Amazon S3 prefirmato. URLs Quando un dispositivo richiama l'operazione GetPendingJobExecutionsAPI, i segnaposto URLs vengono sostituiti da Amazon S3 URLs prefirmato nel documento di lavoro.

Elenco dei processi

Per ottenere un elenco di tutti i lavori presenti nel tuo Account AWS, usa il comando. ListJobs I dati del processo e i dati relativi all'esecuzione del processo vengono conservati per un periodo di tempo limitato. Esegui il comando seguente per elencare tutti i lavori presenti nel tuo Account AWS:

```
aws iot list-jobs
```

Il comando restituisce tutti i processi nell'account ordinati in base allo stato:

```
{
    "jobs": [
        {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486687079.743,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/013",
            "createdAt": 1486687079.743,
            "targetSelection": "SNAPSHOT",
            "jobId": "013"
        },
        {
            "status": "SUCCEEDED",
            "lastUpdatedAt": 1486685868.444,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/012",
            "createdAt": 1486685868.444,
            "completedAt": 148668789.690,
            "targetSelection": "SNAPSHOT",
            "jobId": "012"
        },
        {
            "status": "CANCELED",
            "lastUpdatedAt": 1486678850.575,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/011",
            "createdAt": 1486678850.575,
            "targetSelection": "SNAPSHOT",
            "jobId": "011"
        }
    ]
}
```

Descrizione di un processo

Esegui il comando DescribeJob per ottenere lo stato di un processo. Il comando seguente mostra come descrivere un processo:

```
$ aws iot describe-job --job-id 010
```

Il comando restituisce lo stato del processo specificato. Ad esempio:

```
{
    "documentSource": "https://s3.amazonaws.com/amzn-s3-demo-bucket/job-
document.json",
```

```
    "job": {
        "status": "IN_PROGRESS",
        "jobArn": "arn:aws:iot:us-east-1:123456789012:job/010",
        "targets": [
            "arn:aws:iot:us-east-1:123456789012:thing/myThing"
        ],
        "jobProcessDetails": {
            "numberOfCanceledThings": 0,
            "numberOfFailedThings": 0,
            "numberOfInProgressThings": 0,
            "numberOfQueuedThings": 0,
            "numberOfRejectedThings": 0,
            "numberOfRemovedThings": 0,
            "numberOfSucceededThings": 0,
            "numberOfTimedOutThings": 0,
            "processingTargets": [
                arn:aws:iot:us-east-1:123456789012:thing/thingOne,
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupOne,
                arn:aws:iot:us-east-1:123456789012:thing/thingTwo,
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupTwo
            ]
        },
        "presignedUrlConfig": {
            "expiresInSec": 60,
            "roleArn": "arn:aws:iam::123456789012:role/S3DownloadRole"
        },
        "jobId": "010",
        "lastUpdatedAt": 1486593195.006,
        "createdAt": 1486593195.006,
        "targetSelection": "SNAPSHOT",
        "jobExecutionsRolloutConfig": {
            "exponentialRate": {
                "baseRatePerMinute": integer,
                "incrementFactor": integer,
                "rateIncreaseCriteria": {
                    "numberOfNotifiedThings": integer, // Set one or the other
                    "numberOfSucceededThings": integer // of these two values.
                },
            "maximumPerMinute": integer
         }
        },
        "abortConfig": {
            "criteriaList": [
                {
```

```
                    "action": "string",
                    "failureType": "string",
                    "minNumberOfExecutedThings": integer,
                    "thresholdPercentage": integer
                }
            ]
        },
        "timeoutConfig": {
           "inProgressTimeoutInMinutes": number
         }
    }
}
```

Visualizzazione di un elenco delle esecuzioni per un processo

Un processo in esecuzione in un determinato dispositivo è rappresentato da un oggetto esecuzione del processo. Esegui il comando ListJobExecutionsForJob per elencare tutte le esecuzioni per un processo. Segue una descrizione di come visualizzare l'elenco delle esecuzioni per un processo:

```
aws iot list-job-executions-for-job --job-id 010
```

Il comando restituisce un elenco delle esecuzioni del processo:

```
{
    "executionSummaries": [
    {
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "jobExecutionSummary": {
            "status": "QUEUED",
            "lastUpdatedAt": 1486593196.378,
            "queuedAt": 1486593196.378,
            "executionNumber": 1234567890
        }
    },
    {
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingTwo",
        "jobExecutionSummary": {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486593345.659,
            "queuedAt": 1486593196.378,
            "startedAt": 1486593345.659,
            "executionNumber": 4567890123
```

```
        }
    }
    ]
}
```

Visualizzazione di un elenco delle esecuzioni di un processo per un oggetto

Esegui il comando ListJobExecutionsForThing per elencare tutte le esecuzioni di processo in corso su un oggetto. Segue una descrizione di come visualizzare l'elenco delle esecuzioni di processo per un oggetto:

```
aws iot list-job-executions-for-thing --thing-name thingOne
```

Il comando restituisce un elenco delle esecuzioni del processo in corso o che sono avvenute nell'oggetto specificato:

```
{
    "executionSummaries": [
    {
        "jobExecutionSummary": {
            "status": "QUEUED",
            "lastUpdatedAt": 1486687082.071,
            "queuedAt": 1486687082.071,
            "executionNumber": 9876543210
        },
        "jobId": "013"
    },
    {
        "jobExecutionSummary": {
            "status": "IN_PROGRESS",
            "startAt": 1486685870.729,
            "lastUpdatedAt": 1486685870.729,
            "queuedAt": 1486685870.729,
            "executionNumber": 1357924680
        },
        "jobId": "012"
    },
    {
        "jobExecutionSummary": {
            "status": "SUCCEEDED",
            "startAt": 1486678853.415,
            "lastUpdatedAt": 1486678853.415,
```

```
            "queuedAt": 1486678853.415,
            "executionNumber": 4357680912
        },
        "jobId": "011"
    },
    {
        "jobExecutionSummary": {
            "status": "CANCELED",
            "startAt": 1486593196.378,
            "lastUpdatedAt": 1486593196.378,
            "queuedAt": 1486593196.378,
            "executionNumber": 2143174250
        },
        "jobId": "010"
    }
    ]
}
```

Descrizione dell'esecuzione di un processo

Esegui il comando DescribeJobExecution per ottenere lo stato dell'esecuzione di un processo. Per identificare l'esecuzione del processo, devi specificare un ID processo, un nome di oggetto e, facoltativamente, un numero di esecuzione. La sezione seguente mostra come descrivere l'esecuzione di un processo:

```
aws iot describe-job-execution --job-id 017 --thing-name thingOne
```

Il comando restituisce `JobExecution`. Ad esempio:

```
{
    "execution": {
        "jobId": "017",
        "executionNumber": 4516820379,
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "versionNumber": 123,
        "createdAt": 1489084805.285,
        "lastUpdatedAt": 1489086279.937,
        "startedAt": 1489086279.937,
        "status": "IN_PROGRESS",
        "approximateSecondsBeforeTimedOut": 100,
        "statusDetails": {
            "status": "IN_PROGRESS",
```

```
            "detailsMap": {
                "percentComplete": "10"
            }
        }
    }
}
```

Eliminazione dell'esecuzione di un processo.

Esegui il comando DeleteJobExecution per eliminare l'esecuzione di un processo. Per identificare l'esecuzione del processo, devi specificare un ID processo, un nome di oggetto e un numero di esecuzione. La sezione seguente mostra come eliminare l'esecuzione di un processo:

```
aws iot delete-job-execution --job-id 017 --thing-name thingOne --execution-number
 1234567890 --force|--no-force
```

Il comando non visualizza alcun output.

Per impostazione predefinita, lo stato di esecuzione del processo deve essere `QUEUED` o terminale (`SUCCEEDED`, `FAILED`, `REJECTED`, `TIMED_OUT`, `REMOVED` o `CANCELED`). In caso contrario, si verifica un errore. Per eliminare l'esecuzione di un processo con uno stato `IN_PROGRESS`, è possibile impostare il parametro `force` su `true`.

> **Warning**
>
> Quando elimini l'esecuzione di un processo con stato `IN_PROGRESS`, il dispositivo che sta eseguendo il processo non è in grado di accedere alle informazioni sul processo o di aggiornare lo stato di esecuzione del processo. Prestare attenzione e verificare che il dispositivo possa effettuare il ripristino a uno stato valido.

Modelli di processo

I modelli di processo consentono di preconfigurare i processi in modo da poterli implementare in più set di dispositivi di destinazione. Per implementare le azioni remote eseguite di frequente sui dispositivi, come il riavvio o l'installazione di un'applicazione, è possibile utilizzare dei modelli per definire configurazioni standard. Per eseguire azioni come l'implementazione di patch di sicurezza e correzioni di bug, è inoltre possibile creare modelli da processi esistenti.

Quando crei un modello di processo, puoi specificare le seguenti configurazioni e risorse aggiuntive.

- Proprietà processo
- Documenti e target di processo
- Criteri di rollout, pianificazione e annullamento
- Criteri di timeout e nuovi tentativi

Modelli personalizzati e AWS gestiti

A seconda dell'azione remota che si desidera eseguire, è possibile creare un modello di lavoro personalizzato o utilizzare un modello AWS gestito. Utilizza modelli di lavoro personalizzati per fornire il tuo documento di lavoro personalizzato e crea lavori riutilizzabili da distribuire sui tuoi dispositivi. AWS i modelli gestiti sono modelli di lavoro forniti da AWS IoT Jobs per le azioni più comuni. Questi modelli hanno un documento di processo predefinito per alcune azioni remote, quindi non è necessario creare il proprio documento di processo. I modelli gestiti ti aiutano a creare processi riutilizzabili per un'esecuzione più rapida sui tuoi dispositivi.

Argomenti

- Utilizza modelli AWS gestiti per implementare operazioni remote comuni
- Creazione di un modello di processo personalizzato

Utilizza modelli AWS gestiti per implementare operazioni remote comuni

AWS i modelli gestiti sono modelli di lavoro forniti da AWS. Vengono utilizzati per azioni remote eseguite di frequente, come il riavvio, il download di un file o l'installazione di un'applicazione sui dispositivi. Questi modelli hanno un documento di processo predefinito per ogni azione remota, quindi non è necessario creare il proprio documento di processo.

È possibile scegliere tra un set di configurazioni predefinite e creare processi utilizzando questi modelli senza scrivere alcun codice aggiuntivo. Utilizzando i modelli gestiti, è possibile visualizzare il documento di processo distribuito nei parchi istanze. È possibile creare sia un processo utilizzando questi modelli sia un modello di processo personalizzato da riutilizzare per le azioni remote.

Cosa contengono i modelli gestiti?

Ogni modello AWS gestito contiene:

- L'ambiente in cui eseguire i comandi contenuti nel documento di processo.

- Il documento di processo che specifica il nome dell'azione e i relativi parametri. Ad esempio, se usi un modello Download file, il nome dell'azione è Download file e i parametri possono essere:
  - L'URL del file che desideri scaricare sul dispositivo. Può essere una risorsa Internet oppure un URL pubblico o Amazon Simple Storage Service (Amazon S3) prefirmato.
  - Il percorso del file locale sul dispositivo per memorizzare il file scaricato.

  Per ulteriori informazioni sui parametri e i documenti del processo, consulta Azioni remote dei modelli gestiti e documenti di processo.

Prerequisiti

Affinché i dispositivi eseguano le azioni remote specificate nel documento del processo di modello gestito, è necessario:

- Installare il software specifico sul tuo dispositivo

  Utilizza il software e i gestori dei lavori del tuo dispositivo o il AWS IoT Device Client. A seconda del caso d'uso, puoi anche scegliere di eseguirli entrambi in modo che svolgano funzioni diverse.

  - Utilizzo del software del dispositivo e dei gestori di processo

    Puoi scrivere il tuo codice per i dispositivi utilizzando il SDK per dispositivi AWS IoT e la relativa libreria di gestori che supporta le operazioni remote. Per distribuire ed eseguire processi, verifica che le librerie dell'agente del dispositivo siano state installate correttamente e siano in esecuzione sui dispositivi.

    Puoi anche scegliere di utilizzare i tuoi gestori che supportano le operazioni remote. Per ulteriori informazioni, consulta Sample job handlers nell'archivio Device Client. AWS IoT GitHub
  - Usa il Device Client AWS IoT

    In alternativa, puoi installare ed eseguire AWS IoT Device Client sui tuoi dispositivi perché per impostazione predefinita supporta l'utilizzo di tutti i modelli gestiti direttamente dalla console.

    Device Client è un software open source scritto in C++ che è possibile compilare e installare sui dispositivi IoT integrati basati su Linux. Il Device Client ha un client di base e una discreta funzionalità lato client. Il client di base stabilisce la connettività AWS IoT tramite protocollo MQTT e può connettersi con le diverse funzionalità lato client.

Per eseguire operazioni remote sui dispositivi, utilizzare la funzionalità dei processi lato client del Device Client. Questa funzione contiene un parser per ricevere il documento di processo e i gestori dei processi che implementano le azioni remote specificate nel documento di processo. Per ulteriori informazioni sul Device Client e sulle sue caratteristiche, consulta Device Client di AWS IoT.

Quando è in esecuzione su dispositivi, il Device Client riceve il documento del processo e dispone di un'implementazione specifica della piattaforma, utilizzata per eseguire i comandi nel documento. Per ulteriori informazioni sulla configurazione del Device Client e sull'utilizzo della funzionalità Jobs, consultare Tutorial AWS IoT.

- Utilizzo di un ambiente supportato

Per ogni modello gestito, troverai informazioni sull'ambiente che puoi utilizzare per eseguire le azioni remote. Consigliamo di utilizzare il modello con un ambiente Linux supportato, come specificato nel modello. Usa AWS IoT Device Client per eseguire le azioni remote del modello gestito perché supporta microprocessori e ambienti Linux comuni, come Debian e Ubuntu.

Azioni remote dei modelli gestiti e documenti di processo

La sezione seguente elenca i diversi modelli AWS gestiti per AWS IoT Jobs e descrive le azioni remote che possono essere eseguite sui dispositivi. Nella sezione seguente sono incluse informazioni sul documento di processo e una descrizione dei parametri del documento di processo per ogni azione remota. Il software lato dispositivo utilizza il nome del modello e i parametri per eseguire l'azione remota.

AWS i modelli gestiti accettano parametri di input per i quali si specifica un valore quando si crea un lavoro utilizzando il modello. Tutti i modelli gestiti hanno in comune due parametri di input opzionali: `runAsUser` e `pathToHandler`. Ad eccezione del modello `AWS-Reboot`, i modelli richiedono parametri di input aggiuntivi per i quali è necessario specificare un valore durante la creazione di un processo utilizzando il modello. Questi parametri di input richiesti variano in base al modello scelto. Ad esempio, se scegli il modello `AWS-Download-File`, devi specificare un elenco di pacchetti da installare e un URL da cui scaricare i file.

Specificate un valore per i parametri di input quando utilizzate la AWS IoT console o il AWS Command Line Interface (AWS CLI) per creare un lavoro che utilizza un modello gestito. Quando utilizzi la CLI, fornisci questi valori utilizzando l'oggetto `document-parameters`. Per ulteriori informazioni, consulta documentParameters.

> **Note**
>
> Utilizza i `document-parameters` solo quando crei i processi dai modelli gestiti da AWS . Questo parametro non può essere utilizzato con i modelli di processo personalizzati o per creare processi da tali modelli.

Di seguito viene illustrata una descrizione dei parametri di input opzionali di uso comune. Vedrai una descrizione degli altri parametri di input richiesti da ciascun modello gestito nella sezione successiva.

`runAsUser`

Questo parametro specifica se eseguire il gestore di processi come un altro utente. Se non viene specificato durante la creazione del processo, il gestore del processo viene eseguito come lo stesso utente del Device Client. Quando si esegue il gestore di processi come un altro utente, specificare un valore di stringa non superiore a 256 caratteri.

`pathToHandler`

Il percorso del gestore di processo in esecuzione sul dispositivo. Se non specificato durante la creazione del processo, il Device Client utilizza la directory di processo corrente.

Di seguito vengono illustrate le diverse azioni remote, i relativi documenti di processo e i parametri accettati. Tutti questi modelli supportano l'ambiente Linux per l'esecuzione dell'azione remota sul dispositivo.

AWS–Download–File

Nome del modello

`AWS-Download-File`

Descrizione del modello

Un modello gestito fornito da AWS per scaricare un file.

Parametri di input

Questo modello non prevede parametri obbligatori. Puoi specificare i parametri facoltativi `runAsUser` e `pathToHandler`.

`downloadUrl`

L'URL da cui scaricare il file. Questo può essere una risorsa Internet, un oggetto in Amazon S3 a cui è possibile accedere pubblicamente, o un oggetto in Amazon S3 a cui è possibile accedere solo dal dispositivo utilizzando un URL prefirmato. Per ulteriori informazioni sull'utilizzo di autorizzazioni predefinite URLs e sulla concessione di autorizzazioni, vedere. Predefinito URLs

`filePath`

Percorso del file locale che mostra il percorso in cui memorizzare il file scaricato all'interno del dispositivo.

Comportamento del dispositivo

Il dispositivo scarica il file dal percorso specificato, verifica che il download sia completo e lo memorizza localmente.

Documento del processo

Di seguito viene illustrato il documento di processo e la sua versione più recente. Il modello mostra il percorso del gestore di processo e lo script della shell, `download-file.sh`, che il gestore di processo deve eseguire per scaricare il file. Mostra inoltre i parametri obbligatori `downloadUrl` e `filePath`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Download-File",
        "type": "runHandler",
        "input": {
          "handler": "download-file.sh",
          "args": [
            "${aws:iot:parameter:downloadUrl}",
            "${aws:iot:parameter:filePath}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
```

```
  ]
}
```

AWS–Installa–Applicazione

Nome del modello

`AWS-Install-Application`

Descrizione del modello

Un modello gestito fornito da AWS per l'installazione di una o più applicazioni.

Parametri di input

Questo modello richiede il seguente parametro obbligatorio: `packages`. Puoi specificare i parametri facoltativi `runAsUser` e `pathToHandler`.

`packages`

Un elenco separato da spazi contenente una o più applicazioni da installare.

Comportamento del dispositivo

Il dispositivo installa le applicazioni come specificato nel documento di processo.

Documento del processo

Di seguito viene illustrato il documento di processo e la sua versione più recente. Il modello mostra il percorso del gestore di processo e lo script della shell, `install-packages.sh`, che il gestore di processo deve eseguire per scaricare il file. Mostra inoltre il parametro obbligatorio `packages`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Install-Application",
        "type": "runHandler",
        "input": {
          "handler": "install-packages.sh",
```

```
          "args": [
            "${aws:iot:parameter:packages}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Riavvio

Nome del modello

`AWS-Reboot`

Descrizione del modello

Un modello gestito fornito da AWS per il riavvio del dispositivo.

Parametri di input

Questo modello non prevede parametri obbligatori. Puoi specificare i parametri facoltativi `runAsUser` e `pathToHandler`.

Comportamento del dispositivo

Il dispositivo si riavvia correttamente.

Documento del processo

Di seguito viene illustrato il documento di processo e la sua versione più recente. Il modello mostra il percorso del gestore di processo e lo script della shell, `reboot.sh`, che il gestore del processo deve eseguire per riavviare il dispositivo.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
```

```
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Rimuovi–Applicazione

Nome del modello

`AWS-Remove-Application`

Descrizione del modello

Un modello gestito fornito da AWS per la disinstallazione di una o più applicazioni.

Parametri di input

Questo modello richiede il seguente parametro obbligatorio: `packages`. Puoi specificare i parametri facoltativi `runAsUser` e `pathToHandler`.

`packages`

Un elenco separato da spazi contenente una o più applicazioni da disinstallare.

Comportamento del dispositivo

Il dispositivo disinstalla le applicazioni come specificato nel documento di processo.

Documento del processo

Di seguito viene illustrato il documento di processo e la sua versione più recente. Il modello mostra il percorso del gestore di processo e lo script della shell, `remove-packages.sh`, che il gestore di processo deve eseguire per scaricare il file. Mostra inoltre il parametro obbligatorio `packages`.

```
{
  "version": "1.0",
```

```
  "steps": [
    {
      "action": {
        "name": "Remove-Application",
        "type": "runHandler",
        "input": {
          "handler": "remove-packages.sh",
          "args": [
            "${aws:iot:parameter:packages}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Riavvia–Applicazione

Nome del modello

`AWS-Restart-Application`

Descrizione del modello

Un modello gestito fornito da AWS per l'arresto e il riavvio di uno o più servizi.

Parametri di input

Questo modello richiede il seguente parametro obbligatorio: `services`. Puoi specificare i parametri facoltativi `runAsUser` e `pathToHandler`.

Servizi

Un elenco separato da spazi contenente una o più applicazioni da riavviare.

Comportamento del dispositivo

Le applicazioni specificate vengono arrestate e quindi riavviate sul dispositivo.

Documento del processo

Di seguito viene illustrato il documento di processo e la sua versione più recente. Il modello mostra il percorso del gestore di processo e lo script della shell, `restart-services.sh`, che il gestore di processi deve eseguire per riavviare i servizi di sistema. Mostra inoltre il parametro obbligatorio `services`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Restart-Application",
        "type": "runHandler",
        "input": {
          "handler": "restart-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Avvio–Applicazione

Nome del modello

`AWS-Start-Application`

Descrizione del modello

Un modello gestito fornito da AWS per l'avvio di uno o più servizi.

Parametri di input

Questo modello richiede il seguente parametro obbligatorio: `services`. Puoi specificare i parametri facoltativi `runAsUser` e `pathToHandler`.

`services`

Un elenco separato da spazi contenente una o più applicazioni da avviare.

Comportamento del dispositivo

Le applicazioni specificate iniziano a funzionare sul dispositivo.

Documento del processo

Di seguito viene illustrato il documento di processo e la sua versione più recente. Il modello mostra il percorso del gestore di processo e lo script della shell, `start-services.sh`, che il gestore di processo deve eseguire per avviare i servizi di sistema. Mostra inoltre il parametro obbligatorio `services`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Start-Application",
        "type": "runHandler",
        "input": {
          "handler": "start-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Interruzione–Applicazione

Nome del modello

`AWS-Stop-Application`

Descrizione del modello

Un modello gestito fornito da AWS per interrompere uno o più servizi.

Parametri di input

Questo modello richiede il seguente parametro obbligatorio: `services`. Puoi specificare i parametri facoltativi `runAsUser` e `pathToHandler`.

`services`

Un elenco separato da spazi contenente una o più applicazioni da arrestare.

Comportamento del dispositivo

Le applicazioni specificate smettono di funzionare sul dispositivo.

Documento del processo

Di seguito viene illustrato il documento di processo e la sua versione più recente. Il modello mostra il percorso del gestore di processo e lo script della shell, `stop-services.sh`, che il gestore di processo deve eseguire per arrestare i servizi di sistema. Mostra inoltre il parametro obbligatorio `services`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Stop-Application",
        "type": "runHandler",
        "input": {
          "handler": "stop-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Esegui–Comando

Nome del modello

`AWS-Run-Command`

Descrizione del modello

Un modello gestito fornito da AWS per l'esecuzione di un comando shell.

Parametri di input

Questo modello richiede il seguente parametro obbligatorio: `command`. Puoi specificare il parametro facoltativo `runAsUser`.

`command`

Una stringa di comando separata da virgole. Qualsiasi virgola contenuta nel comando stesso deve essere preceduta da un carattere di escape.

Comportamento del dispositivo

Il dispositivo esegue il comando shell come specificato nel documento del processo.

Documento di processo

Di seguito viene illustrato il documento di processo e la sua versione più recente. Il modello mostra il percorso del comando del processo e il comando che hai fornito per essere eseguito dal dispositivo.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Run-Command",
        "type": "runCommand",
        "input": {
          "command": "${aws:iot:parameter:command}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Argomenti

- Crea un lavoro da modelli AWS gestiti utilizzando Console di gestione AWS

- Crea un lavoro da modelli AWS gestiti utilizzando il AWS CLI

Crea un lavoro da modelli AWS gestiti utilizzando Console di gestione AWS

Utilizza il Console di gestione AWS per ottenere informazioni sui modelli AWS gestiti e creare un lavoro utilizzando questi modelli. Puoi quindi salvare il processo creato come modello personalizzato.

Ottieni dettagli sui modelli gestiti

È possibile ottenere informazioni sui diversi modelli gestiti disponibili per l'uso dalla AWS IoT console.

1. Per visualizzare i modelli gestiti disponibili, vai all'hub Job templates della AWS IoT console e scegli la scheda Modelli gestiti.
2. Per visualizzarne i dettagli, scegli un modello gestito.

Nella pagina Dettagli immetti le seguenti informazioni:

- Nome, descrizione e Amazon Resource Name (ARN) del modello gestito.
- L'ambiente in cui è possibile eseguire le azioni remote, come ad esempio Linux.
- Il documento del processo JSON che specifica il percorso del gestore del processo e i comandi da eseguire sul dispositivo. Di seguito viene illustrato un esempio di documento del processo per il modello AWS-Reboot. Il modello mostra il percorso del gestore di processo e lo script della shell, `reboot.sh`, che il gestore del processo deve eseguire per riavviare il dispositivo.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
```

```
}
```

Per ulteriori informazioni sul documento del processo e sui relativi parametri per varie azioni remote, consulta Azioni remote dei modelli gestiti e documenti di processo.

- L'ultima versione del documento del processo.

Creazione di un processo utilizzando modelli gestiti

Puoi utilizzare la console di AWS gestione per scegliere un modello AWS gestito da utilizzare per creare un lavoro. in questa sezione viene illustrato come procedere.

È inoltre possibile avviare il flusso di lavoro per la creazione del lavoro e quindi scegliere il modello AWS gestito che si desidera utilizzare durante la creazione del lavoro. Per ulteriori informazioni su questo flusso di lavoro, consulta Crea e gestisci lavori utilizzando Console di gestione AWS.

1. Scegli il tuo modello AWS gestito

   Vai all'hub Job templates della AWS IoT console, scegli la scheda Modelli gestiti, quindi scegli il tuo modello.

2. Crea un processo utilizzando il tuo modello gestito

   1. Nella pagina dei dettagli del tuo modello, scegli Create job (Crea processo).

      La console passa alla fase Custom job properties (Proprietà del processo personalizzato) del flusso di lavoro Create Job (Crea processo), in cui è stata aggiunta la configurazione del modello.

   2. Inserire un nome di processo alfanumerico univoco, una descrizione e i tag facoltativi, quindi scegliere Next (Successivo).

   3. Scegli gli oggetti o i gruppi di oggetti che desideri eseguire come target nel processo.

   4. Nella sezione Job document (Documento di processo), il modello viene visualizzato con le relative impostazioni di configurazione e i parametri di input. Inserisci i valori per i parametri di input del modello scelto. Ad esempio, se hai scelto il modello AWS-Download-File:

      - Per downloadUrl inserisci l'URL del file da scaricare, ad esempio: `https://example.com/index.html`.
      - Per filePath inserisci il percorso del dispositivo per memorizzare il file scaricato, ad esempio: *path/to/file*.

Facoltativamente, puoi anche possibile inserire i valori per i parametri `runAsUser` e `pathToHandler`. Per ulteriori informazioni sui parametri di input di ogni modello, consulta Azioni remote dei modelli gestiti e documenti di processo.

5. Nella pagina Job Configuration (Configurazione processo), scegli il tipo di processo come continuo o un processo snapshot. Un processo di snapshot viene completato quando termina la sua esecuzione su dispositivi e gruppi di destinazione. Un processo continuo si applica ai gruppi di oggetti e viene eseguito su qualsiasi dispositivo aggiunto a un gruppo di destinazione specificato.
6. Continua ad aggiungere eventuali configurazioni aggiuntive per il tuo processo, quindi esamina e crea il tuo processo. Per ulteriori informazioni sulle configurazioni aggiuntive, consulta:
   - Configurazioni di rollout, pianificazione e interruzione dei processi
   - Configurazioni di timeout e di nuovo tentativo di esecuzione del processo

Creazione di modelli di processo personalizzati partendo da modelli gestiti

Puoi utilizzare un modello AWS gestito e un lavoro personalizzato come punto di partenza per creare il tuo modello di lavoro personalizzato. Per creare un modello di lavoro personalizzato, crea prima un lavoro dal tuo modello AWS gestito come descritto nella sezione precedente.

È quindi possibile salvare il processo personalizzato come modello, da cui creare un modello di processo personalizzato. Per salvare come modello:

1. Vai al Job hub della AWS IoT console e scegli il lavoro contenente il tuo modello gestito.
2. Scegli Save as job template (Salva come modello di processo) per creare il tuo modello di processo personalizzato. Per ulteriori informazioni sulla creazione di un modello di processo personalizzato, consulta Creazione di un modello del processo da un processo esistente.

Crea un lavoro da modelli AWS gestiti utilizzando il AWS CLI

Utilizza il AWS CLI per ottenere informazioni sui modelli AWS gestiti e creare un lavoro utilizzando questi modelli. Puoi quindi salvare il processo come modello e creare un modello personalizzato.

Elenca i modelli gestiti

Il `list-managed-job-templates` AWS CLI comando elenca tutti i modelli di lavoro presenti nel tuo Account AWS.

```
aws iot list-managed-job-templates
```

Per impostazione predefinita, l'esecuzione di questo comando visualizza tutti i modelli AWS gestiti disponibili e i relativi dettagli.

```
{
    "managedJobTemplates": [
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Reboot:1.0",
            "templateName": "AWS-Reboot",
            "description": "A managed job template for rebooting the device.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Remove-
Application:1.0",
            "templateName": "AWS-Remove-Application",
            "description": "A managed job template for uninstalling one or more
 applications.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Stop-Application:1.0",
            "templateName": "AWS-Stop-Application",
            "description": "A managed job template for stopping one or more system
 services.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },

        ...

        {
```

```
            "templateArn": "arn:aws:iot:us-east-1::jobtemplate/AWS-Restart-
Application:1.0",
            "templateName": "AWS-Restart-Application",
            "description": "A managed job template for restarting one or more system
 services.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        }
    ]
}
```

Per ulteriori informazioni, consulta ListManagedJobTemplates.

Recupero dei dettagli di un modello di processo

Il `describe-managed-job-template` AWS CLI comando ottiene dettagli su un modello di processo specificato. Specificare il nome del modello di processo e una versione opzionale del modello. Se la versione del modello non è specificata, viene restituita la versione predefinita. Di seguito viene illustrato un esempio di esecuzione del comando per recuperare i dettagli del modello `AWS-Download-File`.

```
aws iot describe-managed-job-template \
    --template-name AWS-Download-File
```

Il comando visualizza i dettagli del modello e l'ARN, il relativo documento di processo e il parametro `documentParameters`, che è un elenco di coppie di valori chiave dei parametri di input del modello. Per ulteriori informazioni sui diversi modelli e sui parametri di input, consulta Azioni remote dei modelli gestiti e documenti di processo.

> **Note**
>
> L'`documentParameters`oggetto restituito quando si utilizza questa API deve essere utilizzato solo durante la creazione di lavori da modelli AWS gestiti. L'oggetto non deve essere utilizzato per modelli di processo personalizzati. Per un esempio che mostra come utilizzare questo parametro, consulta Creazione di un processo utilizzando modelli gestiti.

```
{
    "templateName": "AWS-Download-File",
    "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Download-File:1.0",
    "description": "A managed job template for downloading a file.",
    "templateVersion": "1.0",
    "environments": [
        "LINUX"
    ],
    "documentParameters": [
        {
            "key": "downloadUrl",
            "description": "URL of file to download.",
            "regex": "(.*?)",
            "example": "http://www.example.com/index.html",
            "optional": false
        },
        {
            "key": "filePath",
            "description": "Path on the device where downloaded file is written.",
            "regex": "(.*?)",
            "example": "/path/to/file",
            "optional": false
        },
        {
            "key": "runAsUser",
            "description": "Execute handler as another user. If not specified, then
 handler is executed as the same user as device client.",
            "regex": "(.){0,256}",
            "example": "user1",
            "optional": true
        },
        {
            "key": "pathToHandler",
            "description": "Path to handler on the device. If not specified, then
 device client will use the current working directory.",
            "regex": "(.){0,4096}",
            "example": "/path/to/handler/script",
            "optional": true
        }
    ],
    "document": "{\"version\":\"1.0\",\"steps\":[{\"action\":{\"name
\":\"Download-File\",\"type\":\"runHandler\",\"input\":{\"handler\":
\"download-file.sh\",\"args\":[\"${aws:iot:parameter:downloadUrl}\",
```

```
\"${aws:iot:parameter:filePath}\"],\"path\":\"${aws:iot:parameter:pathToHandler}\"},
\"runAsUser\":\"${aws:iot:parameter:runAsUser}\"}}]}"
}
```

Per ulteriori informazioni, consulta DescribeManagedJobTemplate.

Creazione di un processo utilizzando modelli gestiti

Il `create-job` AWS CLI comando può essere utilizzato per creare un lavoro da un modello di lavoro. Si rivolge a un dispositivo denominato `thingOne` e specifica l'Amazon Resource Name (ARN) del modello gestito da utilizzare come base per il processo. È possibile sovrascrivere le configurazioni avanzate, ad esempio le configurazioni di timeout e annullamento, passando i parametri associati del comando `create-job`.

L'esempio mostra come creare un processo che utilizza il modello `AWS-Download-File`. Viene inoltre illustrato come specificare i parametri di input del modello utilizzando il parametro `document-parameters`.

> **Note**
>
> Utilizzate l'`document-parameters`oggetto solo con i modelli AWS gestiti. Non deve essere utilizzato con modelli di processo personalizzati.

```
aws iot create-job \
   --targets arn:aws:iot:region:account-id:thing/thingOne \
   --job-id "new-managed-template-job" \
   --job-template-arn arn:aws:iot:region::jobtemplate/AWS-Download-File:1.0 \
   --document-parameters downloadUrl=https://example.com/index.html,filePath=path/to/
file
```

dove:

- *`region`*è il Regione AWS.
- *`account-id`*è il Account AWS numero univoco.
- *`thingOne`* è il nome dell'oggetto IoT di destinazione del processo.
- *`AWS-Download-File:1.0`*è il nome del modello gestito.
- `https://`*`example.com/index`*`.html` è l'URL da cui scaricare il file.

- `https://pathto/file/index` è il percorso del dispositivo per memorizzare il file scaricato.

Esegui il comando seguente per creare un processo per il modello, *AWS-Download-File*.

```
{
    "jobArn": "arn:aws:iot:region:account-id:job/new-managed-template-job",
    "jobId": "new-managed-template-job",
    "description": "A managed job template for downloading a file."
}
```

Creazione di modelli di processo personalizzati partendo da modelli gestiti

1. Crea un processo utilizzando un modello gestito come descritto nella sezione precedente.
2. Crea un modello di processo personalizzato utilizzando l'ARN del processo creato. Per ulteriori informazioni, consulta Creazione di un modello del processo da un processo esistente.

Creazione di un modello di processo personalizzato

È possibile creare modelli di lavoro utilizzando AWS CLI e la AWS IoT console. È inoltre possibile creare lavori da modelli di lavoro utilizzando le applicazioni Web Fleet Hub for Device Management AWS CLI, la AWS IoT console e Fleet Hub for AWS IoT Device Management. Per ulteriori informazioni sull'utilizzo dei modelli di lavoro nelle applicazioni Fleet Hub, consulta Utilizzo dei modelli di lavoro in Fleet Hub for AWS IoT Device Management.

> **Note**
>
> Il numero totale di modelli di sostituzione in un documento di lavoro non deve superare i dieci.

Argomenti

- Crea modelli di lavoro personalizzati utilizzando Console di gestione AWS
- Crea modelli di lavoro personalizzati utilizzando AWS CLI

Crea modelli di lavoro personalizzati utilizzando Console di gestione AWS

Questo argomento spiega come creare, eliminare e visualizzare i dettagli sui modelli di lavoro utilizzando la AWS IoT console.

Crea di un modello di processo personalizzato

Puoi creare un modello di processo personalizzato originale o un modello di processo partendo da un processo esistente. È inoltre possibile creare un modello di lavoro personalizzato da un lavoro esistente creato utilizzando un modello AWS gestito. Per ulteriori informazioni, consulta Creazione di modelli di processo personalizzati partendo da modelli gestiti.

Creazione di un modello di processo originale

1. Inizia a creare il tuo modello di processo

   1. Vai all'hub Job templates della AWS IoT console e scegli la scheda Modelli personalizzati.
   2. Scegliere Crea un modello.

   > **Note**
   >
   > È anche possibile passare alla pagina Job templates (Modelli di processo) dalla pagina Related services (Servizi correlati) sotto Fleet Hub.

2. Specifica le proprietà del modello di processo

   Nella pagina Create job template (Crea modello di processo), inserisci un identificatore alfanumerico per il nome del processo e una descrizione alfanumerica per fornire ulteriori dettagli sul modello.

   > **Note**
   >
   > Ti sconsigliamo di utilizzare informazioni di identificazione personale nelle tue offerte di lavoro IDs o nelle descrizioni.

3. Fornisci un documento del processo

   Fornisci un file di processo JSON memorizzato in un bucket S3, o come documento di processo in linea specificato all'interno del processo. Questo file di processo diventerà il documento di processo quando creerai un processo utilizzando questo modello.

   Se il file del processo è archiviato in un bucket S3, inserisci l'URL S3 o scegli Browse S3 (Cerca in S3), quindi vai al documento di processo e selezionalo.

> **Note**
>
> È possibile selezionare solo bucket S3 appartenenti alla tua regione corrente.

4. Continua ad aggiungere eventuali configurazioni aggiuntive per il tuo processo, quindi esamina e crea il tuo processo. Per informazioni sulle configurazioni aggiuntive e opzionali, fai riferimento ai collegamenti seguenti:

   - Configurazioni di rollout, pianificazione e interruzione dei processi
   - Configurazioni di timeout e di nuovo tentativo di esecuzione del processo

Creazione di un modello del processo da un processo esistente

1. Seleziona il processo

   1. Vai al Job hub della AWS IoT console e scegli il lavoro che desideri utilizzare come base per il tuo modello di lavoro.
   2. Scegli Save as job template (Salva come modello di processo).

   > **Note**
   >
   > Facoltativo - è possibile selezionare un documento di processo diverso, o modificare le configurazioni avanzate dal processo originale, quindi scegliere Create job templates (Crea modello di processo). Il nuovo modello di processo verrà visualizzato nella pagina Job templates (Modelli di processo).

2. Specifica le proprietà del modello di processo

   Nella pagina Create job template (Crea modello di processo), inserisci un identificatore alfanumerico per il nome del processo e una descrizione alfanumerica per fornire ulteriori dettagli sul modello.

   > **Note**
   >
   > Il documento di processo è il file di processo specificato durante la creazione del modello. Se il documento di processo è specificato all'interno del processo anziché

in una percorso S3, è possibile visualizzare il documento nella pagina dei dettagli del processo.

3. Continua ad aggiungere eventuali configurazioni aggiuntive per il tuo processo, quindi esamina e crea il tuo processo. Per ulteriori informazioni sulle configurazioni aggiuntive, consulta:

   - Configurazioni di rollout, pianificazione e interruzione dei processi
   - Configurazioni di timeout e di nuovo tentativo di esecuzione del processo

Creazione di un processo da un modello di processo

Come descritto in questo argomento, è possibile creare un processo da un modello di processo personalizzato, andando alla pagina dei dettagli del modello di processo. È inoltre possibile creare un processo, o scegliere il modello di processo che si desidera utilizzare durante l'esecuzione del flusso di lavoro per la creazione del processo. Per ulteriori informazioni, consulta Crea e gestisci lavori utilizzando Console di gestione AWS.

Questo argomento mostra come creare un processo partendo dalla pagina dei dettagli di un modello di processo personalizzato. Puoi anche creare un lavoro da un modello AWS gestito. Per ulteriori informazioni, consulta Creazione di un processo utilizzando modelli gestiti.

1. Seleziona il modello di processo personalizzato

   Vai all'hub Job templates della AWS IoT console e scegli la scheda Modelli personalizzati, quindi scegli il tuo modello.

2. Crea un processo utilizzando il tuo modello personalizzato

   Per creare un processo:

   1. Nella pagina dei dettagli del tuo modello, scegli Create job (Crea processo).

      La console passa alla fase Custom job properties (Proprietà del processo personalizzato) del flusso di lavoro Create Job (Crea processo), in cui è stata aggiunta la configurazione del modello.

   2. Inserire un nome di processo alfanumerico univoco, una descrizione e i tag facoltativi, quindi scegliere Next (Successivo).
   3. Scegli gli oggetti o i gruppi di oggetti che desideri eseguire come target nel processo.

Nella sezione Job document (Documento di processo), il modello viene visualizzato con le relative impostazioni di configurazione. Per utilizzare un documento di processo diverso, seleziona Browse (Sfoglia) e seleziona un bucket e un documento diversi. Scegli Next (Successivo).

4. Nella pagina Job Configuration (Configurazione processo), scegli il tipo di processo come continuo o un processo snapshot. Un processo di snapshot viene completato quando termina la sua esecuzione su dispositivi e gruppi di destinazione. Un processo continuo si applica ai gruppi di oggetti e viene eseguito su qualsiasi dispositivo aggiunto a un gruppo di destinazione specificato.
5. Continua ad aggiungere eventuali configurazioni aggiuntive per il tuo processo, quindi esamina e crea il tuo processo. Per ulteriori informazioni sulle configurazioni aggiuntive, consulta:
   - Configurazioni di rollout, pianificazione e interruzione dei processi
   - Configurazioni di timeout e di nuovo tentativo di esecuzione del processo

> **Note**
>
> Quando un processo creato da un modello di processo aggiorna i parametri esistenti forniti dal modello di processo, tali parametri aggiornati sostituiranno i parametri esistenti forniti dal modello di processo per tale processo.

Puoi anche creare processi da modelli di processi con le applicazioni Web Fleet Hub. Per informazioni sulla creazione di lavori in Fleet Hub, consulta Lavorare con i modelli di lavoro in Fleet Hub for AWS IoT Device Management.

Eliminazione di un modello di processo

Per eliminare un modello di lavoro, vai prima all'hub Job templates della AWS IoT console e scegli la scheda Modelli personalizzati. Scegli quindi il modello di processo da eliminare e seleziona Delete (Elimina).

> **Note**
>
> L'eliminazione è permanente e il modello di processo non viene più visualizzato nella scheda Custom Templates (Modelli personalizzati).

Crea modelli di lavoro personalizzati utilizzando AWS CLI

Questo argomento spiega come creare, eliminare e recuperare i dettagli sui modelli di processo utilizzando AWS CLI.

Creazione di un modello di processo da zero

Il AWS CLI comando seguente mostra come creare un lavoro utilizzando un documento di lavoro (*job-document.json*) archiviato in un bucket S3 e un ruolo con autorizzazione a scaricare file da Amazon *S3DownloadRole* S3 ().

```
aws iot create-job-template  \
      --job-template-id 010  \
      --description "My custom job template for updating the device firmware"
      --document-source https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json
 \
      --timeout-config inProgressTimeoutInMinutes=100 \
      --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\":
 50, \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\":
 1000, \"numberOfSucceededThings\": 1000}}, \"maximumPerMinute\": 1000}" \
      --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
      --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

Il parametro opzionale `timeout-config` specifica l'intervallo di tempo a disposizione di ciascun dispositivo per terminare l'esecuzione del processo. Il timer viene avviato quando imposti lo stato di esecuzione del processo su `IN_PROGRESS`. Se lo stato di esecuzione del processo non è impostato su un altro stato terminale prima della scadenza del tempo a disposizione, viene impostato su `TIMED_OUT`.

Il timer in corso non può essere aggiornato e viene applicato a tutti i lanci del processo. Ogni volta che l'avvio di un lavoro rimane nello `IN_PROGRESS` stato per un periodo superiore a questo intervallo, l'avvio del lavoro fallisce e passa allo stato del terminale. `TIMED_OUT` AWS IoT pubblica anche una notifica MQTT.

Per ulteriori informazioni sulla creazione delle configurazioni dei rollout e delle interruzioni di processo, consulta la sezione relativa a Job rollout and abort configuration (Configurazione dei rollout e delle interruzioni di processo).

> **Note**
>
> I documenti dei processi specificati come file Amazon S3 vengono recuperati al momento della creazione del processo. Se, dopo la creazione del processo, modifichi i contenuti del file Amazon S3 usato come sorgente del documento del processo, ciò che viene inviato ai target del processo non subisce variazioni.

Creazione di un modello del processo da un processo esistente

Il AWS CLI comando seguente crea un modello di lavoro specificando l'Amazon Resource Name (ARN) di un lavoro esistente. Il nuovo modello di processo utilizza tutte le configurazioni specificate nel processo. Facoltativamente, è possibile modificare qualsiasi configurazione del processo esistente utilizzando uno qualsiasi dei parametri opzionali.

```
aws iot create-job-template  \
      --job-arn arn:aws:iot:region:123456789012:job/job-name  \
      --timeout-config inProgressTimeoutInMinutes=100
```

Ottieni dettagli su un modello di processo

Il AWS CLI comando seguente ottiene dettagli su un modello di lavoro specificato.

```
aws iot describe-job-template \
      --job-template-id template-id
```

Il comando visualizza il seguente output.

```
{
   "abortConfig": {
      "criteriaList": [
         {
```

```
            "action": "string",
            "failureType": "string",
            "minNumberOfExecutedThings": number,
            "thresholdPercentage": number
         }
      ]
   },
   "createdAt": number,
   "description": "string",
   "document": "string",
   "documentSource": "string",
   "jobExecutionsRolloutConfig": {
      "exponentialRate": {
         "baseRatePerMinute": number,
         "incrementFactor": number,
         "rateIncreaseCriteria": {
            "numberOfNotifiedThings": number,
            "numberOfSucceededThings": number
         }
      },
      "maximumPerMinute": number
   },
   "jobTemplateArn": "string",
   "jobTemplateId": "string",
   "presignedUrlConfig": {
      "expiresInSec": number,
      "roleArn": "string"
   },
   "timeoutConfig": {
      "inProgressTimeoutInMinutes": number
   }
}
```

Elenco dei modelli di processo

Il AWS CLI comando seguente elenca tutti i modelli di lavoro presenti nel tuo Account AWS.

```
aws iot list-job-templates
```

Il comando visualizza il seguente output.

```
{
   "jobTemplates": [
      {
         "createdAt": number,
         "description": "string",
         "jobTemplateArn": "string",
         "jobTemplateId": "string"
      }
   ],
   "nextToken": "string"
}
```

Utilizzare il valore del campo `nextToken` per recuperare ulteriori pagine di risultati.

Eliminazione di un modello di processo

Il AWS CLI comando seguente elimina un modello di lavoro specificato.

```
aws iot delete-job-template \
      --job-template-id template-id
```

Il comando non visualizza alcun output.

Creazione di un processo da un modello di processo

Il AWS CLI comando seguente crea un lavoro da un modello di processo personalizzato. Si rivolge a un dispositivo denominato `thing0ne` e specifica l'Amazon Resource Name (ARN) del modello di processo da utilizzare come base per il processo. È possibile sovrascrivere le configurazioni avanzate, ad esempio le configurazioni di timeout e annullamento, passando i parametri associati del comando `create-job`.

> **Warning**
>
> L'oggetto `document-parameters` deve essere utilizzato con il comando `create-job` solo quando si creano processi da modelli gestiti da AWS . Non deve essere utilizzato con modelli di processo personalizzati. Per un esempio che mostra come creare processi con questo parametro, consulta Creazione di un processo utilizzando modelli gestiti.

```
aws iot create-job \
      --targets arn:aws:iot:region:123456789012:thing/thingOne  \
      --job-template-arn arn:aws:iot:region:123456789012:jobtemplate/template-id
```

Configurazioni diprocesso

È possibile disporre delle seguenti configurazioni aggiuntive per ogni processo implementato nelle destinazioni specificate.

- Rollout: definisce il numero di dispositivi che ricevono il documento di lavoro ogni minuto.
- Scheduling (pianificazione): pianifica un processo per una data e un'ora future, in aggiunta all'utilizzo di finestre di manutenzione ricorrente.
- Abort (interruzione): annulla un processo se alcuni dispositivi non ricevono la notifica del processo o se i dispositivi segnalano degli errori relativamente all'esecuzione dei processi.
- Timeout: se gli obiettivi di processo non ricevono una risposta entro un dato termine dall'avvio dell'esecuzione del processo, quest'ultimo potrebbe fallire.
- Retry (nuovo tentativo): per ritentare l'esecuzione del processo se il dispositivo segnala un errore durante il tentativo di completare l'esecuzione di un processo o se quest'ultima scade.

Utilizzando queste configurazioni, puoi monitorare lo stato dell'esecuzione del processo ed evitare che venga inviato un aggiornamento errato a un intero parco istanze.

Argomenti

- Come funzionano le configurazioni di processo
- Specificare configurazioni aggiuntive

Come funzionano le configurazioni di processo

Le configurazioni di rollout e di interruzione si utilizzano durante l'implementazione di un processo, mentre le configurazioni di timeout e nuovo tentativo si utilizzano per l'esecuzione di un processo. Nelle sezioni seguenti sono visualizzate ulteriori informazioni sul funzionamento di queste configurazioni.

Argomenti

- Configurazioni di rollout, pianificazione e interruzione dei processi
- Configurazioni di timeout e di nuovo tentativo di esecuzione del processo

Configurazioni di rollout, pianificazione e interruzione dei processi

È possibile utilizzare le configurazioni di rollout, interruzione e pianificazione dei processi per definire quanti dispositivi ricevono il documento del processo, pianificare una configurazione dei rollout del processo e determinare i criteri di annullamento di un processo.

Configurazione del rollout del processo

Puoi specificare con che velocità vengono inviate ai target le notifiche relative all'esecuzione di un processo in sospeso. È inoltre possibile creare un rollout per fasi per gestire aggiornamenti, riavvii e altre operazioni. Per specificare le modalità di invio delle notifiche agli obiettivi, usa le velocità di rollout del processo.

Velocità di rollout del processo

Puoi creare una configurazione di rollout utilizzando una velocità di rollout costante o esponenziale. Per specificare il numero massimo di obiettivi di processo a cui inviare notifiche ogni minuto, utilizza una velocità di rollout costante.

AWS IoT i lavori possono essere implementati utilizzando tassi di implementazione esponenziali man mano che vengono soddisfatti vari criteri e soglie. Se il numero di processi non riusciti corrisponde a un insieme di criteri che specifichi, il rollout di processo viene annullato. Puoi impostare i criteri della velocità di rollout del processo quando crei un processo utilizzando l'oggetto `JobExecutionsRolloutConfig`. Anche i criteri di interruzione del processo vanno impostati al momento della creazione di un processo mediante l'oggetto `AbortConfig`.

L'esempio seguente mostra come funziona la velocità di rollout. Ad esempio, un processo di rollout con una frequenza di base di 50 al minuto, un fattore di incremento di 2 e un numero di dispositivi notificati ed esecuzioni riuscite pari a 1.000, funziona come segue: il processo inizia con una frequenza di 50 esecuzioni del processo al minuto e prosegue a tale frequenza fino a quando 1.000 oggetti non hanno ricevuto notifiche di esecuzione del processo o non si sono verificate 1.000 esecuzioni del processo.

La tabella riportata di seguito mostra il modo in cui il rollout dovrebbe procedere oltre i primi quattro incrementi.

Velocità di rollout al minuto	50	100	200	400
Numero di dispositivi notificati o di esecuzioni del processo riuscite per soddisfare l'incremento della velocità	1.000	2.000	3.000	4.000

> **Note**
>
> Se è stato raggiunto il limite massimo di 500 processi simultanei (`isConcurrent = True`), tutti i processi attivi rimarranno in uno stato `IN-PROGRESS` e non verranno implementate nuove esecuzioni del processo finché il numero di processi simultanei non sarà pari o inferiore a 499 (`isConcurrent = False`)). Questo si applica a processi continui e a processi snapshot.
> Se `isConcurrent = True`, il processo sta attualmente implementando esecuzioni del processo in tutti i dispositivi del gruppo di destinazione. Se `isConcurrent = False`, il processo ha completato il rollout di tutte le esecuzioni del processo in tutti i dispositivi del gruppo di destinazione. Aggiornerà il suo stato quando tutti i dispositivi nel gruppo di destinazione raggiungono uno stato terminale o una percentuale di soglia del gruppo di destinazione, se è stata selezionata una configurazione di interruzione del processo. Gli stati di livello del processo per `isConcurrent = True` e `isConcurrent = False` sono entrambi `IN_PROGRESS`.
> Per ulteriori informazioni sui limiti di processi attivi e simultanei, consulta Limiti dei processi attivi e simultanei.

Velocità di rollout per processi continui che utilizzano gruppi di oggetti dinamici

Quando utilizzi un processo continuo per implementare operazioni remote sulla tua flotta, AWS IoT Jobs implementa le esecuzioni di lavoro per i dispositivi del gruppo di oggetti target. Per i nuovi dispositivi aggiunti al gruppo di oggetti dinamico, queste esecuzioni dei processi continuano a essere implementate per questi dispositivi anche dopo la creazione del processo.

La configurazione di rollout può controllare la velocità di rollout solo per i dispositivi aggiunti al gruppo fino alla creazione del processo. Una volta creato il processo, per qualsiasi nuovo dispositivo, le esecuzioni del processo vengono create pressoché in tempo reale non appena i dispositivi vengono aggiunti al gruppo di destinazione.

Configurazione della pianificazione del processo

È possibile pianificare un processo continuo o snapshot fino a un anno in anticipo utilizzando un'ora di inizio, un'ora di fine predeterminati e un comportamento di fine per ciò che accadrà a ogni esecuzione del processo quando si raggiunge l'ora di fine. Inoltre, è possibile creare una finestra di manutenzione ricorrente opzionale con una frequenza, un'ora di inizio e una durata flessibili per processi continui per distribuire un documento del processo in tutti i dispositivi all'interno del gruppo di destinazione.

Configurazioni di pianificazione del processo

Ora di inizio

L'ora di inizio di un processo pianificato è la data e l'ora future di inizio del rollout del documento del processo in tutti i dispositivi del gruppo di destinazione. L'ora di inizio di un processo pianificato si applica ai processi continui e ai processi snapshot. Quando un processo pianificato viene creato, all'inizio mantiene lo stato di `SCHEDULED`. Non appena arriva la `startTime` selezionata, lo stato viene aggiornato in `IN_PROGRESS` e viene avviato il rollout del documento del processo. La `startTime` deve essere inferiore o uguale ad un anno dalla data e ora iniziali in cui hai creato il processo pianificato.

Per ulteriori informazioni sulla sintassi da utilizzare con un comando API o il AWS CLI, consulta Timestamp. `startTime`

Per un processo in cui la configurazione della pianificazione opzionale viene eseguita durante una finestra di manutenzione ricorrente in una località in cui vige l'ora legale (DST), la durata cambierà di un'ora quando si passa dall'ora legale all'ora solare e dall'ora solare all'ora legale.

> **Note**
>
> Il fuso orario visualizzato in Console di gestione AWS è il fuso orario corrente del sistema. Tuttavia, questi fusi orari verranno convertiti in UTC nel sistema.

Ora di fine

L'ora di fine di un processo pianificato è la data e l'ora future in cui il processo interromperà il rollout del documento del processo nei dispositivi rimanenti del gruppo di destinazione. L'ora di fine di un processo pianificato si applica ai processi continui e ai processi snapshot. Una volta che un processo arriva alla `endTime` selezionata e tutte le esecuzioni del processo hanno raggiunto uno

stato terminale, il suo stato viene aggiornato da `IN_PROGRESS` a `COMPLETED`. La `endTime` deve essere inferiore o uguale a due anni dalla data e ora iniziali in cui hai creato il processo pianificato. La durata minima compresa tra `startTime` e `endTime` è di 30 minuti. I tentativi di ripetizione dell'esecuzione del processo verranno eseguiti finché il processo non raggiunge la `endTime`, dopo di che `endBehavior` indicherà come procedere.

Per ulteriori informazioni sulla sintassi da utilizzare con un comando API o il AWS CLI, consulta Timestamp. `endTime`

Per un processo in cui la configurazione della pianificazione opzionale viene eseguita durante una finestra di manutenzione ricorrente in una località in cui vige l'ora legale (DST), la durata cambierà di un'ora quando si passa dall'ora legale all'ora solare e dall'ora solare all'ora legale.

> **Note**
>
> Il fuso orario visualizzato in Console di gestione AWS è il fuso orario corrente del sistema. Tuttavia, questi fusi orari verranno convertiti in UTC nel sistema.

Comportamento di fine

Il comportamento di fine di un processo pianificato determina cosa accade al processo e a tutte le esecuzioni del processo non completate quando il processo raggiunge `endTime` selezionata.

Di seguito sono elencati i comportamenti finali che puoi utilizzare durante la creazione del processo o del modello di processo:

- `STOP_ROLLOUT`
  - `STOP_ROLLOUT` interrompe il rollout del documento del processo in tutti i dispositivi rimanenti nel gruppo di destinazione del processo. Inoltre, tutte le esecuzioni del processo `QUEUED` e `IN_PROGRESS` continueranno finché non raggiungono uno stato terminale. Questo è il comportamento di fine predefinito a meno che non si selezioni `CANCEL` o `FORCE_CANCEL`.
- `CANCEL`
  - `CANCEL` interrompe il rollout del documento del processo in tutti i dispositivi rimanenti nel gruppo di destinazione del processo. Inoltre, tutte le esecuzioni del processo `QUEUED` verranno annullate mentre tutte le esecuzioni del processo `IN_PROGRESS` continueranno finché non raggiungono uno stato terminale.
- `FORCE_CANCEL`

- `FORCE_CANCEL` interrompe il rollout del documento del processo in tutti i dispositivi rimanenti nel gruppo di destinazione del processo. Inoltre, tutte le esecuzioni del processo `QUEUED` e `IN_PROGRESS` verranno annullate.

> **Note**
>
> Per selezionare un`endbehavior`, è necessario selezionare un `endtime`

Durata massima

La durata massima di un processo pianificato deve essere inferiore o uguale a due anni a prescindere da `startTime` e `endTime`.

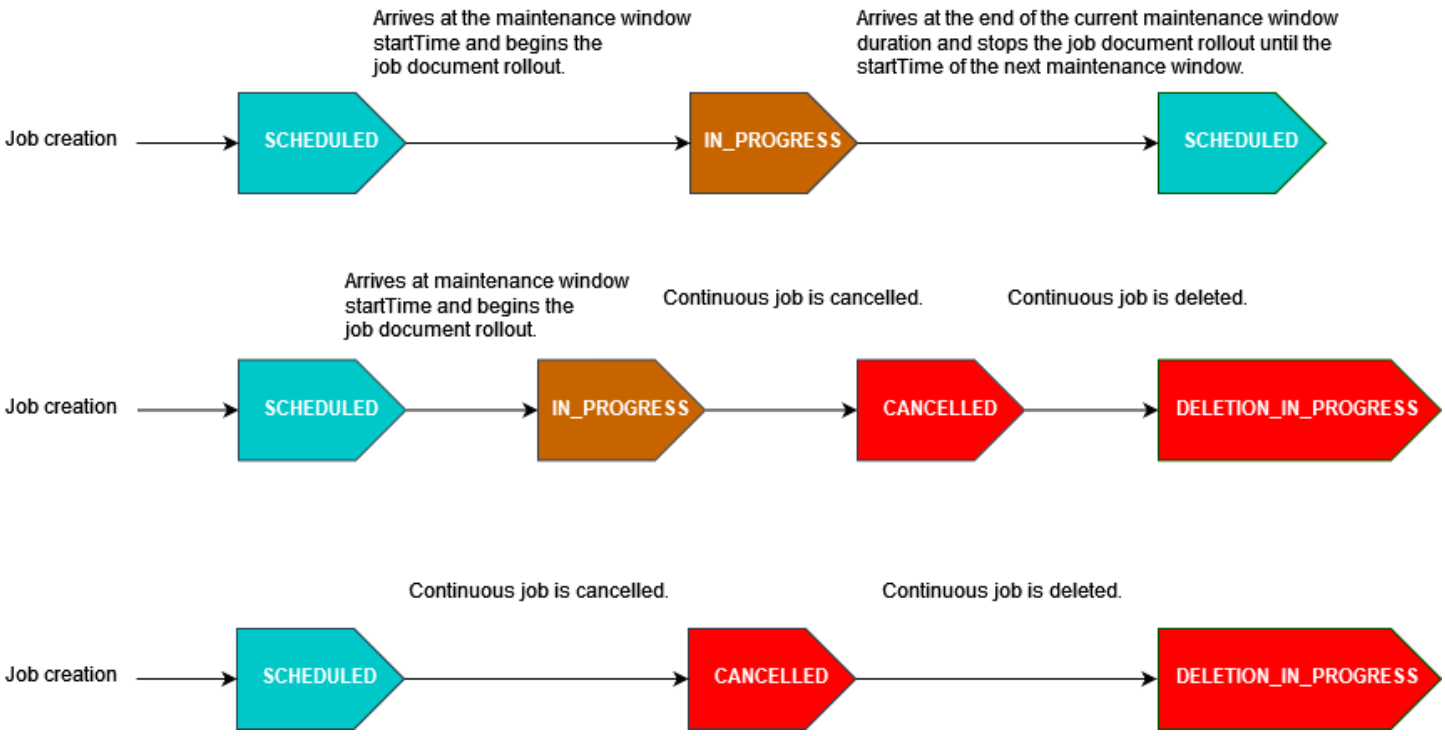
Nella tabella seguente sono elencati gli scenari di durata comuni di un processo pianificato:

Numero del processo pianificato di esempio	startTime	endTime	Durata massima
1	Subito dopo la creazione del processo iniziale.	Un anno dopo la creazione del processo iniziale.	Un anno
2	Un mese dopo la creazione del processo iniziale.	13 mesi dopo la creazione del processo iniziale.	Un anno
3	Un anno dopo la creazione del processo iniziale.	Due anni dopo la creazione del processo iniziale.	Un anno
4	Subito dopo la creazione del processo iniziale.	Due anni dopo la creazione del processo iniziale.	Due anni

Finestra di manutenzione ricorrente

La finestra di manutenzione è una configurazione opzionale all'interno della configurazione di pianificazione di Console di gestione AWS e all'`SchedulingConfig`interno della `CreateJob` e `CreateJobTemplate` APIs. È possibile impostare una finestra di manutenzione ricorrente con un'ora di inizio, una durata e una frequenza predeterminate (giornaliera, settimanale o mensile) in cui si verifica la finestra di manutenzione. Le finestre di manutenzione si applicano solo ai lavori continui. La durata massima di una finestra di manutenzione ricorrente è di 23 ore e 50 minuti.

Nel diagramma seguente vengono illustrati gli stati del processo per diversi scenari di processi pianificati con una finestra di manutenzione opzionale:



Per ulteriori informazioni sugli stati dei processi, consulta [Processi e stati di esecuzione dei processi].

> **Note**
>
> Se un lavoro arriva a `endTime` durante una finestra di manutenzione, verrà aggiornato da `IN_PROGRESS` a `COMPLETED`. Inoltre, tutte le esecuzioni del processo rimanenti seguiranno il `endBehavior` per il processo.

Espressioni cron

Per i processi pianificati che distribuiscono il documento di processo durante una finestra di manutenzione con una frequenza personalizzata, la frequenza personalizzata viene immessa utilizzando un'espressione cron. Un'espressione cron contiene sei campi obbligatori che sono separati da uno spazio vuoto.

Sintassi

```
cron(fields)
```

Campo	Valori	Caratteri jolly
Minuti	0-59	, - * /
Ore	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
Mese	1-12 o JAN-DEC	, - * /
D ay-of-week	1-7 o SUN-SAT	, - * ? L #
Anno	1970-2199	, - * /

Caratteri jolly

- Il carattere jolly , (virgola) include valori aggiuntivi. Nel campo Month (Mese), JAN,FEB,MAR (GEN,FEB,MAR) include gennaio, febbraio e marzo.
- Il carattere jolly - (trattino) specifica gli intervalli. Nel campo Day (Giorno), 1-15 include i giorni dall'1 al 15 del mese specificato.
- Il carattere jolly * (asterisco) include tutti i valori nel campo. Nel campo Hours (Ore), * include ogni ora. Non puoi usare * in entrambi i Day-of-week campi Day-of-month e. Se viene utilizzato in uno di tali campi, è necessario utilizzare ? nell'altro.
- Il carattere jolly / (barra) specifica gli incrementi. Nel campo Minutes (Minuti), puoi inserire 1/10 per specificare ogni decimo minuto, a partire dal primo minuto dell'ora (ad esempio, l'11°, il 21° e il 31° minuto e così via).
- Il carattere jolly ? (punto interrogativo) specifica un valore. Nel Day-of-month campo puoi inserire 7 e se non ti interessa in che giorno della settimana è il 7, puoi inserire? sul Day-of-week campo.

- Il carattere jolly L nel campo Day-of-month o Day-of-week specifica l'ultimo giorno del mese o della settimana.
- Il carattere **`W`** jolly nel Day-of-month campo specifica un giorno della settimana. Nel Day-of-month campo, **`3W`** specifica il giorno della settimana più vicino al terzo giorno del mese.
- Il carattere jolly # nel Day-of-week campo specifica una determinata istanza del giorno della settimana specificato nell'arco di un mese. Ad esempio, 3#2 sarebbe il secondo martedì del mese: il 3 fa riferimento a martedì perché è il terzo giorno di ogni settimana e il 2 fa riferimento al secondo giorno di questo tipo in un mese.

> **Note**
>
> Se si utilizza un carattere '#', è possibile definire una sola espressione nel day-of-week campo. Ad esempio, `"3#1,6#3"` non è valido perché viene interpretato come due espressioni.

Restrizioni

- Non puoi specificare i campi Day-of-month e Day-of-week nella stessa espressione cron. Se specifichi un valore (o un *) in uno dei campi, devi usare un carattere ? nell'altro campo.

Esempi

Fare riferimento alle stringhe cron di esempio seguenti durante l'utilizzo di un'espressione cron per la `startTime` di una finestra di manutenzione ricorrente.

Minuti	Ore	Giorno del mese	Mese	Giorno della settimana	Anno	Significato
0	10	*	*	?	*	Esegui ogni giorno alle 10:00 (UTC)
15	12	*	*	?	*	Esegui ogni giorno

Minuti	Ore	Giorno del mese	Mese	Giorno della settimana	Anno	Significato
						alle 12:15 (UTC)
0	18	?	*	LUN-VEN	*	Esegui dal lunedì al venerdì alle 18:00 (UTC)
0	8	1	*	?	*	Esegui ogni primo giorno del mese alle 8.00 (UTC)

Logica di fine durata della finestra di manutenzione ricorrente

Quando la distribuzione di un processo durante una finestra di manutenzione raggiunge la durata di occorrenza della finestra di manutenzione corrente, si verificheranno le seguenti azioni:

- Il processo interromperà tutte le distribuzioni del documento del processo negli eventuali dispositivi rimanenti nel gruppo di destinazione. Riprenderà alle `startTime` della finestra di manutenzione successiva.
- Tutte le esecuzioni dei processi con stato `QUEUED` rimarranno nello stato `QUEUED` fino al `startTime` della finestra di manutenzione successiva. Nella finestra successiva, possono passare a `IN_PROGRESS` quando il dispositivo è pronto per iniziare a eseguire le azioni specificate nel documento di processo.
- Tutte le esecuzioni del processo con uno stato di `IN_PROGRESS` continueranno a eseguire le azioni specificate nel documento del processo finché non raggiungono uno stato terminale. Qualsiasi tentativo di ripetizione come specificato in `JobExecutionsRetryConfig` verrà eseguito alle `startTime` della finestra di manutenzione successiva.

Configurazione dell'interruzione del processo

Utilizza questa configurazione per creare un criterio per annullare un processo quando una percentuale di soglia dei dispositivi soddisfa tale criterio. Ad esempio, puoi utilizzare questa configurazione per annullare un processo nei seguenti casi:

- Quando una percentuale soglia di dispositivi non riceve le notifiche di esecuzione del lavoro, ad esempio quando il dispositivo non è compatibile per un aggiornamento Over-The-Air (OTA). In questo caso, il dispositivo può segnalare uno stato `REJECTED`.
- Quando una percentuale di soglia dei dispositivi segnala un errore nell'esecuzione del processo, ad esempio nel caso in cui il dispositivo rilevi una disconnessione quando tenta di effettuare il download del documento di processo da un URL Amazon S3. In questi casi, il dispositivo deve essere programmato in modo da segnalare a AWS IoT lo stato `FAILURE`.
- Quando viene segnalato lo stato `TIMED_OUT` in seguito al timeout dell'esecuzione del processo per una percentuale di soglia dei dispositivi dopo l'avvio dell'esecuzione del processo.
- Quando si verificano più tentativi falliti. Quando aggiungi una configurazione di nuovo tentativo, ogni nuovo tentativo può comportare costi aggiuntivi per il tuo Account AWS. In questi casi, l'annullamento del processo può annullare le esecuzioni del processo in coda ed evitare nuovi tentativi per queste esecuzioni. Per ulteriori informazioni sulla configurazione del nuovo tentativo e sul suo utilizzo con la configurazione di interruzione, consulta Configurazioni di timeout e di nuovo tentativo di esecuzione del processo.

Puoi impostare una condizione di interruzione del lavoro utilizzando la AWS IoT console o l'API AWS IoT Jobs.

Configurazioni di timeout e di nuovo tentativo di esecuzione del processo

Utilizza la configurazione del timeout di esecuzione del processo per ricevere Notifiche dei processi quando l'esecuzione di un processo è in corso per un periodo più lungo della durata impostata. Utilizza la configurazione del nuovo tentativo di esecuzione del processo per ritentare l'esecuzione quando il processo fallisce o scade.

Configurazione del timeout di esecuzione del processo

Utilizza la configurazione del timeout di esecuzione del processo per ricevere notifiche ogni volta che l'esecuzione di un processo si blocca nello stato `IN_PROGRESS` per un periodo di tempo inaspettatamente lungo. Quando il processo è `IN_PROGRESS`, è possibile monitorare il progresso dell'esecuzione del processo.

Timer per i timeout di processo

Sono disponibili due tipi di timer: in corso e della fase.

Timer in corso

Quando crei un processo o un modello di processo, puoi specificare un valore per il timer in corso compreso tra 1 minuto e 7 giorni. Puoi aggiornare il valore di questo timer fino all'avvio dell'esecuzione del processo. Dopo l'avvio del timer, il valore non può più essere aggiornato e il valore del timer viene applicato a tutte le esecuzioni del processo. Ogni volta che l'esecuzione di un job rimane nello `IN_PROGRESS` stato per un periodo superiore a questo intervallo, l'esecuzione fallisce e passa allo stato del terminale. `TIMED_OUT` AWS IoT pubblica anche una notifica MQTT.

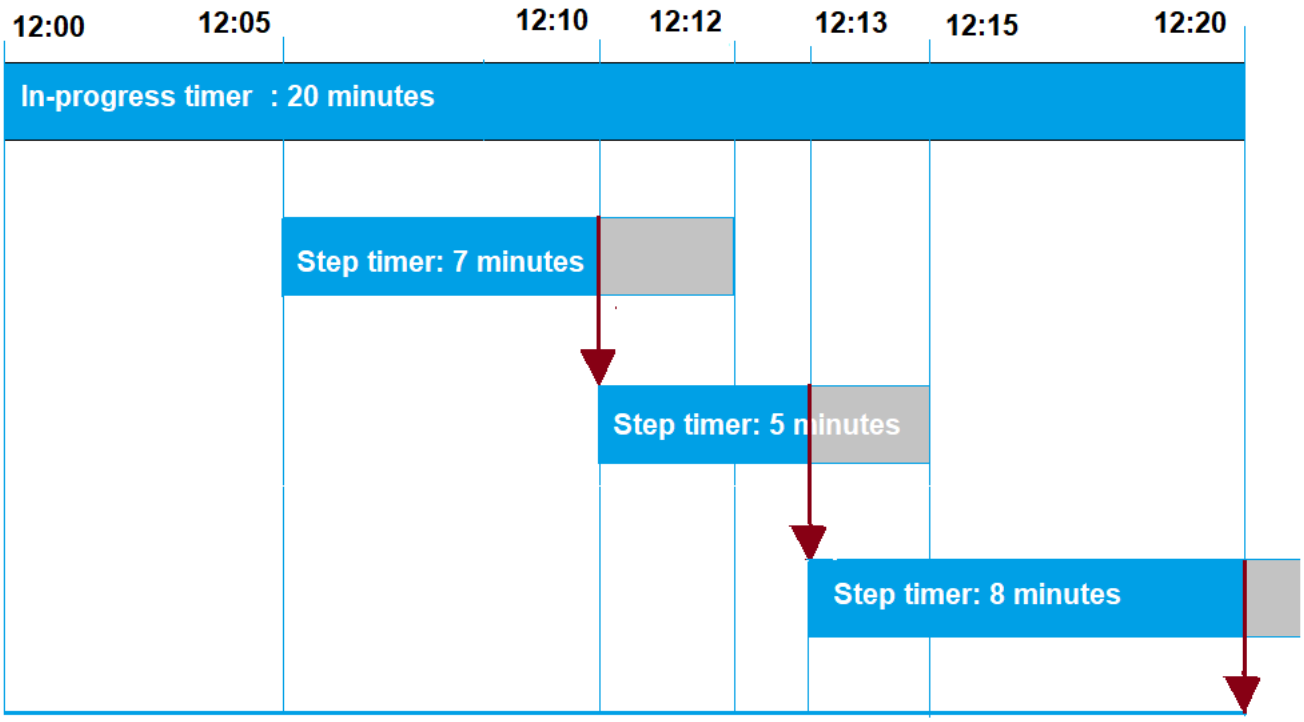
Timer della fase

Puoi anche impostare un timer della fase applicabile solo all'esecuzione del processo che desideri aggiornare. Questo timer non ha alcun effetto sul timer in corso. Puoi impostare un nuovo valore per questo timer ogni volta che aggiorni l'esecuzione di un processo. Puoi anche creare un nuovo timer della fase all'avvio della successiva esecuzione del processo in sospeso per un oggetto. Se l'esecuzione del processo resta nello stato `IN_PROGRESS` per un periodo di tempo superiore a quello consentito dall'intervallo del timer della fase, l'esecuzione del processo non va a buon fine e viene impostato lo stato `TIMED_OUT` terminale.

> **Note**
>
> È possibile impostare il timer in corso utilizzando la AWS IoT console o l' AWS IoT API Jobs. Per specificare il timer della fase, utilizza l'API.

Come funzionano i timer per i timeout di processo

Il seguente esempio illustra le varie interazioni fra i timeout in corso e quelli della fase in un periodo di timeout di 20 minuti.



Di seguito vengono illustrate le diverse fasi:

1. 12:00

   Viene creato un nuovo processo; durante la creazione dello stesso, viene avviato un timer in corso per venti minuti. Il timer in corso avvia l'esecuzione e l'esecuzione del processo passa allo stato `IN_PROGRESS`.

2. 12:05

   Viene creato un nuovo timer della fase con un valore di 7 minuti. L'esecuzione del processo ora scadrà alle 12:12.

3. 12:10

   Viene creato un nuovo timer della fase con un valore di 5 minuti. Quando viene creato il nuovo timer della fase, il timer della fase precedente viene eliminato e l'esecuzione del processo ora scadrà alle 12:15.

4. 12:13

Viene creato un nuovo timer della fase con un valore di 9 minuti. Il timer della fase precedente viene eliminato e l'esecuzione del processo ora scadrà alle 12:20, perché il timer in corso scadrà alle 12:20. Il timer della fase non può superare il limite assoluto creato dal timer in corso.

Configurazione del nuovo tentativo di esecuzione del processo

Puoi utilizzare la configurazione del nuovo tentativo per ritentare l'esecuzione del processo quando viene soddisfatto un determinato set di criteri. È possibile effettuare un nuovo tentativo quando un processo scade o quando un dispositivo sperimenta un errore. Per ritentare l'esecuzione a causa di un errore di timeout, è necessario abilitare la configurazione del timeout.

Come utilizzare la configurazione del nuovo tentativo

Segui la procedura riportata di seguito per questa configurazione:

1. Stabilisci se utilizzare la configurazione di nuovo tentativo per `FAILED`, per `TIMED_OUT` o per entrambi i criteri di errore. Per quanto riguarda lo `TIMED_OUT` stato, dopo la segnalazione dello stato, AWS IoT Jobs riprova automaticamente l'esecuzione del lavoro per il dispositivo.
2. Per lo stato `FAILED`, controlla se l'esecuzione del processo fallita può essere ritentata. Se l'errore è ritentabile, programma il dispositivo in modo da segnalare a AWS IoT lo stato `FAILURE`. Nella sezione seguente vengono fornite ulteriori informazioni sugli errori ritentabili e non ritentabili.
3. Specifica il numero di tentativi da effettuare per ciascun tipo di errore utilizzando le informazioni precedenti. Per ciascun dispositivo, puoi specificare fino a 10 tentativi per entrambi i tipi di errore combinati. I nuovi tentativi si arrestano automaticamente quando un'esecuzione ha esito positivo o al raggiungimento del numero di tentativi specificato.
4. Aggiungi una configurazione di interruzione per annullare il processo in caso di tentativi ripetuti non riusciti per evitare di incorrere in addebiti aggiuntivi se il numero di nuovi tentativi è elevato.

> **Note**
>
> Quando un processo raggiunge la fine di un'occorrenza della finestra di manutenzione ricorrente, tutte le esecuzioni del processo `IN_PROGRESS` continueranno a eseguire le azioni identificate nel documento del processo finché non raggiungono lo stato terminale. Se l'esecuzione di un processo raggiunge uno stato terminale di `FAILED` o `TIMED_OUT` all'esterno di una finestra di manutenzione, si verificherà un tentativo di ripetizione nella finestra di manutenzione successiva, se i tentativi di ripetizione non sono esauriti.

All'`startTime` della successiva finestra di manutenzione, verrà creata una nuova esecuzione del processo che avrà lo stato di `QUEUED` finché il dispositivo non è pronto per iniziare.

Configurazione con nuovo tentativo e interruzione

Ogni nuovo tentativo comporta costi aggiuntivi per il tuo. Account AWS Per evitare costi aggiuntivi dovuti a ripetuti tentativi falliti, ti consigliamo di aggiungere una configurazione di interruzione. Per ulteriori informazioni sui prezzi, consulta Prezzi di AWS IoT Device Management.

È possibile che si verifichino più tentativi falliti quando un'elevata percentuale di soglia dei dispositivi scade o segnala un errore. In questo caso, puoi utilizzare la configurazione di interruzione per annullare il processo ed evitare eventuali esecuzioni del processo in coda o ulteriori tentativi.

> **Note**
>
> Quando vengono soddisfatti i criteri di interruzione per l'annullamento dell'esecuzione di un processo, vengono annullate solo le esecuzioni del processo `QUEUED`. Gli eventuali tentativi in coda per il dispositivo non vengono effettuati. Tuttavia, le attuali esecuzioni del processo con stato `IN_PROGRESS` non vengono annullate.

Prima di ritentare l'esecuzione di un processo non riuscita, ti consigliamo di verificare anche se l'errore dell'esecuzione del processo è ritentabile, come descritto nella sezione seguente.

Nuovo tentativo per il tipo di errore **`FAILED`**

Per effettuare nuovi tentativi per il tipo di errore `FAILED`, i dispositivi devono essere programmati in modo da segnalare a AWS IoT lo stato `FAILURE` se l'esecuzione di un processo è fallita. Imposta la configurazione del nuovo tentativo specificando i criteri per ritentare le esecuzioni del processo `FAILED` e il numero di tentativi da effettuare. Quando AWS IoT Jobs rileva lo `FAILURE` stato, tenterà automaticamente di ripetere l'esecuzione del processo per il dispositivo. I nuovi tentativi continuano fino a quando l'esecuzione del processo non ha esito positivo o fino al raggiungimento del numero massimo di nuovi tentativi.

È possibile monitorare di ciascun nuovo tentativo e il processo in esecuzione su questi dispositivi. Monitorando lo stato di esecuzione, dopo avere effettuato il numero specificato di tentativi, puoi utilizzare il dispositivo per segnalare gli errori e avviare un nuovo tentativo.

Errori ritentabili e non ritentabili

Un errore di esecuzione del processo può essere ripetibile o non ripetibile. Ogni nuovo tentativo può comportare costi aggiuntivi per il tuo Account AWS. Per evitare addebiti aggiuntivi per l'esecuzione di molteplici nuovi tentativi, ti consigliamo innanzitutto di verificare se l'errore di esecuzione del processo è ritentabile. Ad esempio, è ripetibile un errore di connessione rilevato dal dispositivo durante il tentativo di effettuare il download del documento di processo da un URL Amazon S3. Se in base all'errore è possibile ritentare l'esecuzione del processo, puoi programmare il dispositivo in modo da segnalare lo stato di `FAILURE` nel caso in cui l'esecuzione del processo riceva un errore. Quindi, imposta la configurazione dei tentativi per ritentare le esecuzioni `FAILED`.

Se l'esecuzione del processo non può essere ritentata, per evitare di ripetere il tentativo e sostenere potenziali costi aggiuntivi sul tuo account, ti consigliamo di programmare il dispositivo in modo da segnalare lo stato `REJECTED` a AWS IoT. Esempi di errori non ritentabili sono i casi in cui il dispositivo non è compatibile con la ricezione di un aggiornamento del processo o quando si verifica un errore di memoria durante l'esecuzione di un processo. In questi casi, AWS IoT Jobs non ritenterà l'esecuzione del lavoro perché ritenta l'esecuzione del lavoro solo quando rileva uno stato or. `FAILED` `TIMED_OUT`

Dopo avere stabilito che un errore di esecuzione del processo è ritentabile, se il nuovo tentativo continua a non riuscire, prova a esaminare i registri del dispositivo.

> **Note**
>
> Quando un processo con la configurazione di pianificazione opzionale raggiunge la sua `endTime`, il `endBehavior` selezionato interromperà l'implementazione del documento di processo in tutti i dispositivi rimanenti del gruppo di destinazione e indicherà come procedere con le esecuzioni del processo rimanenti. Per poter essere ripetuti, i tentativi devono essere selezionati tramite la configurazione dei nuovi tentativi.

Nuovo tentativo per il tipo di errore **`TIMEOUT`**

Se abiliti il timeout durante la creazione di un lavoro, AWS IoT Jobs tenterà di riprovare l'esecuzione del lavoro per il dispositivo quando lo stato cambia da a. `IN_PROGRESS` `TIMED_OUT` Questa modifica dello stato può verificarsi quando il timer in corso scade o quando un timer della fase specificato è in stato `IN_PROGRESS` e successivamente scade. I nuovi tentativi continuano fino a quando l'esecuzione del processo non ha esito positivo o fino al raggiungimento del numero massimo di nuovi tentativi per questo tipo di errore.

Processi continui e aggiornamenti dell'appartenenza a gruppi di oggetti

Per i processi continui con stato di processo `IN_PROGRESS`, il numero di nuovi tentativi viene azzerato quando l'appartenenza al gruppo di un oggetto viene aggiornata. Ad esempio, poniamo che tu abbia specificato cinque nuovi tentativi e tre tentativi siano già stati effettuati. Se un oggetto viene ora rimosso da un gruppo di oggetti e successivamente aggiunto di nuovo allo stesso gruppo, come nei gruppi di oggetti dinamici, il numero di nuovi tentativi viene azzerato. Ora per il tuo gruppo di oggetti puoi effettuare cinque tentativi invece dei due rimasti. Inoltre, quando un oggetto viene rimosso da un gruppo di oggetti, eventuali nuovi tentativi vengono annullati.

Specificare configurazioni aggiuntive

Puoi specificare queste configurazioni aggiuntive quando crei un processo o un modello di processo. Di seguito viene illustrato quando è possibile specificare queste configurazioni.

- Al momento della creazione di un modello di processo personalizzato. Le impostazioni di configurazione aggiuntive specificate vengono salvate quando crei un processo dal modello.
- Al momento della creazione di un processo utilizzando un file di processo. Il file di processo può essere un file JSON caricato in un bucket S3.
- Al momento della creazione di un processo personalizzato utilizzando un modello di processo personalizzato. Se nel modello sono già state specificate queste impostazioni, puoi riutilizzarle o sostituirle specificando le nuove impostazioni di configurazione.
- Quando si crea un lavoro personalizzato utilizzando un modello AWS gestito.

Argomenti

- Specificare le configurazioni dei lavori utilizzando il Console di gestione AWS
- Specificate le configurazioni dei lavori utilizzando l'API AWS IoT Jobs

Specificare le configurazioni dei lavori utilizzando il Console di gestione AWS

È possibile aggiungere le diverse configurazioni per il lavoro utilizzando la AWS IoT console. Dopo avere creato un processo, puoi visualizzare i dettagli sullo stato delle configurazioni del processo nella pagina dei dettagli del processo. Per ulteriori informazioni sulle diverse configurazioni e su come funzionano, consulta Come funzionano le configurazioni di processo.

Aggiungi le configurazioni del processo quando crei un processo o un modello di processo.

Al momento della creazione di un modello di processo personalizzato

Per impostare la configurazione di rollout durante la creazione di un modello di processo personalizzato

1. Vai all'hub Job templates della AWS IoT console e scegli Crea modello di lavoro.
2. Specifica le proprietà del modello di processo, fornisci il documento di processo, espandi la configurazione che desideri aggiungere e quindi specifica i parametri di configurazione.

Al momento della creazione di un processo personalizzato

Per impostare la configurazione di rollout durante la creazione di un processo personalizzato

1. Vai al Job hub della AWS IoT console e scegli Crea lavoro.
2. Scegli Create a custom job (Crea un processo personalizzato) e specifica le proprietà del processo, le destinazioni e se utilizzare un modello o un file di processo per il documento di processo. Puoi utilizzare un modello personalizzato o un modello AWS gestito.
3. Scegli la configurazione del processo e quindi espandi Rollout configuration (Configurazione del rollout) per specificare se utilizzare Constant rate (Velocità costante) o Exponential rate (Velocità esponenziale). Specifica quindi i parametri di configurazione.

La sezione successiva mostra quali parametri puoi specificare per ogni configurazione.

Configurazione del rollout

Puoi specificare se utilizzare una velocità di rollout costante o esponenziale.

- Impostazione di una velocità di rollout costante

  Per impostare una velocità costante per le esecuzioni dei processi, scegli Constant rate (Velocità costante) e quindi specifica il valore Maximum per minute (Massimo al minuto) per il limite di velocità superiore. Questo valore è facoltativo e varia da 1 a 1.000. Se non lo imposti, viene utilizzato 1.000 come valore predefinito.
- Impostazione di una velocità di rollout esponenziale

  Per impostare una velocità esponenziale, scegli Exponential rate (Velocità esponenziale) e quindi specifica questi parametri:
  - Velocità base al minuto

Velocità alla quale i processi vengono eseguiti fino al raggiungimento della soglia di Number of notified devices (Numero di dispositivi notificati) o Number of succeeded devices (Numero di dispositivi con esito positivo) per Rate increase criteria (Criterio di incremento della velocità).

- Fattore di incremento

  Il fattore esponenziale con il quale la velocità di rollout aumenta dopo che la soglia di Number of notified devices (Numero di dispositivi notificati) o Number of succeeded devices (Numero di dispositivi riusciti) viene raggiunta per Rate increase criteria (Criterio di incremento della velocità).

- Criteri di incremento della velocità

  La soglia per il valore Number of notified devices (Numero di dispositivi notificati) o Number of succeeded devices (Numero di dispositivi riusciti).

Configurazione dell'interruzione

Scegli Add new configuration (Aggiungi nuova configurazione) e specifica i seguenti parametri per ogni configurazione:

- Tipo di errore

  Specifica i tipi di errore che avviano l'interruzione di un processo. Il valore può essere: FAILED, REJECTED, TIMED_OUT o ALL.

- Fattore di incremento

  Specifica il numero di esecuzioni del processo completate che devono verificarsi prima che il criterio di interruzione del processo sia soddisfatto.

- Percentuale di soglia

  Specifica il numero totale di oggetti eseguiti che avvia l'interruzione del processo.

Scheduling configuration (Configurazione della pianificazione)

Ogni processo può iniziare immediatamente dopo la creazione iniziale, essere pianificato per iniziare in una data e ora successive o essere eseguito durante una finestra di manutenzione ricorrente.

Scegli Add new configuration (Aggiungi nuova configurazione) e specifica i seguenti parametri per ogni configurazione:

- Avvio del processo

  Specificare la data e l'ora di avvio del processo.

- Finestra di manutenzione ricorrente

  Una finestra di manutenzione ricorrente definisce la data e l'ora specifiche in cui un processo può distribuire il documento del processo ai dispositivi di destinazione nel processo. La finestra di manutenzione può essere ripetuta ogni giorno, ogni settimana, ogni mese o in un giorno e ora ricorrenti personalizzati.

- Fine processo

  Specificare la data e l'ora di fine del processo.

- Comportamento di fine processo

  Selezionare un comportamento di fine per tutte le esecuzioni del processo non completate al termine del processo.

> **Note**
>
> Quando un processo con configurazione di pianificazione opzionale e ora di fine selezionata raggiunge l'ora di fine, il processo interromperà il rollout in tutti i dispositivi rimanenti del gruppo di destinazione. Inoltre, sfrutta il comportamento finale selezionato su come procedere con le restanti esecuzioni dei processi e i nuovi tentativi relativi in base alla configurazione dei nuovi tentativi.

Configurazione del timeout

Per impostazione predefinita, il timeout non è impostato e il processo viene annullato o eliminato. Per utilizzare i timeout, scegli Enable timeout (Abilita il timeout) e specifica un valore di timeout compreso tra 1 minuto e 7 giorni.

Configurazione del nuovo tentativo

> **Note**
>
> Dopo avere creato un processo, il numero di nuovi tentativi non può essere modificato. È possibile rimuovere la configurazione del nuovo tentativo solo per tutti i tipi di errore. Quando

crei un processo, valuta un numero di nuovi tentativi appropriato per la configurazione. Per evitare costi eccessivi a causa di potenziali tentativi falliti, puoi aggiungere una configurazione di interruzione.

Scegli Add new configuration (Aggiungi nuova configurazione) e specifica i seguenti parametri per ogni configurazione:

- Tipo di errore

  Specifica i tipi di errore che attivano un nuovo tentativo di esecuzione del processo. Il valore può essere: Failed (Fallito), Timeout (scaduto) e All (Tutti).
- Numero di tentativi

  Specifica il numero di tentativi per il tipo di errore selezionato. Per entrambi i tipi di errore combinati, è possibile effettuare fino a 10 nuovi tentativi.

Specificate le configurazioni dei lavori utilizzando l'API AWS IoT Jobs

È possibile utilizzare CreateJobo l'CreateJobTemplateAPI per specificare le diverse configurazioni di lavoro. Le sezioni seguenti descrivono come aggiungere queste configurazioni. Dopo aver aggiunto le configurazioni, puoi utilizzarle JobExecutionSummarye JobExecutionSummaryForJobvisualizzarne lo stato.

Per ulteriori informazioni sulle diverse configurazioni e su come funzionano, consulta Come funzionano le configurazioni di processo.

Rollout configuration (Configurazione rollout)

Per la configurazione di velocità puoi specificare una velocità di rollout costante o esponenziale.

- Impostazione di una velocità di rollout costante

  Per impostare una velocità di rollout costante, utilizza l'oggetto `JobExecutionsRolloutConfig` per aggiungere il parametro `maximumPerMinute` alla richiesta `CreateJob`. Questo parametro specifica il limite superiore della velocità alla quale possono verificarsi le esecuzioni dei processi. Questo valore è facoltativo e varia da 1 a 1.000. Se non lo imposti, viene utilizzato 1.000 come valore predefinito.

```
    "jobExecutionsRolloutConfig": {
```

```
        "maximumPerMinute": 1000
    }
```

- Impostazione di una velocità di rollout esponenziale

  Per impostare una velocità di rollout del processo variabile, utilizza l'oggetto `JobExecutionsRolloutConfig`. Puoi configurare la proprietà `ExponentialRolloutRate` quando esegui l'operazione API `CreateJob`. L'esempio seguente imposta una velocità di rollout esponenziale utilizzando il parametro `exponentialRate`. Per ulteriori informazioni sui parametri, consulta `ExponentialRolloutRate`.

```
{
...
  "jobExecutionsRolloutConfig": {
    "exponentialRate": {
      "baseRatePerMinute": 50,
      "incrementFactor": 2,
      "rateIncreaseCriteria": {
        "numberOfNotifiedThings": 1000,
        "numberOfSucceededThings": 1000
      },
      "maximumPerMinute": 1000
    }
  }
...
}
```

Dove il parametro:

baseRatePerMinuto

Specifica la velocità alla quale i processi vengono eseguiti fino a quando non viene raggiunta la soglia di `numberOfNotifiedThings` o `numberOfSucceededThings`.

incrementFactor

Specifica il fattore esponenziale con il quale la velocità di rollout aumenta dopo che è stata raggiunta la soglia di `numberOfNotifiedThings` o `numberOfSucceededThings`.

rateIncreaseCriteria

Specifica la soglia di `numberOfNotifiedThings` o `numberOfSucceededThings`.

Configurazione dell'interruzione

Per aggiungere questa configurazione utilizzando l'API, specifica il parametro `AbortConfig` quando esegui `CreateJob` o l'operazione API `CreateJobTemplate`. L'esempio seguente mostra una configurazione di interruzione per il rollout di un processo in cui si verificano più esecuzioni fallite, come specificato con l'operazione API `CreateJob`.

> **Note**
>
> L'eliminazione dell'esecuzione di un processo influisce sul valore del calcolo dell'esecuzione totale completata. Quando un processo viene interrotto, il servizio crea codici `comment` e `reasonCode` automaticamente per differenziare una cancellazione dipendente dall'utente o una cancellazione per interruzione di processo.

```
"abortConfig": {
   "criteriaList": [
      {
         "action": "CANCEL",
         "failureType": "FAILED",
         "minNumberOfExecutedThings": 100,
         "thresholdPercentage": 20
      },
      {
         "action": "CANCEL",
         "failureType": "TIMED_OUT",
         "minNumberOfExecutedThings": 200,
         "thresholdPercentage": 50
      }
   ]
 }
```

Dove il parametro:

action

Specifica l'operazione da eseguire quando vengono soddisfatti i criteri di interruzione. Questo parametro è obbligatorio ed `CANCEL` è il solo valore valido.

failureType

Specifica i tipi di errore che devono avviare l'interruzione di un processo. I valori validi sono `FAILED`, `REJECTED`, `TIMED_OUT` e `ALL`.

minNumberOfExecutedThings

Specifica il numero di esecuzioni del processo completate che devono verificarsi prima che il criterio di interruzione del processo sia soddisfatto. In questo esempio, AWS IoT non verifica se un'interruzione di processo si verifica quando almeno 100 dispositivi hanno completato le esecuzioni di processo.

thresholdPercentage

Specifica il numero totale di oggetti per i quali i processi sono stati eseguiti che avvia l'interruzione del processo. In questo esempio, AWS IoT verifica in sequenza e avvia un'interruzione del lavoro se viene raggiunta la percentuale di soglia. Se almeno il 20% delle esecuzioni complete non è riuscito dopo il completamento di 100 esecuzioni, annulla il rollout del processo. Se questo criterio non AWS IoT viene soddisfatto, verifica se almeno il 50% delle esecuzioni completate è scaduto dopo il completamento di 200 esecuzioni. In questo caso, annulla il rollout del processo.

Scheduling configuration (Configurazione della pianificazione)

Per aggiungere questa configurazione utilizzando l'API, specifica il parametro `SchedulingConfig` opzionale quando esegui `CreateJob` o l'operazione API `CreateJobTemplate`.

```
  "SchedulingConfig": {
    "endBehavior": string
    "endTime": string
    "maintenanceWindows": string
    "startTime": string
  }
```

Dove il parametro:

startTime

Specifica la data e l'ora di avvio del processo.

endTime

Specifica la data e l'ora di fine del processo.

maintenanceWindows

Specifica se è stata selezionata una finestra di manutenzione opzionale per il processo pianificato di distribuzione del documento del processo in tutti i dispositivi del gruppo di destinazione. Il formato stringa per `maintenanceWindow` è YYYY/MM/DD per la data e hh:mm per l'ora.

endBehavior

Specifica il comportamento del processo per un processo pianificato al raggiungimento di `endTime`.

> **Note**
>
> L'opzione `SchedulingConfig` relativa a un lavoro è visualizzabile nella finestra e. `DescribeJobDescribeJobTemplate` APIs

Configurazione del timeout

Per aggiungere questa configurazione utilizzando l'API, specifica il parametro `TimeoutConfig` opzionale quando esegui `CreateJob` o l'operazione API `CreateJobTemplate`.

Per utilizzare la configurazione di timeout

1. Per impostare il timer in corso durante la creazione di un lavoro o di un modello di lavoro, impostate un valore per la `inProgressTimeoutInMinutes` proprietà dell'oggetto opzionale TimeoutConfig.

   ```
   "timeoutConfig": {
     "inProgressTimeoutInMinutes": number
   }
   ```

2. Per specificare uno step timer per l'esecuzione di un lavoro, impostate un valore per `stepTimeoutInMinutes` quando chiamate UpdateJobExecution. Il timer della fase si applica solo all'esecuzione del processo che stai aggiornando. È possibile impostare un nuovo valore per questo timer ogni volta che si aggiorna l'esecuzione di un processo.

> **Note**
>
> `UpdateJobExecution` può eliminare un timer della fase già esistente creandone uno nuovo con un valore di -1.

```
{
   ...
    "statusDetails": {
      "string" : "string"
   },
   "stepTimeoutInMinutes": number
}
```

3. Per creare un nuovo step timer, puoi anche chiamare l'operazione StartNextPendingJobExecutionAPI.

Configurazione del nuovo tentativo

> **Note**
>
> Quando crei un processo, valuta un numero di nuovi tentativi appropriato per la configurazione. Per evitare costi eccessivi a causa di potenziali tentativi falliti, puoi aggiungere una configurazione di interruzione. Dopo avere creato un processo, il numero di nuovi tentativi non può essere modificato. È possibile impostare il numero di tentativi su 0 solo utilizzando l'operazione UpdateJobAPI.

Per aggiungere questa configurazione utilizzando l'API, specifica il parametro `jobExecutionsRetryConfig` opzionale quando esegui `CreateJob` o l'operazione API `CreateJobTemplate`.

```
{
...
  "jobExecutionsRetryConfig": {
      "criteriaList": [
         {
            "failureType": "string",
```

```
            "numberOfRetries": number
         }
      ]
  }
...
}
```

Dove criteriaList è un array che specifica l'elenco di criteri che determina il numero di nuovi tentativi consentiti per ogni tipo di errore di un processo.

Dispositivi e processi

I dispositivi possono comunicare con AWS IoT Jobs utilizzando MQTT, HTTP Signature Version 4 o HTTP TLS. Per determinare l'endpoint da utilizzare quando il dispositivo comunica con AWS IoT Jobs, esegui il comando. DescribeEndpoint Ad esempio, se si esegue questo comando:

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

si ottiene un risultato simile al seguente:

```
{
    "endpointAddress": "a1b2c3d4e5f6g7-ats.iot.us-west-2.amazonaws.com"
}
```

Uso del protocollo MQTT

I dispositivi possono comunicare con AWS IoT Jobs utilizzando il protocollo MQTT. I dispositivi si iscrivono agli argomenti MQTT per ricevere notifiche relative a nuovi lavori e ricevere risposte dal servizio AWS IoT Jobs. I dispositivi pubblicano negli argomenti MQTT per eseguire query o aggiornare lo stato del lancio di un processo. Ogni dispositivo ha il proprio argomento MQTT generale. Per ulteriori informazioni sulla pubblicazione e sulla sottoscrizione di argomenti MQTT, consulta Protocolli di dispositivo di comunicazione.

Con questo metodo di comunicazione, il dispositivo utilizza il certificato specifico del dispositivo e la chiave privata per l'autenticazione con Jobs. AWS IoT

I tuoi dispositivi possono sottoscrivere i seguenti argomenti. `thing-name` è il nome dell'oggetto associato al dispositivo.

- **`$aws/things/thing-name/jobs/notify`**

  I dispositivi sottoscrivono questo argomento per ricevere notifiche quando viene aggiunto o rimosso il lancio di un processo nell'elenco dei lanci di processi in sospeso.

- **`$aws/things/thing-name/jobs/notify-next`**

  Effettuare la sottoscrizione a questo argomento per ricevere una notifica quando la successiva esecuzione del processo in sospeso ha subito modifiche.

- **`$aws/things/thing-name/jobs/request-name/accepted`**

  Il servizio AWS IoT Jobs pubblica messaggi di successo e di fallimento su un argomento MQTT. L'argomento viene formato accodando `accepted` o `rejected` all'argomento utilizzato per effettuare la richiesta. `request-name`Ecco il nome di una richiesta come `Get` e l'argomento può essere:. `$aws/things/myThing/jobs/get` AWS IoTJobs pubblica quindi messaggi di successo sull'`$aws/things/myThing/jobs/get/accepted`argomento.

- **`$aws/things/thing-name/jobs/request-name/rejected`**

  Qui, `request-name` è il nome di una richiesta come `Get`. Se la richiesta non è riuscita, AWS IoT Jobs pubblica messaggi di errore sull'`$aws/things/myThing/jobs/get/rejected`argomento.

È inoltre possibile utilizzare le seguenti operazioni API HTTPS:

- Aggiorna lo stato dell'esecuzione di un processo chiamando l'API `UpdateJobExecution`.
- Esegui una query sullo stato dell'esecuzione di un processo chiamando l'API `DescribeJobExecution`.
- Recupera un elenco delle esecuzioni del processo in sospeso chiamando l'API `GetPendingJobExecutions`.
- Recupera l'esecuzione del processo in sospeso successiva chiamando l'API `DescribeJobExecution` con `jobId` come `$next`.
- Ottieni e avvia l'esecuzione del processo in sospeso successiva chiamando l'API `StartNextPendingJobExecution`.

Utilizzo di HTTP Signature Version 4

I dispositivi possono comunicare con AWS IoT Jobs utilizzando la versione 4 della firma HTTP sulla porta 443. Questo è il metodo utilizzato da AWS SDKs and CLI. Per ulteriori informazioni su questi

strumenti, consulta AWS CLICommand Reference: iot-jobs-data or AWSSDKs and Tools e consulta la IotJobsDataPlane sezione relativa alla lingua preferita.

Con questo metodo di comunicazione, il dispositivo utilizza le credenziali IAM per l'autenticazione con AWS IoT Jobs.

Se si usa questo metodo sono disponibili i seguenti comandi:

- DescribeJobExecution

```
aws iot-jobs-data describe-job-execution ...
```

- GetPendingJobExecutions

```
aws iot-jobs-data get-pending-job-executions ...
```

- StartNextPendingJobExecution

```
aws iot-jobs-data start-next-pending-job-execution ...
```

- UpdateJobExecution

```
aws iot-jobs-data update-job-execution ...
```

Uso di HTTP TLS

I dispositivi possono comunicare con AWS IoT Jobs utilizzando HTTP TLS sulla porta 8443 utilizzando un client software di terze parti che supporta questo protocollo.

Con questo metodo, il dispositivo usa l'autenticazione basata su certificato X.509 (ad esempio, usando il proprio certificato specifico del dispositivo e la chiave privata).

Se si usa questo metodo sono disponibili i seguenti comandi:

- DescribeJobExecution
- GetPendingJobExecutions
- StartNextPendingJobExecution
- UpdateJobExecution

Programmazione dei dispositivi per l'uso di Jobs

Gli esempi di questa sezione usano MQTT per illustrare come funziona un dispositivo che utilizza il servizio AWS IoT Jobs. In alternativa, puoi utilizzare i comandi API o CLI corrispondenti. Per questi esempi, supponiamo che un dispositivo chiamato `MyThing` effettuerà la sottoscrizione ai seguenti argomenti MQTT:

- `$aws/things/MyThing/jobs/notify` (o `$aws/things/MyThing/jobs/notify-next`)
- `$aws/things/MyThing/jobs/get/accepted`
- `$aws/things/MyThing/jobs/get/rejected`
- `$aws/things/MyThing/jobs/jobId/get/accepted`
- `$aws/things/MyThing/jobs/jobId/get/rejected`

Se utilizzi la firma tramite codice perAWS IoT, il codice del dispositivo deve verificare la firma del file di codice. La firma si trova nel documento del processo all'interno della proprietà `codesign`. Per ulteriori informazioni sulla verifica della firma del file di codice, consulta la sezione relativa all'esempio di agente del dispositivo.

Argomenti

- Flusso di lavoro dei dispositivi
- Flusso di lavoro dei processi
- Notifiche dei processi

Flusso di lavoro dei dispositivi

Un dispositivo è in grado di gestire i processi eseguiti attraverso uno dei seguenti modi.

- Ottieni il processo successivo

  1. Quando un dispositivo passa online per la prima volta, deve sottoscrivere l'argomento `notify-next` del dispositivo.
  2. Chiama l'API MQTT DescribeJobExecution con jobId `$next` per ottenere il processo successivo, il relativo documento e altri dettagli, inclusi gli stati salvati in `statusDetails`. Se il documento del processo ha una firma del file del codice, è necessario verificare la firma prima di continuare con l'elaborazione della richiesta di processo.

3. Viene chiamata l'API MQTT UpdateJobExecution per aggiornare lo stato del processo. In alternativa, combinando questa fase con la precedente, il dispositivo può chiamare StartNextPendingJobExecution.
4. (Facoltativo) Puoi aggiungere un timer della fase impostando un valore per `stepTimeoutInMinutes` quando richiami UpdateJobExecution o StartNextPendingJobExecution.
5. Vengono eseguite le operazioni specificate dal documento del processo usando l'API MQTT UpdateJobExecution per segnalare lo stato del processo.
6. Continua a monitorare l'esecuzione del processo chiamando l'API MQTT DescribeJobExecution con questo jobId. Se l'esecuzione del processo viene eliminata, DescribeJobExecution restituisce un `ResourceNotFoundException`.

   Se l'esecuzione del processo viene annullata o eliminata mentre il dispositivo sta eseguendo il processo, il dispositivo dovrebbe essere in grado di effettuare il ripristino del processo a uno stato valido.
7. Chiama l'UpdateJobExecutionAPI MQTT al termine del processo per aggiornare lo stato del processo e segnalare l'esito positivo o negativo.
8. Poiché lo stato dell'esecuzione di questo processo è stato modificato in stato terminale, il processo successivo disponibile per l'esecuzione (se presente) cambia. Al dispositivo viene inviata una notifica che segnala che l'esecuzione del processo in sospeso successiva è cambiata. A questo punto, il dispositivo deve continuare come descritto nella fase 2.

Se il dispositivo rimane online, continua a ricevere le notifiche per le successive esecuzioni di processo in sospeso. Ciò include i dati di esecuzione del processo, quando un processo viene completato o viene aggiunta una nuova esecuzione del processo in sospeso. Quando ciò si verifica, il dispositivo continua come descritto nella fase 2.

- Seleziona tra i processi disponibili

  1. Quando un dispositivo passa online per la prima volta, deve sottoscrivere l'argomento `notify` dell'oggetto.
  2. Chiama l'API MQTT GetPendingJobExecutions per ottenere un elenco delle esecuzioni del processo in sospeso.
  3. Se l'elenco contiene una o più esecuzioni, ne viene selezionata una.
  4. Chiama l'API MQTT DescribeJobExecution per ottenere il documento di processo e altri dettagli, inclusi gli stati salvati in `statusDetails`.

5. Viene chiamata l'API MQTT UpdateJobExecution per aggiornare lo stato del processo. Se il campo `includeJobDocument` è impostato su `true` in questo comando, il dispositivo può ignorare la fase precedente e recuperare il documento del processo da questo punto.
6. Se lo desideri, puoi aggiungere un timer della fase impostando un valore per `stepTimeoutInMinutes` quando richiami UpdateJobExecution.
7. Vengono eseguite le operazioni specificate dal documento del processo usando l'API MQTT UpdateJobExecution per segnalare lo stato del processo.
8. Continua a monitorare l'esecuzione del processo chiamando l'API MQTT DescribeJobExecution con questo jobId. Se l'esecuzione del processo viene annullata o eliminata mentre il dispositivo sta eseguendo il processo, il dispositivo dovrebbe essere in grado di effettuare il ripristino a uno stato valido.
9. Chiama l'UpdateJobExecutionAPI MQTT al termine del processo per aggiornare lo stato del processo e per segnalare l'esito positivo o negativo.

   Se il dispositivo rimane online, riceve una notifica di tutte le esecuzioni del processo in sospeso quando una nuova esecuzione in sospeso diventa disponibile. Quando ciò si verifica, il dispositivo può continuare come descritto nella fase 2.

Se il dispositivo non è in grado di eseguire il processo, deve chiamare l'API MQTT UpdateJobExecution per aggiornare lo stato del processo in `REJECTED`.

Flusso di lavoro dei processi

Di seguito vengono illustrati i diversi passaggi del flusso di lavoro dei processi, dall'avvio di un nuovo processo alla segnalazione dello stato di completamento dell'esecuzione di un processo.

Avvio di un nuovo processo

Quando viene creato un nuovo lavoro, AWS IoT Jobs pubblica un messaggio sull'`$aws/things/`*`thing-name`*`/jobs/notify`argomento per ogni dispositivo di destinazione.

Il messaggio contiene le informazioni seguenti:

```
{
    "timestamp":1476214217017,
    "jobs":{
        "QUEUED":[{
```

```
            "jobId":"0001",
            "queuedAt":1476214216981,
            "lastUpdatedAt":1476214216981,
            "versionNumber" : 1
        }]
    }
}
```

Il dispositivo riceve questo messaggio sull'argomento `'$aws/things/thingName/jobs/notify'` quando l'esecuzione del processo viene aggiunta alla coda.

> **Note**
>
> Per i processi con `SchedulingConfig` opzionale, il processo manterrà lo stato iniziale di `SCHEDULED`. Quando il processo raggiunge `startTime` selezionata, si verificherà quanto segue:
>
> - Lo stato del processo verrà aggiornato a `IN_PROGRESS`.
> - Il processo avvierà l'implementazione del documento del processo in tutti i dispositivi nel gruppo di destinazione.

Recupero delle informazioni sul processo

Per ottenere ulteriori informazioni sull'esecuzione di un processo, il dispositivo chiama l'API MQTT DescribeJobExecution con il campo `includeJobDocument` impostato su `true` (impostazione di default).

Se la richiesta ha esito positivo, il servizio AWS IoT Jobs pubblica un messaggio sull'`$aws/things/MyThing/jobs/0023/get/accepted`argomento:

```
{
    "clientToken" : "client-001",
    "timestamp" : 1489097434407,
    "execution" : {
        "approximateSecondsBeforeTimedOut": number,
        "jobId" : "023",
        "status" : "QUEUED",
        "queuedAt" : 1489097374841,
        "lastUpdatedAt" : 1489097374841,
        "versionNumber" : 1,
```

```
        "jobDocument" : {
            < contents of job document >
        }
    }
}
```

Se la richiesta fallisce, il servizio AWS IoT Jobs pubblica un messaggio sull'`$aws/things/MyThing/jobs/0023/get/rejected`argomento.

Il dispositivo a questo punto dispone del documento del processo che può utilizzare per eseguire le operazioni remote per il processo. Se il documento del processo contiene un URL Amazon S3 prefirmato, il dispositivo può usare l'URL per scaricare i file necessari per il processo.

Segnalazione dello stato dell'esecuzione del processo

Quando un dispositivo esegue il processo, può chiamare l'API MQTT UpdateJobExecution per aggiornare lo stato dell'esecuzione del processo.

Ad esempio, un dispositivo può aggiornare lo stato dell'esecuzione di un processo impostandolo su `IN_PROGRESS` pubblicando il messaggio seguente nell'argomento `$aws/things/MyThing/jobs/0023/update`:

```
{
    "status":"IN_PROGRESS",
    "statusDetails": {
        "progress":"50%"
    },
    "expectedVersion":"1",
    "clientToken":"client001"
}
```

Jobs risponde pubblicando un messaggio nell'argomento `$aws/things/MyThing/jobs/0023/update/accepted` o `$aws/things/MyThing/jobs/0023/update/rejected`:

```
{
    "clientToken":"client001",
    "timestamp":1476289222841
}
```

Il dispositivo può combinare le due richieste precedenti chiamando StartNextPendingJobExecution. In questo modo si ottiene e si avvia la successiva esecuzione di processo in sospeso e si consente

al dispositivo di aggiornare lo stato di esecuzione del processo. Questa richiesta, inoltre, restituisce il documento del processo quando c'è un'esecuzione del processo in sospeso.

Se il processo contiene un TimeoutConfig, il timer in corso inizia a funzionare. È inoltre possibile impostare un timer a intervalli per l'esecuzione di un lavoro impostando un valore per `stepTimeoutInMinutes` quando si chiama UpdateJobExecution. Il timer della fase si applica solo all'esecuzione del processo che stai aggiornando. È possibile impostare un nuovo valore per questo timer ogni volta che si aggiorna l'esecuzione di un processo. È inoltre possibile creare uno step timer quando si chiama StartNextPendingJobExecution. Se l'esecuzione del processo resta nello stato `IN_PROGRESS` per un periodo di tempo superiore a quello consentito dall'intervallo del timer della fase, l'esecuzione del processo non va a buon fine e viene impostato lo stato `TIMED_OUT` terminale. Il timer della fase non ha alcun effetto su quello in corso impostato al momento della creazione di un processo.

Il campo `status` può essere impostato su `IN_PROGRESS`, `SUCCEEDED` o `FAILED`. Non è possibile aggiornare lo stato dell'esecuzione di un processo già in stato terminale.

Segnalazione del completamento dell'esecuzione

Quando il dispositivo completa l'esecuzione del processo, chiama l'API MQTT UpdateJobExecution. Se il processo ha avuto esito positivo, impostare `status` su `SUCCEEDED` e, nel payload del messaggio, in `statusDetails`, aggiungere altre informazioni sul processo come coppie nome-valore. Il timer in corso e quello della fase si interrompono al completamento dell'esecuzione del processo.

Esempio:

```
{
    "status":"SUCCEEDED",
    "statusDetails": {
        "progress":"100%"
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

Se il processo ha avuto esito negativo, il valore di `status` viene impostato su `FAILED` e in `statusDetails` vengono aggiunte informazioni sull'errore che si è verificato:

```
{
```

```
    "status":"FAILED",
    "statusDetails": {
        "errorCode":"101",
        "errorMsg":"Unable to install update"
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

> **Note**
>
> L'attributo `statusDetails` può contenere un numero qualsiasi di coppie nome-valore.

Quando il servizio AWS IoT Jobs riceve questo aggiornamento, pubblica un messaggio sull'`$aws/things/MyThing/jobs/notify`argomento per indicare che l'esecuzione del lavoro è completa:

```
{
    "timestamp":1476290692776,
    "jobs":{}
}
```

Processi aggiuntivi

Se ci sono altre esecuzioni dei processi in sospeso per il dispositivo, vengono incluse nel messaggio pubblicato in `$aws/things/MyThing/jobs/notify`.

Esempio:

```
{
    "timestamp":1476290692776,
    "jobs":{
        "QUEUED":[{
            "jobId":"0002",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }],
        "IN_PROGRESS":[{
            "jobId":"0003",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
```

```
        }]
    }
}
```

Notifiche dei processi

Il servizio AWS IoT Jobs pubblica messaggi MQTT su argomenti riservati quando i lavori sono in sospeso o quando cambia la prima esecuzione del lavoro nell'elenco. I dispositivi possono monitorare i processi in sospeso iscrivendosi a questi argomenti.

Tipo di notifica processo

Le notifiche dei processi vengono pubblicate in MQTT come payload JSON. Sono disponibili due tipi di notifiche:

ListNotification

`ListNotification` contiene un elenco di un massimo di 15 esecuzioni del processo in sospeso. Sono ordinate in base allo stato (le esecuzioni dei processi `IN_PROGRESS` hanno la precedenza rispetto alle esecuzioni dei processi `QUEUED`), quindi in base al momento in cui sono state messi in coda.

Una notifica `ListNotification` viene pubblicata ogni volta che viene soddisfatto uno dei seguenti criteri.

- Una nuova esecuzione di un processo è in coda o passa a uno stato non terminale (`IN_PROGRESS` o `QUEUED`).
- Un'esecuzione precedente di un processo passa a uno stato terminale (`FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED` o `REMOVED`).

Per ulteriori informazioni sui limiti con e senza la configurazione di pianificazione, vedere. Limiti delle esecuzioni di Job

NextNotification

- Una `NextNotification` contiene informazioni di riepilogo sull'esecuzione di processo successiva nella coda.

  Una notifica `NextNotification` viene pubblicata ogni volta che la prima esecuzione di un processo nell'elenco cambia.

- Una nuova esecuzione di un processo viene aggiunta all'elenco come `QUEUED` ed è la prima nell'elenco.
- Lo stato dell'esecuzione di un processo che non era la prima nell'elenco passa da `QUEUED` a `IN_PROGRESS` e l'esecuzione diventa la prima nell'elenco. Ciò si verifica quando non sono presenti altre esecuzioni di processi `IN_PROGRESS` nell'elenco o quando l'esecuzione del processo il cui stato passa da `QUEUED` a `IN_PROGRESS` era stata messa in coda prima di ogni altra esecuzione di un processo `IN_PROGRESS` nell'elenco.
- L'esecuzione del processo prima nell'elenco passa a uno stato terminale e viene rimossa dall'elenco.

Per ulteriori informazioni sulla pubblicazione e sulla sottoscrizione di argomenti MQTT, consulta the section called “Protocolli di dispositivo di comunicazione”.

> **Note**
>
> Quando si utilizza HTTP Signature Version 4 o HTTP TLS per comunicare con i processi, le notifiche non sono disponibili.

Processo in sospeso

Il servizio AWS IoT Jobs pubblica un messaggio su un argomento MQTT quando un job viene aggiunto o rimosso dall'elenco delle esecuzioni di job in sospeso per un oggetto o quando la prima esecuzione di job nell'elenco cambia:

- `$aws/things/thingName/jobs/notify`
- `$aws/things/thingName/jobs/notify-next`

I messaggi contengono i payload di esempio seguenti:

`$aws/things/thingName/jobs/notify`:

```
{
  "timestamp" : 10011,
  "jobs" : {
    "IN_PROGRESS" : [ {
      "jobId" : "other-job",
```

```
      "queuedAt" : 10003,
      "lastUpdatedAt" : 10009,
      "executionNumber" : 1,
      "versionNumber" : 1
    } ],
    "QUEUED" : [ {
      "jobId" : "this-job",
      "queuedAt" : 10011,
      "lastUpdatedAt" : 10011,
      "executionNumber" : 1,
      "versionNumber" : 0
    } ]
  }
}
```

Se l'esecuzione del processo denominata `this-job` è stata originata da un processo con la configurazione della pianificazione opzionale selezionata e la distribuzione del documento del processo pianificata per essere eseguita durante una finestra di manutenzione, verrà visualizzata solo durante una finestra di manutenzione ricorrente. Al di fuori di una finestra di manutenzione, il processo denominato `this-job` verrà escluso dall'elenco delle esecuzioni del processo in sospeso, come illustrato nell'esempio seguente.

```
{
  "timestamp" : 10011,
  "jobs" : {
    "IN_PROGRESS" : [ {
      "jobId" : "other-job",
      "queuedAt" : 10003,
      "lastUpdatedAt" : 10009,
      "executionNumber" : 1,
      "versionNumber" : 1
    } ],
    "QUEUED" : []
  }
}
```

`$aws/things/`*`thingName`*`/jobs/notify-next`:

```
{
  "timestamp" : 10011,
  "execution" : {
    "jobId" : "other-job",
```

```
    "status" : "IN_PROGRESS",
    "queuedAt" : 10009,
    "lastUpdatedAt" : 10009,
    "versionNumber" : 1,
    "executionNumber" : 1,
    "jobDocument" : {"c":"d"}
  }
}
```

Se l'esecuzione del processo denominata `other-job` è stata originata da un processo con la configurazione della pianificazione opzionale selezionata e la distribuzione del documento del processo pianificata per essere eseguita durante una finestra di manutenzione, verrà visualizzata solo durante una finestra di manutenzione ricorrente. Al di fuori di una finestra di manutenzione, il processo denominato `other-job` non verrà elencato come l'esecuzione del processo successivo, come illustrato nell'esempio seguente.

```
{} //No other pending jobs
```

```
{
  "timestamp" : 10011,
  "execution" : {
      "jobId" : "this-job",
      "queuedAt" : 10011,
      "lastUpdatedAt" : 10011,
      "executionNumber" : 1,
      "versionNumber" : 0,
      "jobDocument" : {"a":"b"}
  }
} // "this-job" is pending next to "other-job"
```

Possibili valori di stato dell'esecuzione del processo sono `QUEUED`, `IN_PROGRESS`, `FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED` e `REMOVED`.

La serie di esempi riportata di seguito mostra le notifiche che vengono pubblicate in ciascun argomento quando le esecuzioni dei processi vengono create e passano da uno stato all'altro.

Per prima cosa, viene creato un processo chiamato `job1`. Questa notifica viene pubblicata nell'argomento `jobs/notify`:

```
{
```

```
  "timestamp": 1517016948,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

Questa notifica viene pubblicata nell'argomento `jobs/notify-next`:

```
{
  "timestamp": 1517016948,
  "execution": {
    "jobId": "job1",
    "status": "QUEUED",
    "queuedAt": 1517016947,
    "lastUpdatedAt": 1517016947,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

Quando viene creato un altro processo (`job2`), questa notifica viene pubblicata nell'argomento `jobs/notify`:

```
{
  "timestamp": 1517017192,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
```

```
        "versionNumber": 1
      },
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

Una notifica non viene pubblicata nell'argomento `jobs/notify-next` poiché il processo successivo nella coda (`job1`) non è cambiato. Quando inizia l'esecuzione di `job1`, passa allo stato `IN_PROGRESS`. Non vengono pubblicate notifiche perché l'elenco dei processi e il processo successivo in coda non sono cambiati.

Quando viene aggiunto un terzo processo (`job3`), questa notifica viene pubblicata nell'argomento `jobs/notify`:

```
{
  "timestamp": 1517017906,
  "jobs": {
    "IN_PROGRESS": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517017472,
        "startedAt": 1517017472,
        "executionNumber": 1,
        "versionNumber": 2
      }
    ],
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
```

```
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

Una notifica non viene pubblicata nell'argomento `jobs/notify-next` poiché il processo successivo nella coda è ancora `job1`.

Quando `job1` viene completato, passa allo stato `SUCCEEDED` e questa notifica viene pubblicata nell'argomento `jobs/notify`:

```
{
  "timestamp": 1517186269,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

A questo punto, `job1` è stato rimosso dalla coda e il processo successivo da eseguire è `job2`. Questa notifica viene pubblicata nell'argomento `jobs/notify-next`:

```
{
```

```
  "timestamp": 1517186269,
  "execution": {
    "jobId": "job2",
    "status": "QUEUED",
    "queuedAt": 1517017191,
    "lastUpdatedAt": 1517017191,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

Se `job3` deve essere eseguito prima di `job2` (operazione non consigliata), è possibile modificare lo stato di `job3` in `IN_PROGRESS`. In questo caso, `job2` non è più il successivo processo in coda e questa notifica viene pubblicata sull'argomento `jobs/notify-next`:

```
{
  "timestamp": 1517186779,
  "execution": {
    "jobId": "job3",
    "status": "IN_PROGRESS",
    "queuedAt": 1517017905,
    "startedAt": 1517186779,
    "lastUpdatedAt": 1517186779,
    "versionNumber": 2,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

Non vengono pubblicate notifiche nell'argomento `jobs/notify` perché non sono stati aggiunti o rimossi processi.

Se il dispositivo respinge `job2` e aggiorna lo stato in `REJECTED`, questa notifica viene pubblicata sull'argomento `jobs/notify`:

```
{
  "timestamp": 1517189392,
```

```
  "jobs": {
    "IN_PROGRESS": [
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517186779,
        "startedAt": 1517186779,
        "executionNumber": 1,
        "versionNumber": 2
      }
    ]
  }
}
```

Se `job3` (che è ancora in corso) viene eliminato in modo forzato, questa notifica viene pubblicata sull'argomento `jobs/notify`:

```
{
  "timestamp": 1517189551,
  "jobs": {}
}
```

A questo punto, la coda è vuota. Questa notifica viene pubblicata nell'argomento `jobs/notify-next`:

```
{
  "timestamp": 1517189551
}
```

AWS IoT operazioni API per lavori

AWS IoT L'API Jobs può essere utilizzata per una delle seguenti categorie:

- Attività amministrative, come la gestione e il controllo dei processi. Questo è il piano di controllo.
- Dispositivi che eseguono questi processi. Questo è il piano dati che permette di inviare e ricevere dati.

Per la gestione e il controllo dei processi viene usata un'API del protocollo HTTPS. I dispositivi possono usare un'API del protocollo MQTT o HTTPS. L'API del piano di controllo è progettata per

un basso volume di chiamate, come avviene in genere quando si creano e si monitorano i processi. In genere apre una connessione per una singola richiesta e quindi chiude la connessione dopo la ricezione della risposta. L'API HTTPS e MQTT del piano dati consente il polling lungo. Queste operazioni API sono progettate per grandi quantità di traffico, che può essere dimensionato fino a milioni di dispositivi.

Ogni API HTTPS di AWS IoT Jobs ha un comando corrispondente che consente di chiamare l'API da AWS Command Line Interface (AWS CLI). I comandi sono in caratteri minuscoli, con trattini tra le parole che compongono il nome dell'API. È ad esempio possibile richiamare l'API `CreateJob` nell'interfaccia a riga di comando digitando:

```
aws iot create-job ...
```

Se si verifica un errore durante un'operazione, viene fornita una risposta di errore contenente le informazioni sull'errore.

ErrorResponse

Contiene informazioni su un errore che si è verificato durante un'operazione del servizio AWS IoT Jobs.

L'esempio seguente mostra la sintassi di questa operazione:

```
{
    "code": "ErrorCode",
    "message": "string",
    "clientToken": "string",
    "timestamp": timestamp,
    "executionState": JobExecutionState
}
```

Di seguito è riportata una descrizione di `ErrorResponse`:

`code`

ErrorCode può essere impostato su:

InvalidTopic

La richiesta è stata inviata a un argomento nel namespace AWS IoT Jobs che non è mappato a nessuna operazione API.

InvalidJson

Non è stato possibile interpretare i contenuti della richiesta come contenuto JSON valido con codifica UTF-8.

InvalidRequest

I contenuti della richiesta non sono validi. Ad esempio, questo codice viene restituito quando una richiesta `UpdateJobExecution` contiene dettagli sullo stato non validi. Il messaggio contiene dettagli sull'errore.

InvalidStateTransition

Un aggiornamento ha tentato di modificare l'esecuzione del processo in uno stato non valido a causa dello stato attuale dell'esecuzione del processo. Ad esempio, un tentativo di modificare una richiesta dallo stato SUCCEEDED a quello IN_PROGRESS. In questo caso, il corpo del messaggio di errore contiene anche il campo `executionState`.

ResourceNotFound

Il valore di `JobExecution` specificato dall'argomento della richiesta non esiste.

VersionMismatch

La versione prevista specificata nella richiesta non corrisponde alla versione dell'esecuzione del lavoro nel servizio AWS IoT Jobs. In questo caso, il corpo del messaggio di errore contiene anche il campo `executionState`.

InternalError

Si è verificato un errore interno durante l'elaborazione della richiesta.

RequestThrottled

La richiesta è stata sottoposta a throttling.

TerminalStateReached

Si verifica quando viene eseguito un comando per descrivere un processo in un processo che si trova in uno stato terminale.

`message`

Stringa di messaggio di errore.

`clientToken`

Stringa arbitraria usata per mettere in relazione una richiesta con la relativa risposta.

`timestamp`

Tempo, in secondi, dall'epoca (Unix epoch).

`executionState`

Oggetto `JobExecutionState`. Questo campo è incluso solo quando il campo `code` ha il valore `InvalidStateTransition` o `VersionMismatch`. In questi casi, non è necessario eseguire una richiesta `DescribeJobExecution` separata per ottenere i dati sullo stato dell'esecuzione del processo corrente.

Di seguito sono elencate le operazioni API e i tipi di dati Jobs.

- Tipi di dati e API di gestione e controllo dei processi
- Tipi di dati e operazioni API MQTT e HTTPS per i dispositivi per il servizio Jobs

Tipi di dati e API di gestione e controllo dei processi

Per la gestione e il controllo delle applicazioni su CLI e tramite il protocollo HTTPS sono disponibili i comandi seguenti.

- Tipi di dati di gestione e controllo dei processi
- Operazioni API di gestione e controllo dei processi

Per determinare il *`endpoint-url`* parametro per i comandi CLI, esegui questo comando.

```
aws iot describe-endpoint --endpoint-type=iot:Jobs
```

Questo comando restituisce il seguente output.

```
{
"endpointAddress": "account-specific-prefix.jobs.iot.aws-region.amazonaws.com"
}
```

> **Note**
>
> L'endpoint di Jobs non supporta ALPN `x-amzn-http-ca`.

Se utilizzi endpoint dual-stack (IPv6 and IPv6), usa l'endpoint. `iot:Data-ATS` L'endpoint supporta solo. `iot:Jobs` IPv4

Tipi di dati di gestione e controllo dei processi

I seguenti tipi di dati vengono utilizzati dalle applicazioni di gestione e controllo per comunicare con AWS IoT Jobs.

Processo

L'oggetto `Job` contiene i dettagli di un processo. L'esempio seguente mostra la sintassi:

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED",
    "forceCanceled": boolean,
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "comment": "string",
    "targets": ["string"],
    "description": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp,
    "jobProcessDetails": {
        "processingTargets": ["string"],
        "numberOfCanceledThings": long,
        "numberOfSucceededThings": long,
        "numberOfFailedThings": long,
        "numberOfRejectedThings": long,
        "numberOfQueuedThings": long,
        "numberOfInProgressThings": long,
        "numberOfRemovedThings": long,
        "numberOfTimedOutThings": long
    },
    "presignedUrlConfig": {
        "expiresInSec": number,
        "roleArn": "string"
    },
    "jobExecutionsRolloutConfig": {
        "exponentialRate": {
           "baseRatePerMinute": integer,
```

```
            "incrementFactor": integer,
            "rateIncreaseCriteria": {
               "numberOfNotifiedThings": integer, // Set one or the other
               "numberOfSucceededThings": integer // of these two values.
            },
            "maximumPerMinute": integer
      }
    },
    "abortConfig": {
       "criteriaList": [
          {
             "action": "string",
             "failureType": "string",
             "minNumberOfExecutedThings": integer,
             "thresholdPercentage": integer
          }
       ]
    },
    "SchedulingConfig": {
      "startTime": string
      "endTime": string
      "timeZone": string


      "endTimeBehavior": string

   },
    "timeoutConfig": {
        "inProgressTimeoutInMinutes": long
    }
}
```

Per ulteriori informazioni, consulta `Job` o `job`.

JobSummary

L'oggetto `JobSummary` contiene il riepilogo di un processo. L'esempio seguente mostra la sintassi:

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED|SCHEDULED",
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "thingGroupId": "string",
```

```
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp
}
```

Per ulteriori informazioni, consulta `JobSummary` o `job-summary`.

JobExecution

L'oggetto `JobExecution` rappresenta l'esecuzione di un processo in un dispositivo. L'esempio seguente mostra la sintassi:

> **Note**
>
> Quando si utilizzano le operazioni API del piano di controllo (control-plane), il tipo di dati `JobExecution` non contiene un campo `JobDocument`. Per ottenere queste informazioni, puoi utilizzare l'operazione API `GetJobDocument` o il comando della CLI `get-job-document`.

```
{
    "approximateSecondsBeforeTimedOut": 50,
    "executionNumber": 1234567890,
    "forceCanceled": true|false,
    "jobId": "string",
    "lastUpdatedAt": timestamp,
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "forceCanceled": boolean,
    "statusDetails": {
        "detailsMap": {
            "string": "string" ...
        },
        "status": "string"
    },
    "thingArn": "string",
    "versionNumber": 123
}
```

Per ulteriori informazioni, consulta `JobExecution` o `job-execution`.

JobExecutionSummary

L'oggetto `JobExecutionSummary` contiene le informazioni di riepilogo sull'esecuzione del processo. L'esempio seguente mostra la sintassi:

```
{
    "executionNumber": 1234567890,
    "queuedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED"
}
```

Per ulteriori informazioni, consulta `JobExecutionSummary` o `job-execution-summary`.

JobExecutionSummaryForJob

L'oggetto `JobExecutionSummaryForJob` contiene un riepilogo delle informazioni sulle esecuzioni di un determinato processo. L'esempio seguente mostra la sintassi:

```
{
    "executionSummaries": [
        {
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyThing",
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            }
        },
        ...
    ]
}
```

Per ulteriori informazioni, consulta `JobExecutionSummaryForJob` o `job-execution-summary-for-job`.

JobExecutionSummaryForThing

L'`JobExecutionSummaryForThing`oggetto contiene un riepilogo delle informazioni sull'esecuzione di un lavoro su un oggetto specifico. FThe l'esempio seguente mostra la sintassi:

```
{
    "executionSummaries": [
        {
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            },
            "jobId": "MyThingJob"
        },
        ...
    ]
}
```

Per ulteriori informazioni, consulta JobExecutionSummaryForThing o job-execution-summary-for-thing.

Operazioni API di gestione e controllo dei processi

Utilizza le seguenti operazioni API o comandi CLI:

AssociateTargetsWithJob

Associa un gruppo a un processo continuo. Devono essere soddisfatti i criteri seguenti:

- Il processo deve essere stato creato con il campo `targetSelection` impostato su `CONTINUOUS`.
- Lo stato del processo deve essere `IN_PROGRESS`.
- Il numero totale di target associati a un processo non deve essere superiore a 100.

HTTPS request

```
POST /jobs/jobId/targets

{
"targets": [ "string" ],
"comment": "string"
}
```

Per ulteriori informazioni, consulta AssociateTargetsWithJob.

CLI syntax

```
aws iot  associate-targets-with-job \
--targets <value> \
--job-id <value> \
[--comment <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"targets": [
"string"
],
"jobId": "string",
"comment": "string"
}
```

Per ulteriori informazioni, consulta `associate-targets-with-job`.

CancelJob

Annulla un processo.

HTTPS request

```
PUT /jobs/jobId/cancel

{
"force": boolean,
"comment": "string",
"reasonCode": "string"
}
```

Per ulteriori informazioni, consulta `CancelJob`.

CLI syntax

```
aws iot cancel-job \
    --job-id <value> \
    [--force <value>]  \
```

```
    [--comment <value>]  \
    [--reasonCode <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
    "jobId": "string",
    "force": boolean,
    "comment": "string"
}
```

Per ulteriori informazioni, consulta `cancel-job`.

CancelJobExecution

Annulla l'esecuzione di un processo su un dispositivo.

HTTPS request

```
PUT /things/thingName/jobs/jobId/cancel

{
"force": boolean,
"expectedVersion": "string",
"statusDetails": {
    "string": "string"
    ...
}
}
```

Per ulteriori informazioni, consulta `CancelJobExecution`.

CLI syntax

```
aws iot cancel-job-execution \
--job-id <value> \
--thing-name <value> \
[--force | --no-force] \
[--expected-version <value>] \
[--status-details <value>]  \
```

```
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"jobId": "string",
"thingName": "string",
"force": boolean,
"expectedVersion": long,
"statusDetails": {
"string": "string"
}
}
```

Per ulteriori informazioni, consulta `cancel-job-execution`.

CreateJob

Crea un processo. È possibile fornire il documento del processo come collegamento a un file in un bucket Amazon S3 (parametro `documentSource`) oppure nel corpo della richiesta (parametro `document`).

Un processo può essere reso continuo impostando il parametro opzionale `targetSelection` su `CONTINUOUS` (quello di default è `SNAPSHOT`). È possibile utilizzare un processo continuo per eseguire l'onboarding o l'aggiornamento dei dispositivi quando vengono aggiunti a un gruppo, in modo che continui a essere eseguito e venga lanciato sui nuovi dispositivi aggiunti. Ciò può verificarsi anche dopo che gli oggetti presenti nel gruppo al momento della creazione del processo hanno completato il processo.

Un lavoro può avere un valore opzionale TimeoutConfig, che imposta il valore del timer in corso. Il timer in corso non può essere aggiornato e viene applicato a tutte le esecuzioni del processo.

Negli argomenti per l'API `CreateJob` vengono eseguite le convalide seguenti:

- L'`targets`argomento deve essere un elenco di oggetti o gruppi ARNs di oggetti validi. Tutte le cose e i gruppi di cose devono essere nel tuo Account AWS.
- L'argomento `documentSource` deve essere un URL Amazon S3 valido per un documento del processo. Amazon S3 ha URLs la forma:. `https://s3.amazonaws.com/`*`bucketName`*`/`*`objectName`*

- Il documento archiviato nell'URL specificato dall'argomento `documentSource` deve essere un documento JSON con codifica UTF-8.
- Le dimensioni di un documento del processo non possono essere superiori a 32 KB, a causa dei limiti relativi alle dimensioni di un messaggio MQTT (128 KB) e della crittografia.
- `jobId`Devono essere unici nel tuo Account AWS.

HTTPS request

```
PUT /jobs/jobId

{
"targets": [ "string" ],
"document": "string",
"documentSource": "string",
"description": "string",
"jobTemplateArn": "string",
"presignedUrlConfigData": {
    "roleArn": "string",
    "expiresInSec": "integer"
},
"targetSelection": "CONTINUOUS|SNAPSHOT",
"jobExecutionsRolloutConfig": {
    "exponentialRate": {
       "baseRatePerMinute": integer,
       "incrementFactor": integer,
       "rateIncreaseCriteria": {
          "numberOfNotifiedThings": integer, // Set one or the other
          "numberOfSucceededThings": integer // of these two values.
       },
       "maximumPerMinute": integer
  }
},
"abortConfig": {
   "criteriaList": [
      {
         "action": "string",
         "failureType": "string",
         "minNumberOfExecutedThings": integer,
         "thresholdPercentage": integer
      }
   ]
```

```
},
"SchedulingConfig": {
    "startTime": string
    "endTime": string
    "timeZone": string


    "endTimeBehavior": string

   }
"timeoutConfig": {
  "inProgressTimeoutInMinutes": long
}
}
```

Per ulteriori informazioni, consulta `CreateJob`.

CLI syntax

```
aws iot create-job \
    --job-id <value> \
    --targets <value> \
    [--document-source <value>] \
    [--document <value>] \
    [--description <value>] \
    [--job-template-arn <value>] \
    [--presigned-url-config <value>] \
    [--target-selection <value>] \
    [--job-executions-rollout-config <value>] \
    [--abort-config <value>] \
    [--timeout-config <value>] \
    [--document-parameters <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
    "jobId": "string",
    "targets": [ "string" ],
    "documentSource": "string",
    "document": "string",
    "description": "string",
```

```
    "jobTemplateArn": "string",
    "presignedUrlConfig": {
        "roleArn": "string",
        "expiresInSec": long
     },
    "targetSelection": "string",
    "jobExecutionsRolloutConfig": {
          "exponentialRate": {
              "baseRatePerMinute": integer,
              "incrementFactor": integer,
              "rateIncreaseCriteria": {
                 "numberOfNotifiedThings": integer, // Set one or the other
                 "numberOfSucceededThings": integer // of these two values.
              },
      "maximumPerMinute": integer
      }
    },
    "abortConfig": {
    "criteriaList": [
        {
           "action": "string",
           "failureType": "string",
           "minNumberOfExecutedThings": integer,
           "thresholdPercentage": integer
         }
      ]
    },
    "timeoutConfig": {
          "inProgressTimeoutInMinutes": long
    },
    "documentParameters": {
    "string": "string"
    }
}
```

Per ulteriori informazioni, consulta [create-job](#).

DeleteJob

Elimina un processo e le relative esecuzioni.

L'eliminazione di un processo potrebbe richiedere del tempo, a seconda del numero di esecuzioni create per il processo e di altri fattori. Mentre il processo viene eliminato, lo stato del processo viene

indicato come "DELETION_IN_PROGRESS". Il tentativo di eliminare o annullare un processo il cui stato è già "DELETION_IN_PROGRESS" restituirà un errore.

HTTPS request

```
DELETE /jobs/jobId?force=force
```

Per ulteriori informazioni, consulta DeleteJob.

CLI syntax

```
aws iot  delete-job \
--job-id <value> \
[--force | --no-force]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"jobId": "string",
"force": boolean
}
```

Per ulteriori informazioni, consulta delete-job.

DeleteJobExecution

Elimina l'esecuzione di un processo.

HTTPS request

```
DELETE /things/thingName/jobs/jobId/executionNumber/executionNumber?force=force
```

Per ulteriori informazioni, consulta DeleteJobExecution.

CLI syntax

```
aws iot  delete-job-execution \
--job-id <value> \
--thing-name <value> \
```

```
--execution-number <value> \
[--force | --no-force]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"jobId": "string",
"thingName": "string",
"executionNumber": long,
"force": boolean
}
```

Per ulteriori informazioni, consulta `delete-job-execution`.

DescribeJob

Ottiene i dettagli di un'esecuzione del processo.

HTTPS request

```
GET /jobs/jobId
```

Per ulteriori informazioni, consulta `DescribeJob`.

CLI syntax

```
aws iot describe-job \
--job-id <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"jobId": "string"
}
```

Per ulteriori informazioni, consulta `describe-job`.

DescribeJobExecution

Ottiene i dettagli di un'esecuzione del processo. Lo stato dell'esecuzione del processo deve essere `SUCCEEDED` o `FAILED`.

HTTPS request

```
GET /things/thingName/jobs/jobId?executionNumber=executionNumber
```

Per ulteriori informazioni, consulta DescribeJobExecution.

CLI syntax

```
aws iot  describe-job-execution \
--job-id <value> \
--thing-name <value> \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"jobId": "string",
"thingName": "string",
"executionNumber": long
}
```

Per ulteriori informazioni, consulta describe-job-execution.

GetJobDocument

Ottiene il documento per un processo.

> **Note**
>
> I segnaposto non URLs vengono sostituiti con Amazon S3 URLs prefirmato nel documento restituito. I prefirmati URLs vengono generati solo quando il servizio AWS IoT Jobs riceve una richiesta tramite MQTT.

HTTPS request

```
GET /jobs/jobId/job-document
```

Per ulteriori informazioni, consulta GetJobDocument.

CLI syntax

```
aws iot get-job-document \
--job-id <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"jobId": "string"
}
```

Per ulteriori informazioni, consulta get-job-document.

ListJobExecutionsForJob

Ottiene un elenco delle esecuzioni per un processo.

HTTPS request

```
GET /jobs/jobId/things?status=status&maxResults=maxResults&nextToken=nextToken
```

Per ulteriori informazioni, consulta ListJobExecutionsForJob.

CLI syntax

```
aws iot  list-job-executions-for-job \
--job-id <value> \
[--status <value>] \
[--max-results <value>] \
[--next-token <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"jobId": "string",
"status": "string",
"maxResults": "integer",
"nextToken": "string"
}
```

Per ulteriori informazioni, consulta `list-job-executions-for-job`.

ListJobExecutionsForThing

Ottiene un elenco delle esecuzioni di un processo per un oggetto.

HTTPS request

```
GET /things/thingName/jobs?status=status&maxResults=maxResults&nextToken=nextToken
```

Per ulteriori informazioni, consulta `ListJobExecutionsForThing`.

CLI syntax

```
aws iot list-job-executions-for-thing \
--thing-name <value> \
[--status <value>] \
[--max-results <value>] \
[--next-token <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"thingName": "string",
"status": "string",
"maxResults": "integer",
"nextToken": "string"
}
```

Per ulteriori informazioni, consulta `list-job-executions-for-thing`.

ListJobs

Ottiene un elenco di lavori nel tuo. Account AWS

HTTPS request

```
GET /jobs?
status=status&targetSelection=targetSelection&thingGroupName=thingGroupName&thingGroupId=thi
```

Per ulteriori informazioni, consulta `ListJobs`.

CLI syntax

```
aws iot list-jobs \
[--status <value>] \
[--target-selection <value>] \
[--max-results <value>] \
[--next-token <value>] \
[--thing-group-name <value>] \
[--thing-group-id <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"status": "string",
"targetSelection": "string",
"maxResults": "integer",
"nextToken": "string",
"thingGroupName": "string",
"thingGroupId": "string"
}
```

Per ulteriori informazioni, consulta `list-jobs`.

UpdateJob

Aggiorna i campi supportati del processo specificato. I valori aggiornati per `timeoutConfig` diventano effettivi solo per i nuovi avvii in corso. Attualmente, gli avvii in corso continuano a essere lanciati con la configurazione del timeout precedente.

HTTPS request

```
PATCH /jobs/jobId
{
"description": "string",
"presignedUrlConfig": {
  "expiresInSec": number,
  "roleArn": "string"
},
"jobExecutionsRolloutConfig": {
  "exponentialRate": {
     "baseRatePerMinute": number,
     "incrementFactor": number,
     "rateIncreaseCriteria": {
        "numberOfNotifiedThings": number,
        "numberOfSucceededThings": number
     },
  "maximumPerMinute": number
  },
"abortConfig": {
  "criteriaList": [
     {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": number,
        "thresholdPercentage": number
     }
  ]
},
"timeoutConfig": {
  "inProgressTimeoutInMinutes": number
}
}
```

Per ulteriori informazioni, consulta UpdateJob.

CLI syntax

```
aws iot  update-job \
--job-id <value> \
[--description <value>] \
[--presigned-url-config <value>] \
[--job-executions-rollout-config <value>] \
[--abort-config <value>] \
```

```
[--timeout-config <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"description": "string",
"presignedUrlConfig": {
  "expiresInSec": number,
  "roleArn": "string"
},
"jobExecutionsRolloutConfig": {
  "exponentialRate": {
     "baseRatePerMinute": number,
     "incrementFactor": number,
     "rateIncreaseCriteria": {
        "numberOfNotifiedThings": number,
        "numberOfSucceededThings": number
     }
  },
  "maximumPerMinute": number
},
"abortConfig": {
  "criteriaList": [
     {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": number,
        "thresholdPercentage": number
     }
  ]
},
"timeoutConfig": {
  "inProgressTimeoutInMinutes": number
}
}
```

Per ulteriori informazioni, consulta `update-job`.

Tipi di dati e operazioni API MQTT e HTTPS per i dispositivi per il servizio Jobs

I comandi seguenti sono disponibili tramite i protocolli MQTT e HTTPS. Utilizza queste operazioni API sul piano dati per i dispositivi che eseguono i processi.

Tipi di dati MQTT e HTTPS del dispositivo di processi

I seguenti tipi di dati vengono utilizzati per comunicare con il servizio AWS IoT Jobs tramite i protocolli MQTT e HTTPS.

JobExecution

L'oggetto `JobExecution` rappresenta l'esecuzione di un processo in un dispositivo. L'esempio seguente mostra la sintassi:

> **Note**
>
> Quando si utilizzano le operazioni API MQTT e HTTP del piano dati, il tipo di dati `JobExecution` contiene un campo `JobDocument`. I dispositivi possono utilizzare queste informazioni per recuperare il documento di lavoro dall'esecuzione di un processo.

```
{
    "jobId" : "string",
    "thingName" : "string",
    "jobDocument" : "string",
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "statusDetails": {
        "string": "string"
    },
    "queuedAt" : "timestamp",
    "startedAt" : "timestamp",
    "lastUpdatedAt" : "timestamp",
    "versionNumber" : "number",
    "executionNumber": long
}
```

Per ulteriori informazioni, consulta `JobExecution` o `job-execution`.

JobExecutionState

`JobExecutionState` contiene informazioni sullo stato di esecuzione di un processo. L'esempio seguente mostra la sintassi:

```
{
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "statusDetails": {
        "string": "string"
        ...
    }
    "versionNumber": "number"
}
```

Per ulteriori informazioni, consulta `JobExecutionState` o `job-execution-state`.

JobExecutionSummary

Contiene un sottoinsieme delle informazioni sull'esecuzione di un processo. L'esempio seguente mostra la sintassi:

```
{
    "jobId": "string",
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "versionNumber": "number",
    "executionNumber": long
}
```

Per ulteriori informazioni, consulta `JobExecutionSummary` o `job-execution-summary`.

Ulteriori informazioni sulle operazioni API MQTT e HTTPS sono disponibili nelle seguenti sezioni:

- Operazioni API MQTT del dispositivo dei processi
- API HTTP del dispositivo di processi

Operazioni API MQTT del dispositivo dei processi

Puoi inviare i comandi del dispositivo di processi pubblicando i messaggi MQTT negli Argomenti riservati utilizzati per i comandi Jobs.

Il client lato dispositivo deve disporre della sottoscrizione agli argomenti del messaggio di risposta di questi comandi. Se si utilizza il AWS IoT Device Client, il dispositivo si iscriverà automaticamente agli argomenti di risposta. Il che significa che il broker di messaggi pubblica gli argomenti dei messaggi di risposta nel client che ha pubblicato il messaggio di comando, a prescindere che il client abbia effettuato la sottoscrizione agli argomenti del messaggio di risposta. Questi messaggi di risposta non passano attraverso il broker di messaggi e non possono essere sottoscritti da altri client o regole.

Durante la sottoscrizione agli argomenti di processo ed evento `jobExecution` per la tua soluzione di monitoraggio del parco istanze, per prima cosa abilita gli eventi di processo e di esecuzione di processi per ricevere eventi sul lato cloud. Messaggi di avanzamento del processo che vengono elaborati tramite il broker di messaggi e possono essere utilizzati dalle regole di AWS IoT sono pubblicate come Eventi del servizio Jobs. Poiché il broker di messaggi pubblica i messaggi di risposta, anche senza una sottoscrizione esplicita, il client deve essere configurato per ricevere e identificare i messaggi ricevuti. Il client deve inoltre confermare che l'argomento del messaggio *`thingName`* in arrivo si applichi al nome dell'oggetto del client prima che il client agisca sul messaggio.

> **Note**
>
> I messaggi AWS IoT inviati in risposta ai messaggi di comando dell'API MQTT Jobs vengono addebitati sul tuo account, indipendentemente dal fatto che tu li abbia sottoscritti in modo esplicito o meno.

Di seguito vengono illustrate le operazioni API MQTT e la relativa sintassi di richiesta e risposta. Tutte le operazioni API MQTT hanno i seguenti parametri:

clientToken

Un token client opzionale utilizzato per mettere in relazione richieste e risposte. Inserisci un valore arbitrario, che viene riportato nella risposta.

`timestamp`

Periodo di tempo, in secondi, dall'epoca all'invio del messaggio.

GetPendingJobExecutions

Ottiene l'elenco di tutti i processi che non si trovano in uno stato terminale, per un oggetto specifico.

Per richiamare quest'API, pubblica un messaggio in `$aws/things/thingName/jobs/get`.

Payload della richiesta:

```
{ "clientToken": "string" }
```

Il broker dei messaggi pubblicherà `$aws/things/thingName/jobs/get/accepted` e `$aws/things/thingName/jobs/get/rejected` anche senza un abbonamento specifico. Tuttavia, affinché il client riceva i messaggi, deve essere in ascolto. Per ulteriori informazioni, consulta la nota sui messaggi dell'API Jobs.

Payload della risposta:

```
{
"inProgressJobs" : [ JobExecutionSummary ... ],
"queuedJobs" : [ JobExecutionSummary ... ],
"timestamp" : 1489096425069,
"clientToken" : "client-001"
}
```

Dove `inProgressJobs` e `queuedJobs` restituiscono un elenco di oggetti JobExecutionSummary che hanno lo stato `IN_PROGRESS` o `QUEUED`.

StartNextPendingJobExecution

Recupera e avvia la successiva esecuzione del processo in sospeso per un oggetto (stato `IN_PROGRESS` o `QUEUED`).

- Le esecuzioni di un processo con stato `IN_PROGRESS` vengono restituite per prime.
- Le esecuzioni di un processo vengono restituite in base all'ordine di coda. Quando un elemento viene aggiunto o rimosso dal gruppo target per il tuo processo, conferma l'ordine di implementazione di eventuali nuove esecuzioni di processi rispetto alle esecuzioni di processi esistenti.
- Se lo stato della successiva esecuzione del processo in sospeso è `QUEUED`, il relativo stato viene modificato in `IN_PROGRESS` e i dettagli dello stato dell'esecuzione del processo vengono impostati come specificato.
- Se la successiva esecuzione del processo in sospeso è già nello stato `IN_PROGRESS`, i dettagli dello stato non vengono modificati.

- Se non sono presenti esecuzioni in sospeso, la risposta non include il campo `execution`.
- Facoltativamente, puoi creare un timer della fase impostando un valore per la proprietà `stepTimeoutInMinutes`. Se non aggiorni il valore di questa proprietà eseguendo `UpdateJobExecution`, il timeout dell'esecuzione del processo si verifica alla scadenza del timer della fase.

Per richiamare quest'API, pubblica un messaggio in `$aws/things/`*`thingName`*`/jobs/start-next`.

Payload della richiesta:

```
{
"statusDetails": {
    "string": "job-execution-state"
    ...
},
"stepTimeoutInMinutes": long,
"clientToken": "string"
}
```

`statusDetails`

Raccolta di coppie nome-valore che descrivono lo stato dell'esecuzione del processo. Se non è specificato, `statusDetails` resta invariato.

`stepTimeOutInMinutes`

Specifica l'intervallo di tempo a disposizione di ciascun dispositivo per terminare l'esecuzione di questo processo. Se lo stato di esecuzione del processo non è impostato su uno stato terminale prima del timeout o prima della reimpostazione del timer (chiamando `UpdateJobExecution`, impostando lo stato su `IN_PROGRESS` e specificando un nuovo valore di timeout nel campo `stepTimeoutInMinutes`), lo stato di esecuzione del processo viene impostato su `TIMED_OUT`. L'impostazione di questo timeout non ha alcun effetto sul timeout dell'esecuzione del processo, che potresti avere specificato al momento della creazione del processo (`CreateJob` utilizzando il campo `timeoutConfig`).

I valori validi di questo parametro variano da 1 a 10080 (da 1 minuto a 7 giorni). Anche il valore -1 è valido e annullerà lo step timer corrente (creato da un uso precedente di). UpdateJobExecutionRequest

Il broker dei messaggi pubblicherà `$aws/things/thingName/jobs/start-next/accepted` e `$aws/things/thingName/jobs/start-next/rejected` anche senza un abbonamento specifico. Tuttavia, affinché il client riceva i messaggi, deve essere in ascolto. Per ulteriori informazioni, consulta la nota sui messaggi dell'API Jobs.

Payload della risposta:

```
{
"execution" : JobExecutionData,
"timestamp" : timestamp,
"clientToken" : "string"
}
```

Dove `execution` è un oggetto JobExecution. Esempio:

```
{
"execution" : {
    "jobId" : "022",
    "thingName" : "MyThing",
    "jobDocument" : "< contents of job document >",
    "status" : "IN_PROGRESS",
    "queuedAt" : 1489096123309,
    "lastUpdatedAt" : 1489096123309,
    "versionNumber" : 1,
    "executionNumber" : 1234567890
},
"clientToken" : "client-1",
"timestamp" : 1489088524284,
}
```

DescribeJobExecution

Ottiene informazioni dettagliate sull'esecuzione di un processo.

Puoi impostare `jobId` su `$next` per restituire la successiva esecuzione del processo in sospeso per un oggetto (stato `IN_PROGRESS` o `QUEUED`).

Per richiamare quest'API, pubblica un messaggio in `$aws/things/thingName/jobs/jobId/get`.

Payload della richiesta:

```
{
"jobId" : "022",
"thingName" : "MyThing",
"executionNumber": long,
"includeJobDocument": boolean,
"clientToken": "string"
}
```

`thingName`

Nome dell'oggetto associato al dispositivo.

`jobId`

Identificatore univoco assegnato al processo al momento della creazione.

In alternativa, utilizza `$next` per restituire la successiva esecuzione del processo in sospeso per un oggetto (stato `IN_PROGRESS` o `QUEUED`). In questo caso, le esecuzioni del processo con stato `IN_PROGRESS` vengono restituite per prime. Le esecuzioni di un processo vengono restituite in base all'ordine di creazione.

`executionNumber`

(Facoltativo) Numero che identifica l'esecuzione di un processo in un dispositivo. Se non è specificato, viene restituita l'ultima esecuzione del processo.

`includeJobDocument`

(Facoltativo) A meno che non sia impostato su `false`, la risposta contiene il documento del processo. Il valore predefinito è `true`.

Il broker dei messaggi pubblicherà `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/accepted` e `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/rejected` anche senza un abbonamento specifico. Tuttavia, affinché il client riceva i messaggi, deve essere in ascolto. Per ulteriori informazioni, consulta la nota sui messaggi dell'API Jobs.

Payload della risposta:

```
{
"execution" : JobExecutionData,
"timestamp": "timestamp",
"clientToken": "string"
```

```
}
```

Dove `execution` è un oggetto JobExecution.

UpdateJobExecution

Aggiorna lo stato dell'esecuzione di un processo. Se lo desideri, puoi creare un timer della fase impostando un valore per la proprietà `stepTimeoutInMinutes`. Se non aggiorni il valore di questa proprietà eseguendo nuovamente `UpdateJobExecution`, il timeout dell'esecuzione del processo si verifica alla scadenza del timer della fase.

Per richiamare quest'API, pubblica un messaggio in `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update`.

Payload della richiesta:

```
{
"status": "job-execution-state",
"statusDetails": {
    "string": "string"
    ...
},
"expectedVersion": "number",
"executionNumber": long,
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"stepTimeoutInMinutes": long,
"clientToken": "string"
}
```

`status`

Il nuovo stato per l'esecuzione del processo (`IN_PROGRESS`, `FAILED`, `SUCCEEDED` o `REJECTED`). Questo valore deve essere specificato per ogni aggiornamento.

`statusDetails`

Raccolta di coppie nome-valore che descrivono lo stato dell'esecuzione del processo. Se non è specificato, `statusDetails` resta invariato.

`expectedVersion`

Versione corrente prevista dell'esecuzione del processo. Ogni volta che aggiorni l'esecuzione del processo, la versione viene incrementata. Se la versione dell'esecuzione del lavoro

memorizzata nel servizio AWS IoT Jobs non corrisponde, l'aggiornamento viene rifiutato con un `VersionMismatch` errore. Viene restituito anche un file ErrorResponse contenente i dati sullo stato di esecuzione del processo attuale. Questo comportamento rende inutile una richiesta `DescribeJobExecution` separata per ottenere i dati sullo stato dell'esecuzione del processo.

`executionNumber`

(Facoltativo) Numero che identifica l'esecuzione di un processo in un dispositivo. Se non è specificato, viene usata l'ultima esecuzione del processo.

`includeJobExecutionState`

(Facoltativo) Quando è incluso e impostato su `true`, la risposta contiene il campo `JobExecutionState`. Il valore predefinito è `false`.

`includeJobDocument`

(Facoltativo) Quando è incluso e impostato su `true`, la risposta contiene `JobDocument`. Il valore predefinito è `false`.

`stepTimeoutInMinutes`

Specifica l'intervallo di tempo a disposizione di ciascun dispositivo per terminare l'esecuzione di questo processo. Lo stato di esecuzione del processo è impostato su `TIMED_OUT`, a meno che non venga impostato su uno stato terminale prima del timeout o prima che il timer venga reimpostato. L'impostazione o la reimpostazione di questo timeout non ha alcun effetto su quello dell'esecuzione del processo che potresti avere specificato al momento della creazione del processo.

Il broker dei messaggi pubblicherà `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update/accepted` e `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update/rejected` anche senza un abbonamento specifico. Tuttavia, affinché il client riceva i messaggi, deve essere in ascolto. Per ulteriori informazioni, consulta la nota sui messaggi dell'API Jobs.

Payload della risposta:

```
{
"executionState": JobExecutionState,
"jobDocument": "string",
"timestamp": timestamp,
"clientToken": "string"
```

```
}
```

`executionState`

Oggetto JobExecutionState.

`jobDocument`

Un oggetto documento del processo.

> **Note**
>
> Nelle risposte MQTT, il `jobDocument` campo è un oggetto JSON. Nelle risposte HTTP, è una rappresentazione in formato stringa dell'oggetto JSON.

`timestamp`

Periodo di tempo, in secondi, dall'epoca all'invio del messaggio.

`clientToken`

Token client usato per mettere in relazione richieste e risposte.

Quando si utilizza il protocollo MQTT, è anche possibile eseguire i seguenti aggiornamenti:

JobExecutionsChanged

Inviato ogni volta che un'esecuzione di un processo viene aggiunta o rimossa dall'elenco di esecuzioni del processo in sospeso per un oggetto.

Utilizza l'argomento :

`$aws/things/thingName/jobs/notify`

Payload del messaggio:

```
{
"jobs" : {
    "JobExecutionState": [ JobExecutionSummary ... ]
         },
```

```
    "timestamp": timestamp
}
```

NextJobExecutionChanged

Inviato ogni volta che c'è una modifica a cui segue l'esecuzione del processo nell'elenco delle esecuzioni di processi in sospeso per qualcosa, come definito per `DescribeJobExecution` `jobId` con `$next`. Questo messaggio non viene inviato quando cambiano i dettagli della successiva esecuzione del processo, ma solo quando cambia il processo successivo che verrebbe restituito da `DescribeJobExecution` con `jobId` `$next`. Considera le esecuzioni di processo J1 e J2 con stato `QUEUED`. J1 è l'esecuzione successiva nell'elenco di esecuzioni del processo in sospeso. Se lo stato di J2 viene modificato in `IN_PROGRESS`, mentre lo stato di J1 rimane invariato, questa notifica viene inviata e contiene i dettagli di J2.

Utilizza l'argomento :

`$aws/things/`*`thingName`*`/jobs/notify-next`

Payload del messaggio:

```
{
"execution" : JobExecution,
"timestamp": timestamp,
}
```

API HTTP del dispositivo di processi

I dispositivi possono comunicare con AWS IoT Jobs utilizzando la versione 4 della firma HTTP sulla porta 443. Questo è il metodo utilizzato da AWS SDKs and CLI. Per ulteriori informazioni su questi strumenti, vedere AWS CLI Command Reference: iot-jobs-data or AWS SDKs and Tools.

I seguenti comandi sono disponibili per i dispositivi che eseguono i processi. Per informazioni sull'utilizzo delle operazioni API con il protocollo MQTT, consulta Operazioni API MQTT del dispositivo dei processi.

GetPendingJobExecutions

Ottiene l'elenco di tutti i processi che non si trovano in uno stato terminale, per un oggetto specifico.

HTTPS request

```
GET /things/thingName/jobs
```

Risposta:

```
{
"inProgressJobs" : [ JobExecutionSummary ... ],
"queuedJobs" : [ JobExecutionSummary ... ]
}
```

Per ulteriori informazioni, consulta GetPendingJobExecutions.

CLI syntax

```
aws iot-jobs-data get-pending-job-executions \
--thing-name <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"thingName": "string"
}
```

Per ulteriori informazioni, consulta get-pending-job-executions.

StartNextPendingJobExecution

Recupera e avvia la successiva esecuzione del processo in sospeso per un oggetto (con uno stato `IN_PROGRESS` o `QUEUED`).

- Le esecuzioni di un processo con stato `IN_PROGRESS` vengono restituite per prime.
- Le esecuzioni di un processo vengono restituite in base all'ordine di creazione.
- Se lo stato della successiva esecuzione del processo in sospeso è `QUEUED`, il relativo stato viene modificato in `IN_PROGRESS` e i dettagli dello stato dell'esecuzione del processo vengono impostati come specificato.

- Se la successiva esecuzione del processo in sospeso ha già lo stato `IN_PROGRESS`, i dettagli dello stato non vengono modificati.
- Se non sono presenti esecuzioni in sospeso, la risposta non include il campo `execution`.
- Facoltativamente, puoi creare un timer della fase impostando un valore per la proprietà `stepTimeoutInMinutes`. Se non aggiorni il valore di questa proprietà eseguendo `UpdateJobExecution`, il timeout dell'esecuzione del processo si verifica alla scadenza del timer della fase.

HTTPS request

L'esempio seguente mostra la sintassi della richiesta:

```
PUT /things/thingName/jobs/$next
{
"statusDetails": {
    "string": "string"
    ...
},
"stepTimeoutInMinutes": long
}
```

Per ulteriori informazioni, consulta `StartNextPendingJobExecution`.

CLI syntax

Riepilogo:

```
aws iot-jobs-data  start-next-pending-job-execution \
--thing-name <value> \
{--step-timeout-in-minutes <value>] \
[--status-details <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"thingName": "string",
"statusDetails": {
"string": "string"
```

```
},
"stepTimeoutInMinutes": long
}
```

Per ulteriori informazioni, consulta start-next-pending-job-execution.

DescribeJobExecution

Ottiene informazioni dettagliate sull'esecuzione di un processo.

Puoi impostare `jobId` su `$next` per restituire la successiva esecuzione del processo in sospeso per un oggetto. Lo stato dell'esecuzione del processo deve essere `QUEUED` o `IN_PROGRESS`.

HTTPS request

Richiesta:

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument
```

Risposta:

```
{
"execution" : JobExecution,
}
```

Per ulteriori informazioni, consulta DescribeJobExecution.

CLI syntax

Riepilogo:

```
aws iot-jobs-data  describe-job-execution \
--job-id <value> \
--thing-name <value> \
[--include-job-document | --no-include-job-document] \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"jobId": "string",
"thingName": "string",
"includeJobDocument": boolean,
"executionNumber": long
}
```

Per ulteriori informazioni, consulta describe-job-execution.

UpdateJobExecution

Aggiorna lo stato dell'esecuzione di un processo. Facoltativamente, puoi creare un timer della fase impostando un valore per la proprietà `stepTimeoutInMinutes`. Se non aggiorni il valore di questa proprietà eseguendo nuovamente `UpdateJobExecution`, il timeout dell'esecuzione del processo si verifica alla scadenza del timer della fase.

HTTPS request

Richiesta:

```
POST /things/thingName/jobs/jobId
{
"status": "job-execution-state",
"statusDetails": {
    "string": "string"
    ...
},
"expectedVersion": "number",
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"stepTimeoutInMinutes": long,
"executionNumber": long
}
```

Per ulteriori informazioni, consulta UpdateJobExecution.

CLI syntax

Riepilogo:

```
aws iot-jobs-data  update-job-execution \
--job-id <value> \
```

```
--thing-name <value> \
--status <value> \
[--status-details <value>] \
[--expected-version <value>] \
[--include-job-execution-state | --no-include-job-execution-state] \
[--include-job-document | --no-include-job-document] \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--step-timeout-in-minutes <value>] \
[--generate-cli-skeleton]
```

Formato di `cli-input-json`:

```
{
"jobId": "string",
"thingName": "string",
"status": "string",
"statusDetails": {
"string": "string"
},
"stepTimeoutInMinutes": number,
"expectedVersion": long,
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"executionNumber": long
}
```

Per ulteriori informazioni, consulta `update-job-execution`.

Proteggere utenti e dispositivi con Jobs AWS IoT

Per autorizzare gli utenti a utilizzare AWS IoT Jobs con i propri dispositivi, devi concedere loro le autorizzazioni utilizzando le policy IAM. I dispositivi devono quindi essere autorizzati utilizzando AWS IoT Core policy per connettersi in modo sicuro AWS IoT, ricevere esecuzioni di job e aggiornare lo stato di esecuzione.

Tipo di policy richiesto per Jobs AWS IoT

Nella tabella seguente vengono illustrati i diversi tipi di policy da utilizzare per l'autorizzazione. Per ulteriori informazioni sulla policy richiesta da usare, consulta Autorizzazione.

Tipo di policy richiesta

Caso d'uso	Protocollo	Autenticazione	plane/data Piano di controllo	Tipo di identità	Tipo di policy richiesta
Autorizza un amministratore, un operatore o un servizio cloud a lavorare in modo sicuro con Jobs	HTTPS	AWS Autenticazione Signature versione 4 (porta 443)	Piano di controllo e piano dati	Identità Amazon Cognito, IAM o utente federato	Policy IAM
Autorizza il tuo dispositivo IoT a funzionare in modo sicuro con Jobs	MQTT/HTTPS	Autenticazione reciproca TCP o TLS (porta 8883 o 443)	Piano dati	Certificati X.509	AWS IoT Core politica

Per autorizzare le operazioni AWS IoT Jobs che possono essere eseguite sia sul piano di controllo che sul piano dati, è necessario utilizzare le policy IAM. Le identità devono essere state autenticate con AWS IoT per eseguire queste operazioni e devono essere Identità Amazon Cognito o Utenti, gruppi e ruoli IAM. Per ulteriori informazioni sull'autenticazione, consulta Autenticazione.

I dispositivi devono ora essere autorizzati sul piano dati utilizzando AWS IoT Core policy per connettersi in modo sicuro al gateway del dispositivo. Il device gateway consente ai dispositivi di comunicare in modo sicuro AWS IoT, ricevere esecuzioni di job e aggiornare lo stato di esecuzione del lavoro. La comunicazione del dispositivo è protetta utilizzando i protocolli di comunicazione sicuri MQTT o Pubblicazione HTTPS. Questi protocolli utilizzano Certificati client X.509 quelli forniti da AWS IoT per autenticare le connessioni dei dispositivi.

Di seguito viene illustrato come autorizzare gli utenti, i servizi cloud e i dispositivi a utilizzare Jobs. AWS IoT Per informazioni sulle operazioni API del piano di controllo e del piano dati, consulta AWS IoT operazioni API per lavori.

Argomenti

- Autorizzazione di utenti e servizi cloud a utilizzare Jobs AWS IoT
- Autorizzazione dei dispositivi a utilizzare AWS IoT Jobs in modo sicuro sul piano dati

Autorizzazione di utenti e servizi cloud a utilizzare Jobs AWS IoT

Per autorizzare gli utenti e i servizi cloud, è necessario utilizzare le policy IAM sia sul piano di controllo che sul piano dati. Le policy devono essere utilizzate con il protocollo HTTPS e devi usare l'autenticazione AWS Signature Version 4 (porta 443) per autenticare gli utenti.

> **Note**
>
> AWS IoT Core le politiche non devono essere utilizzate sul piano di controllo. Solo le policy IAM vengono utilizzati per autorizzare gli utenti o i servizi cloud. Per ulteriori informazioni sull'uso del tipo di policy richiesta, consulta Tipo di policy richiesto per Jobs AWS IoT.

Le policy IAM sono documenti JSON che contengono le istruzioni delle policy. Le istruzioni delle policy usano gli elementi Effect (Effetto), Action (Operazione) e Resource (Risorsa) per specificare risorse, operazioni consentite o rifiutate e le condizioni in base alle quali le operazioni sono consentite o rifiutate. Per ulteriori informazioni, consulta Documentazione di riferimento degli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

> **Warning**
>
> Ti consigliamo di non utilizzare autorizzazioni jolly, ad esempio nelle policy o `"Action": ["iot:*"]` AWS IoT Core nelle policy IAM. L'utilizzo di autorizzazioni con caratteri jolly non è una best practice consigliata per la sicurezza. Per ulteriori informazioni, consulta AWS IoT policy overly permissive.

Policy IAM sul piano di controllo

Sul piano di controllo, le policy IAM utilizzano il prefisso `iot:` con l'operazione per autorizzare l'operazione API dei processi corrispondente. Ad esempio, l'operazione di policy `iot:CreateJob` concede all'utente l'autorizzazione per utilizzare l'API `CreateJob`.

Operazioni di policy

Nella tabella seguente viene illustrato un elenco delle operazioni di policy criteri IAM e delle autorizzazioni per utilizzare le operazioni API. Per informazioni sui tipi di risorse, vedere Tipi di risorse definiti da. AWS IoT Per ulteriori informazioni sulle AWS IoT azioni, vedere Azioni definite da AWS IoT.

Operazioni di policy IAM sul piano di controllo

Operazione di policy	Operazione API	Tipi di risorse	Description
`iot:AssociateTargetsWithJob`	`AssociateTargetsWithJob`	• job • thing • thinggroup	Rappresenta l'autorizzazione per associare un gruppo a un processo continuo. L'autorizzazione `iot:AssociateTargetsWithJob` viene controllata ogni volta che viene effettuata una richiesta di associare le destinazioni.
`iot:CancelJob`	`CancelJob`	job	Rappresenta l'autorizzazione per annullare un processo. L'autorizzazione `iot:CancelJob` viene controllata ogni volta che viene effettuata una richiesta di annullare un processo.
`iot:CancelJobExecution`	`CancelJobExecution`	• job • thing	Rappresenta l'autorizzazione per aggiornare l'esecuzione di un processo. L'autorizzazione `iot: CancelJobExecution` viene controllata ogni volta che viene effettuata una richiesta di annullare l'esecuzione di un processo.
`iot:CreateJob`	`CreateJob`	• job • thing • thinggroup • jobtemplate	Rappresenta l'autorizzazione per creare un processo. L'autorizzazione `iot:CreateJob` viene controllata ogni volta che viene effettuata una richiesta di creare un processo.

Operazione di policy	Operazione API	Tipi di risorse	Description
		• package	
iot:CreateJobTemplate	CreateJobTemplate	• job • jobtemplate • package	Rappresenta l'autorizzazione per creare un modello di processo. L'autorizzazione iot: CreateJobTemplate viene controllata ogni volta che viene effettuat a una richiesta di creare un modello di processo.
iot:DeleteJob	DeleteJob	job	Rappresenta l'autorizzazione per eliminare un processo. L'autorizzazione iot: DeleteJob viene controllata ogni volta che viene effettuata una richiesta di eliminare un processo.
iot:DeleteJobTemplate	DeleteJobTemplate	jobtemplate	Rappresenta l'autorizzazione per eliminare un modello di processo. L'autorizzazione iot: CreateJobTemplate viene controllata ogni volta che viene effettuat a una richiesta di eliminare un modello di processo.
iot:DeleteJobExecution	DeleteJobTemplate	• job • thing	Rappresenta l'autorizzazione per eliminare l'esecuzione di un processo. L'autoriz zazione iot: DeleteJobExecution viene controllata ogni volta che viene effettuata una richiesta di eliminare l'esecuzione di un processo.
iot:DescribeJob	DescribeJob	job	Rappresenta l'autorizzazione per descriver e un processo. L'autorizzazione iot: DescribeJob viene controllata ogni volta che viene effettuata una richiesta di descrivere un processo.

Operazione di policy	Operazione API	Tipi di risorse	Description
`iot:DescribeJobExecution`	DescribeJobExecution	• job • thing	Rappresenta l'autorizzazione per descrivere l'esecuzione di un processo. L'autorizzazione `iot: DescribeJobExecution` viene controllata ogni volta che viene effettuata una richiesta di descrivere l'esecuzione di un processo.
`iot:DescribeJobTemplate`	DescribeJobTemplate	jobtemplate	Rappresenta l'autorizzazione per descrivere un modello di processo. L'autorizzazione `iot: DescribeJobTemplate` viene controllata ogni volta che viene effettuata una richiesta di descrivere un modello di processo.
`iot:DescribeManagedJobTemplate`	DescribeManagedJobTemplate	jobtemplate	Rappresenta l'autorizzazione per descrivere un modello di processo gestito. L'autorizzazione `iot: DescribeManagedJobTemplate` viene controllata ogni volta che viene effettuata una richiesta di descrivere un modello di processo gestito.
`iot:GetJobDocument`	GetJobDocument	job	Rappresenta l'autorizzazione per ottenere il documento di un processo. L'autorizzazione `iot:GetJobDocument` viene controllata ogni volta che viene effettuata una richiesta di ottenere un documento di processo.
`iot:ListJobExecutionsForJob`	ListJobExecutionsForJob	job	Rappresenta l'autorizzazione per elencare le esecuzioni per un processo. L'autorizzazione `iot:ListJobExecutionsForJob` viene controllata ogni volta che viene effettuata una richiesta di elencare le esecuzioni di un processo.

Operazione di policy	Operazione API	Tipi di risorse	Description
iot:ListJobExecutionsForThing	ListJobExecutionsForThing	thing	Rappresenta l'autorizzazione per elencare le esecuzioni per un processo. L'autorizzazione iot:ListJobExecutionsForThing viene controllata ogni volta che viene effettuata una richiesta di elencare le esecuzioni di processo per un oggetto.
iot:ListJobs	ListJobs	nessuno	Rappresenta l'autorizzazione per elencare i processi. L'autorizzazione iot:ListJobs viene controllata ogni volta che viene effettuata una richiesta di elencare i processi.
iot:ListJobTemplates	ListJobTemplates	nessuno	Rappresenta l'autorizzazione per elencare i modelli di processo. L'autorizzazione iot:ListJobTemplates viene controllata ogni volta che viene effettuata una richiesta di elencare i modelli di processo.
iot:ListManagedJobTemplates	ListManagedJobTemplates	nessuno	Rappresenta l'autorizzazione per elencare i modelli di processo gestiti. L'autorizzazione iot:ListManagedJobTemplates viene controllata ogni volta che viene effettuata una richiesta di elencare i modelli di processo gestiti.
iot:UpdateJob	UpdateJob	job	Rappresenta l'autorizzazione per aggiornare un processo. L'autorizzazione iot:UpdateJob viene controllata ogni volta che viene effettuata una richiesta di aggiornare un processo.

Operazione di policy	Operazione API	Tipi di risorse	Description
`iot:TagResource`	TagResource	• job • jobtemplate • thing	Concede l'autorizzazione per taggare una risorsa specifica.
`iot:UntagResource`	UntagResource	• job • jobtemplate • thing	Concede l'autorizzazione per rimuovere i tag da una risorsa specifica.

Esempio di policy IAM di base

Di seguito viene illustrato un esempio di policy IAM che consente all'utente di eseguire le seguenti operazioni per il gruppo di oggetti e l'oggetto IoT.

Nell'esempio, sostituisci:

- *region*con il tuo Regione AWS, ad esempio`us-east-1`.
- *account-id*con il tuo Account AWS numero, ad esempio`57EXAMPLE833`.
- *thing-group-name*con il nome del gruppo di oggetti IoT a cui stai indirizzando i lavori, ad esempio`FirmwareUpdateGroup`.
- *thing-name*con il nome dell'oggetto IoT a cui stai indirizzando lavori, ad esempio`MyIoTThing`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iot:CreateJobTemplate",
                "iot:CreateJob"
            ],
            "Effect": "Allow",
```

```
            "Resource": "arn:aws:iot:us-east-1:123456789012:thinggroup/thing-group-
name"
        },
        {
            "Action": [
                "iot:DescribeJob",
                "iot:CancelJob",
                "iot:DeleteJob"
            ],
            "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:job/*"
        },
        {
            "Action": [
                "iot:DescribeJobExecution",
                "iot:CancelJobExecution",
                "iot:DeleteJobExecution"
            ],
            "Effect": "Allow",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:thing/thing-123",
                "arn:aws:iot:us-east-1:123456789012:job/*"
            ]
        }
    ]
}
```

Esempio di policy IAM per l'autorizzazione basata su IP

Puoi limitare a specifici indirizzi IP le origini da cui i principali possono effettuare chiamate API all'endpoint del piano di controllo. Per specificare gli indirizzi IP che sono ammessi, nell'elemento Condition della policy IAM, utilizza la chiave di condizione globale `aws:SourceIp`.

L'utilizzo di questa chiave condizionale può anche impedire ad altri Servizio AWS utenti di effettuare queste chiamate API per tuo conto, ad esempio. AWS CloudFormation Per consentire l'accesso a questi servizi, usa la chiave `aws:ViaAWSService`global condition con la SourceIp chiave aws:. In questo modo garantisci che la restrizione di accesso all'indirizzo IP di origine si applichi solo alle richieste effettuate direttamente da un principale. Per maggiori informazioni, vedi AWS: Nega l'accesso a in AWS base all'IP di origine.

L'esempio seguente mostra come consentire che solo un indirizzo IP specifico sia in grado di effettuare chiamate API all'endpoint del piano di controllo. La chiave `aws:ViaAWSService` è impostata su `true`, che consente ad altri servizi di effettuare chiamate API per tuo conto.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:CreateJobTemplate",
                "iot:CreateJob"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                 "IpAddress": {
                     "aws:SourceIp": "123.45.167.89"
                 },
                 "Bool": {
                     "aws:ViaAWSService": "false"
        }
        }
    }]
}
```

Policy IAM sul piano dati

Le policy`iotjobsdata:` IAM sul piano dati utilizzano il prefisso per autorizzare le operazioni API dei processi che gli utenti possono eseguire. Sul piano dati, puoi concedere a un utente l'autorizzazione per utilizzare l'API `DescribeJobExecution` utilizzando l'operazione di policy `iotjobsdata:DescribeJobExecution`.

> **Warning**
>
> L'utilizzo di policy IAM sul piano dati non è consigliato quando si usa AWS IoT Jobs come destinazione per i tuoi dispositivi. Ti consigliamo di usare le policy IAM sul piano di controllo per consentire agli utenti di creare e gestire i processi. Per autorizzare i dispositivi a

recuperare le esecuzioni dei processi e aggiornare lo stato di esecuzione, utilizza AWS IoT Core politiche per il protocollo HTTPS sul piano dati.

Esempio di policy IAM di base

Solitamente eseguirai le operazioni API che devono essere autorizzate utilizzando i comandi della CLI. Di seguito viene riportato l'esempio di un utente che esegue un'operazione `DescribeJobExecution`.

Nell'esempio, sostituisci:

- *`region`*con il tuo Regione AWS, ad esempio`us-east-1`.
- *`account-id`*con il tuo Account AWS numero, ad esempio`57EXAMPLE833`.
- *`thing-name`*con il nome dell'oggetto IoT a cui stai indirizzando lavori, ad esempio`myRegisteredThing`.
- *`job-id`* è l'identificatore univoco per il processo di destinazione dell'API.

```
aws iot-jobs-data describe-job-execution \
    --endpoint-url "https://account-id.jobs.iot.region.amazonaws.com" \
    --job-id jobID --thing-name thing-name
```

Di seguito viene illustrata una policy IAM di esempio che autorizza questa operazione:

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": [
            "iotjobsdata:DescribeJobExecution"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:thing/thing-123"
    }
}
```

Esempi di policy IAM per l'autorizzazione basata su IP

Puoi limitare a specifici indirizzi IP le origini da cui i principali possono effettuare chiamate API all'endpoint del piano dati. Per specificare gli indirizzi IP che sono ammessi, nell'elemento Condition della policy IAM, utilizza la chiave di condizione globale `aws:SourceIp`.

L'utilizzo di questa chiave condizionale può anche impedire ad altri Servizio AWS utenti di effettuare queste chiamate API per tuo conto, ad esempio. AWS CloudFormation Per consentire l'accesso a questi servizi, utilizza la chiave di condizione globale `aws:ViaAWSService` con la chiave di condizione `aws:SourceIp`. In questo modo garantisci che la restrizione di accesso all'indirizzo IP di origine si applichi solo alle richieste effettuate direttamente dal principale. Per ulteriori informazioni, vedere AWS: Nega l'accesso a in AWS base all'IP di origine.

L'esempio seguente mostra come consentire che solo un indirizzo IP specifico sia in grado di effettuare chiamate API all'endpoint del piano dati.

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iotjobsdata:*"
        ],
        "Resource": [
            "*"
        ],
        "Condition": {
            "IpAddress": {
                "aws:SourceIp": "123.45.167.89"
        },
            "Bool": {
                "aws:ViaAWSService": "false"
        }
        }
    }]
}
```

L'esempio seguente mostra come limitare a specifici indirizzi IP o intervalli di indirizzi l'esecuzione di chiamate API all'endpoint del piano dati.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iotjobsdata:*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "123.45.167.89",
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
                }
            },
                "Resource": [
                    "*"
                ]
        }
    ]
}
```

Esempio di policy IAM per il piano di controllo e il piano dati

Se esegui un'operazione API sia sul piano di controllo che sul piano dati, l'operazione della policy del piano di controllo deve utilizzare il prefisso `iot:` e l'operazione della policy del piano dati deve utilizzare il prefisso `iotjobsdata:`.

Ad esempio, l'API `DescribeJobExecution` può essere utilizzata sia nel piano di controllo che nel piano dati. Sul piano di controllo, l'DescribeJobExecutionAPI viene utilizzata per descrivere l'esecuzione di un lavoro. Sul piano dati, l' DescribeJobExecutionAPI viene utilizzata per ottenere dettagli sull'esecuzione di un lavoro.

La seguente policy IAM autorizza l'utente ad utilizzare l'API `DescribeJobExecution` sia sul piano di controllo che sul piano dati.

Nell'esempio, sostituisci:

- *region*con il tuo Regione AWS, ad esempio`us-east-1`.

- *account-id*con il tuo Account AWS numero, ad esempio57EXAMPLE833.
- *thing-name*con il nome dell'oggetto IoT a cui stai indirizzando lavori, ad esempioMyIoTThing.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iotjobsdata:DescribeJobExecution"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thing/thing-123"
        },
        {
            "Action": [
                "iot:DescribeJobExecution",
                "iot:CancelJobExecution",
                "iot:DeleteJobExecution"
            ],
            "Effect": "Allow",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:thing/thing-123",
    "arn:aws:iot:us-east-1:123456789012:job/*"
            ]
        }
    ]
}
```

Autorizzazione dell'assegnazione di tag delle risorse IoT

Per un migliore controllo sui processi e sui modelli di processo che puoi creare, modificare o utilizzare, puoi aggiungere i tag ai processi o ai modelli di processo. I tag ti aiutano anche a distinguere la proprietà e ad assegnare e allocare i costi inserendoli nei gruppi di fatturazione e aggiungendo loro i tag.

Quando un utente desidera taggare i propri lavori o i modelli di lavoro che ha creato utilizzando Console di gestione AWS o il AWS CLI, la tua policy IAM deve concedere all'utente le autorizzazioni per taggarlo. Per concedere le autorizzazioni, la tua policy IAM deve utilizzare l'operazione `iot:TagResource`.

> **Note**
>
> Se la tua policy IAM non include l'`iot:TagResource`azione, qualsiasi `CreateJob`o `CreateJobTemplate`con un tag restituirà un `AccessDeniedException` errore.

Quando desideri etichettare i tuoi lavori o i modelli di lavoro che hai creato utilizzando Console di gestione AWS o il AWS CLI, la tua policy IAM deve concedere l'autorizzazione a taggarli. Per concedere le autorizzazioni, la tua policy IAM deve utilizzare l'operazione `iot:TagResource`.

Per informazioni sull'assegnazione di tag alle risorse, consulta Taggare le tue risorse AWS IoT.

Esempio di policy IAM

Fai riferimento ai seguenti esempi di policy IAM che concedono le autorizzazioni di tagging:

Esempio 1

Un utente che esegue il comando seguente per creare un job e taggarlo in un ambiente specifico.

In questo esempio, sostituisci:

- *`region`*con il tuo Regione AWS, ad esempio`us-east-1`.
- *`account-id`*con il tuo Account AWS numero, ad esempio`57EXAMPLE833`.
- *`thing-name`*con il nome dell'oggetto IoT a cui stai indirizzando lavori, ad esempio`MyIoTThing`.

```
aws iot create-job
    --job-id test_job
    --targets "arn:aws:iot:region:account-id:thing/thingOne"
    --document-source "https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json"
    --description "test job description"
    --tags Key=environment,Value=beta
```

Per questo esempio, devi utilizzare la seguente policy IAM:

```
{
    "Version":"2012-10-17",
    "Statement": {
```

```
        "Action": [
            "iot:CreateJob",
            "iot:CreateJobTemplate",
            "iot:TagResource"
        ],
        "Effect": "Allow",
        "Resource": [
           "arn:aws:iot:us-east-1:123456789012:job/*",
           "arn:aws:iot:us-east-1:123456789012:jobtemplate/*"
        ]
    }
}
```

Autorizzazione dei dispositivi a utilizzare AWS IoT Jobs in modo sicuro sul piano dati

Per autorizzare i dispositivi a interagire in modo sicuro con AWS IoT Jobs sul piano dati, è necessario utilizzare le politiche. AWS IoT Core AWS IoT Core le politiche per le offerte di lavoro sono documenti JSON contenenti dichiarazioni di policy. Queste policy utilizzano anche gli elementi Effect (Effetto), Action (Operazione) e Resource (Risorsa) e seguono una convenzione simile a quella delle policy IAM. Per ulteriori informazioni su questi elementi, consulta Documentazione di riferimento degli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

Le policy possono essere utilizzate con i protocolli MQTT e HTTPS e devono usare l'autenticazione reciproca TCP o TLS per autenticare i dispositivi. Di seguito viene illustrato come utilizzare queste policy con i diversi protocolli di comunicazione.

> **Warning**
>
> Ti consigliamo di non utilizzare autorizzazioni jolly, ad esempio `"Action": ["iot:*"]` nelle tue policy o policy IAM. AWS IoT Core L'utilizzo di autorizzazioni con caratteri jolly non è una best practice consigliata per la sicurezza. Per ulteriori informazioni, consulta AWS IoT policy overly permissive.

AWS IoT Core politiche per il protocollo MQTT

AWS IoT Core le politiche per il protocollo MQTT ti concedono le autorizzazioni per utilizzare le azioni dell'API MQTT del dispositivo jobs. Le operazioni API MQTT vengono utilizzate con argomenti

MQTT riservati ai comandi dei processi. Per ulteriori informazioni su queste operazioni API, consulta Operazioni API MQTT del dispositivo dei processi.

Le policy MQTT utilizzano le operazioni di policy come `iot:Connect`, `iot:Publish`, `iot:Subscribe` e `iot:Receievecon` gli argomenti dei processi. Queste policy ti permettono di connetterti al broker di messaggi, sottoscrivere gli argomenti MQTT dei processi e inviare e ricevere messaggi MQTT tra i dispositivi e il cloud. Per ulteriori informazioni su queste operazioni consulta AWS IoT Core azioni politiche.

Per informazioni sugli argomenti relativi ai AWS IoT lavori, vedere. Argomenti di processo

Esempio di policy MQTT di base

L'esempio seguente mostra come utilizzare `iot:Publish` e `iot:Subscribe` per pubblicare e sottoscrivere processi ed esecuzioni di processo.

Nell'esempio, sostituisci:

- *`region`*con i tuoi Regione AWS, ad esempio`us-east-1`.
- *`account-id`*con il tuo Account AWS numero, ad esempio`57EXAMPLE833`.
- *`thing-name`*con il nome dell'oggetto IoT a cui stai indirizzando lavori, ad esempio`MyIoTThing`.

```
{
    "Version":"2012-10-17",

    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Subscribe"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/$aws/events/job/*",
    "arn:aws:iot:us-east-1:123456789012:topic/$aws/events/jobExecution/*",
    "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thing-123/jobs/*"
            ]
        }
    ]
}
```

AWS IoT Core politiche per il protocollo HTTPS

AWS IoT Core le politiche sul piano dati possono anche utilizzare il protocollo HTTPS con il meccanismo di autenticazione TLS per autorizzare i dispositivi. Le policy utilizzano il prefisso `iotjobsdata:` sul piano dati per autorizzare le operazioni API dei processi che i tuoi dispositivi possono eseguire. Ad esempio, l'operazione di policy `iotjobsdata:DescribeJobExecution` concede all'utente l'autorizzazione per utilizzare l'API `DescribeJobExecution`.

> **Note**
>
> Le operazioni di policy del piano dati devono utilizzare il prefisso `iotjobsdata:`. Sul piano di controllo, le operazioni devono utilizzare il prefisso `iot:`. Per un esempio di policy IAM quando vengono utilizzate le operazioni di policy del piano di controllo e del piano dati, consulta Esempio di policy IAM per il piano di controllo e il piano dati.

Operazioni di policy

La tabella seguente mostra un elenco di azioni AWS IoT Core politiche e autorizzazioni per l'autorizzazione dei dispositivi a utilizzare le azioni API. Per un elenco di operazioni API che puoi eseguire nel piano dati, consultaAPI HTTP del dispositivo di processi.

> **Note**
>
> Le operazioni di policy di esecuzione del processo si applicano solo all'endpoint TLS HTTP. Se utilizzi l'endpoint MQTT devi utilizzare le operazioni di policy MQTT definite in precedenza.

AWS IoT Core azioni politiche sul piano dati

Operazione di policy	Operazione API	Tipi di risorse	Description
`iotjobsdata:DescribeJobExecution`	`DescribeJobExecution`	• job • thing	Rappresenta l'autorizzazione per recuperare l'esecuzione di un processo. L'autorizzazione `iotjobsdata:DescribeJobExecution` viene controllata ogni volta che viene effettuata una

Operazione di policy	Operazione API	Tipi di risorse	Description
			richiesta per recuperare l'esecuzione di un processo.
`iotjobsdata:GetPendingJobExecutions`	`GetPendingJobExecutions`	thing	Rappresenta l'autorizzazione a recuperare l'elenco dei processi che non si trovano in uno stato terminale per un oggetto. L'autorizzazione `iotjobsdata:GetPendingJobExecutions` viene controllata ogni volta che viene effettuata una richiesta di recupero dell'elenco.
`iotjobsdata:StartNextPendingJobExecution`	`StartNextPendingJobExecution`	thing	Rappresenta l'autorizzazione a ottenere e avviare la successiva esecuzione in sospeso di un processo per un oggetto (ossia, per aggiornare l'esecuzione di un processo con stato da `QUEUED` a `IN_PROGRESS`). L'autorizzazione `iotjobsdata:StartNextPendingJobExecution` viene controllata ogni volta che viene effettuata una richiesta di avvio della successiva esecuzione in sospeso di un processo.

Operazione di policy	Operazione API	Tipi di risorse	Description
`iotjobsdata:UpdateJobExecution`	UpdateJobExecution	thing	Rappresenta l'autorizzazione ad aggiornare l'esecuzione di un processo. L'autorizzazione `iotjobsdata:UpdateJobExecution` viene controllata ogni volta che viene effettuata una richiesta di aggiornamento dell'esecuzione di un processo.

Esempio di policy di base

Di seguito viene mostrato un esempio di AWS IoT Core policy che concede l'autorizzazione a eseguire le azioni sulle operazioni dell'API del piano dati per qualsiasi risorsa. Puoi assegnare la policy a una risorsa specifica, ad esempio un oggetto IoT. Nell'esempio, sostituisci:

- *`region`*con i tuoi Regione AWS ad esempio`us-east-1`.
- *`account-id`*con il tuo Account AWS numero, ad esempio`57EXAMPLE833`.
- *`thing-name`*con il nome di IoT, ad esempio`MyIoTthing`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iotjobsdata:GetPendingJobExecutions",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:DescribeJobExecution",
                "iotjobsdata:UpdateJobExecution"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thing/thing-123"
        }
    ]
}
```

È necessario utilizzare queste policy quando i dispositivi IoT utilizzano una policy AWS IoT Core per accedere a una di queste operazioni API, come nel seguente esempio di API `DescribeJobExecution`:

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument&namespaceId=namespaceId
 HTTP/1.1
```

AWS IoT Limiti dei lavori

AWS IoT Jobs prevede quote o limiti di servizio che corrispondono al numero massimo di risorse o operazioni di servizio a tua Account AWS disposizione.

Argomenti

- Limiti delle esecuzioni di Job
- Limiti dei processi attivi e simultanei

Limiti delle esecuzioni di Job

Questa sezione fornisce informazioni sui limiti di esecuzione del lavoro per AWS IoT Device Management.

> **Note**
>
> Questi limiti non fanno parte delle quote di servizio che puoi trovare nella documentazione di AWS IoT Device Management Service Quotas.

Per ottenere informazioni sul numero di esecuzioni di job in sospeso, puoi utilizzare l'`GetPendingJobExecutions`API o iscriverti agli argomenti riservati di MQTT per Jobs e ricevere. AWS IoT Tipo di notifica processo

Il numero di esecuzioni di job in sospeso nel tuo account può variare a seconda che tu abbia abilitato la configurazione di pianificazione e utilizzi una finestra di manutenzione periodica.

Numero massimo di esecuzioni di lavori in sospeso

Nome API/ notifica	Description	Senza pianificazione della configurazione	Con configurazione di pianificazione
`ListNotification`	A `ListNotification` viene pubblicato ogni volta che una vecchia esecuzione di lavoro entra in uno stato terminale o quando una nuova esecuzione di lavoro viene messa in coda o passa a uno stato non terminale. Può visualizzare fino a 15 esecuzioni di job in sospeso che sono o. `QUEUED` `IN_PROGRESS`	10	15 (Fino a 5 esecuzioni di lavoro vengono visualizzate solo `ListNotification` durante una finestra di manutenzione).
`GetPendingJobExecutions`	Quando si richiama l'`GetPendingJobExecutions` API, viene restituito un elenco di esecuzioni di job che non sono ancora iniziate e che possono essere avviate dopo la chiamata API. L'API può restituire fino a un massimo di 10 esecuzioni di job in sospeso. • Delle 10 esecuzioni di processi in sospeso, le esecuzioni corrispondenti `IN_PROGRESS` verranno filtrate dal risultato. • Delle 10 esecuzioni di lavoro in sospeso, se le relative operazioni sono in corso, verranno `SCHEDULED` filtrate dal risultato.	10	15

Limiti dei processi attivi e simultanei

Questa sezione offre ulteriori informazioni sui processi attivi e simultanei e sui limiti correlati.

Processi attivi e limite dei processi attivi

Quando crei un lavoro utilizzando la AWS IoT console o l'`CreateJob`API, lo stato del lavoro cambia in. `IN_PROGRESS` Tutti i processi in corso sono processi attivi e vengono conteggiati rispetto al limite dei processi attivi. Sono inclusi i processi che implementano nuove esecuzioni di processi e i processi in attesa che i dispositivi completino le proprie esecuzioni dei processi. Il limite si applica sia ai processi continui sia ai processi snapshot.

Processi simultanei e limite di simultaneità dei processi

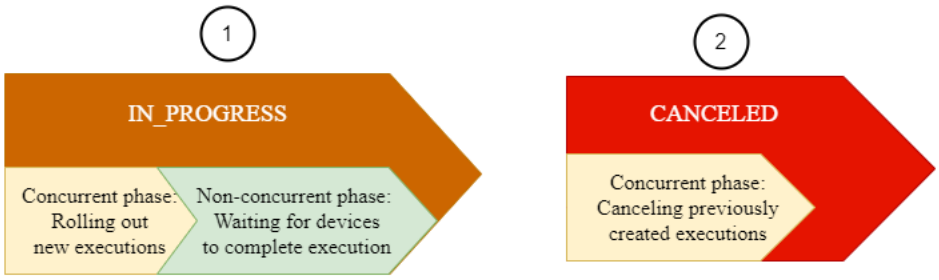
I lavori in corso che stanno implementando nuove esecuzioni di lavori o i lavori che annullano esecuzioni di lavori creati in precedenza sono lavori simultanei e vengono conteggiati ai fini del limite di contemporaneità dei lavori. AWS IoT I processi possono essere implementati e annullati rapidamente a una velocità di 1000 dispositivi al minuto. Ogni processo è `concurrent` e conta ai fini del limite di simultaneità tra processi solo per un breve periodo. Una volta implementate o annullate le esecuzioni del processo, il processo non è più simultaneo e non viene conteggiato rispetto al limite di simultaneità dei processi. Puoi usare la simultaneità dei processi per creare un numero elevato di processi in attesa che i dispositivi completino l'esecuzione del processo.

> **Note**
>
> Se un processo con la configurazione della pianificazione opzionale e distribuzione del documento del processo pianificata durante una finestra di manutenzione raggiunge la `startTime` selezionata e si è raggiunto il limite massimo di simultaneità del processo, il processo pianificato passerà nello stato di `CANCELED`.

Per determinare se un processo è simultaneo, puoi utilizzare la `IsConcurrent` proprietà di un processo dalla AWS IoT console o utilizzando l'API o. `DescribeJob` `ListJob` Il limite si applica sia ai processi continui sia ai processi snapshot.

Per visualizzare i lavori attivi e i limiti di contemporanea dei lavori e le quote di altri AWS IoT lavori per te Account AWS e per richiedere un aumento del limite, consulta Endpoint e quote di AWS IoT Device Management nel. Riferimenti generali di AWS

Il diagramma seguente mostra in che modo la simultaneità di processi si applica ai processi in corso e ai processi in fase di cancellazione.



> **Note**
>
> I nuovi processi con `SchedulingConfig` opzionale manterranno lo stato iniziale `SCHEDULED` e vengono aggiornati a `IN_PROGRESS` una volta raggiunta `startTime` selezionata. Dopo che il nuovo processo con `SchedulingConfig` opzionale raggiunge la `startTime` selezionata e viene aggiornato a `IN_PROGRESS`, verrà conteggiato ai fini del limite di processi attivi e del limite di processi simultanei. I processi con uno stato `SCHEDULED` verranno conteggiati ai fini del limite di processi attivi, ma non verranno conteggiati ai fini del limite di processi simultanei.

La tabella riportata di seguito mostra i limiti che si applicano a processi attivi e simultanei e le fasi di simultaneità e non simultaneità del processo.

Limiti dei processi attivi e simultanei

Stato del Job	Fase	Limite dei processi attivi	Limite di simultaneità dei processi
`SCHEDULED`	Fase non simultanea: AWS IoT Jobs attende che la pianificazione `startTime` del lavoro inizi le notifiche di esecuzione del lavoro sui tuoi dispositivi. I processi in questa fase vengono conteggiati solo rispetto al limite dei processi attivi e per questi processi la proprietà `IsConcurrent` è impostata su false.	Applicato	Non applicato

Stato del Job	Fase	Limite dei processi attivi	Limite di simultaneità dei processi
`IN_PROGRESS`	Fase simultanea: AWS IoT Jobs accetta la richiesta di creazione del lavoro e inizia a distribuire le notifiche di esecuzione del lavoro sui dispositivi. I processi in questa fase sono simultanei, come indicato dalla proprietà `IsConcurrent` impostata su true, e vengono conteggiati rispetto ai limiti dei processi attivi e di simultaneità dei processi.	Applicato	Applicato
	Fase non simultanea: AWS IoT Jobs attende che i dispositivi segnalino i risultati delle loro esecuzioni di lavoro. I processi in questa fase vengono conteggiati solo rispetto al limite dei processi attivi e per questi processi la proprietà `IsConcurrent` è impostata su false.	Applicato	Non applicato
`Canceled`	Fase simultanea: AWS IoT Jobs accetta la richiesta di annullamento del lavoro e inizia ad annullare le esecuzioni di lavori precedentemente create per i tuoi dispositivi. I processi in questa fase sono simultanei e la proprietà `IsConcurrent` è impostata su true. Una volta annullati il processo e le esecuzioni del processo, il processo non è più simultaneo e non viene conteggiato rispetto al limite di simultaneità dei processi.	Non applicato	Applicato

> **Note**
>
> La durata massima di una finestra di manutenzione ricorrente è di 23 ore e 50 minuti.

AWS IoT Device Management comandi

> **Important**
>
> Questa documentazione descrive come utilizzare la funzionalità dei comandi in AWS IoT Device Management. Per informazioni sull'utilizzo di questa funzionalità per AWS IoT FleetWise, consulta Comandi remoti.
> L'utente è l'unico responsabile della distribuzione dei comandi in modo sicuro e conforme alle leggi applicabili. Per ulteriori informazioni sulle tue responsabilità, consulta i Termini di AWS servizio per i servizi. AWS IoT

Usa AWS IoT Device Management i comandi per inviare un'istruzione dal cloud a un dispositivo a cui è connesso AWS IoT. I comandi sono indirizzati a un dispositivo alla volta e possono essere utilizzati per applicazioni a bassa latenza e ad alto rendimento, come il recupero dei log lato dispositivo o l'avvio di una modifica dello stato del dispositivo.

Il comando è una risorsa riutilizzabile gestita da. AWS IoT Device Management Contiene configurazioni che vengono applicate prima di essere pubblicate sul dispositivo. È possibile predefinire un set di comandi per casi d'uso specifici, come l'accensione di una lampadina o lo sblocco della portiera di un veicolo.

La funzionalità AWS IoT Comandi consente di:

- Crea modelli di comandi riutilizzabili con payload statici o dinamici, quindi eseguili su dispositivi specifici.
- Dispositivi di destinazione registrati come AWS IoT oggetti o client MQTT non registrati.
- Esegui più comandi contemporaneamente sullo stesso dispositivo.
- Abilita le notifiche degli eventi e monitora lo stato di esecuzione man mano che i dispositivi elaborano i comandi.

I seguenti argomenti mostrano come creare comandi, inviarli al dispositivo e recuperare lo stato riportato dal dispositivo.

Argomenti

- Avvio rapido

- Comandi, concetti e stato
- Flusso di lavoro dei comandi di alto livello
- Crea e gestisci i comandi
- Avvia e monitora le esecuzioni di comandi
- Deprecare una risorsa di comando

Avvio rapido

Completa questi passaggi per inviare il primo comando:

1. Prerequisiti: assicurati di aver effettuato l'onboarding su IoT Core, un dispositivo connesso a IoT Core, le autorizzazioni IAM per le operazioni dell'API Commands e una libreria client MQTT installata sul tuo dispositivo.
2. Crea un comando: definisci un comando con un payload statico o un modello di payload che specifica le azioni del dispositivo. Per informazioni, consulta Crea una risorsa di comando.
3. Identifica il tuo dispositivo: registra il tuo dispositivo come dispositivo IoT o annota il suo ID client MQTT. Per informazioni, consulta Considerazioni sul dispositivo di destinazione.
4. Configura le autorizzazioni: configura le politiche IAM che consentono al dispositivo di ricevere comandi e pubblicare risposte.
5. Sottoscrizione agli argomenti: configura il dispositivo per sottoscrivere gli argomenti relativi alla richiesta e alla risposta dei comandi. Per informazioni, consulta Scegliete il dispositivo di destinazione per i vostri comandi e iscrivetevi agli argomenti MQTT.
6. Esegui il comando: avvia l'esecuzione del comando sul dispositivo di destinazione. Per informazioni, consulta Avvia l'esecuzione di un comando.

Casi d'uso comuni:

- Diagnostica remota: recupero dei registri, esecuzione della diagnostica, controllo dello stato del dispositivo
- Aggiornamenti della configurazione: modifica delle impostazioni, aggiornamento dei parametri, attivazione delle funzionalità
- Controllo del dispositivo -Lock/unlock, power on/off, modifiche alla modalità

Comandi, concetti e stato

Usa AWS IoT i comandi per inviare istruzioni dal cloud ai dispositivi connessi. Per utilizzare questa funzionalità:

1. Crea un comando con un payload contenente le configurazioni necessarie per l'esecuzione sul dispositivo.
2. Specificate il dispositivo di destinazione che riceverà il payload ed eseguirà le azioni.
3. Esegui il comando sul dispositivo di destinazione e recupera le informazioni sullo stato. Per risolvere i problemi, consulta i log. CloudWatch

Per ulteriori informazioni su questo flusso di lavoro, consulta Flusso di lavoro dei comandi di alto livello.

Argomenti

- Comandi, concetti chiave
- Stati dei comandi
- Stato di esecuzione del comando

Comandi, concetti chiave

I seguenti concetti chiave aiutano a comprendere la funzionalità Comandi. I termini vengono utilizzati in modo coerente in tutta la presente documentazione:

- Comando: un modello riutilizzabile che definisce le istruzioni del dispositivo
- Esecuzione: istanza di un comando in esecuzione su un dispositivo
- Nome dell'oggetto: identificatore per i dispositivi registrati nel registro IoT
- Client ID: identificatore MQTT per dispositivi non registrati
- Payload: i dati delle istruzioni inviati ai dispositivi
- Argomento - Canale MQTT per la comunicazione dei comandi

Comandi

I comandi sono istruzioni inviate dal cloud ai dispositivi IoT come messaggi MQTT. Dopo aver ricevuto il payload, i dispositivi elaborano le istruzioni e intraprendono le azioni corrispondenti,

come la modifica delle impostazioni di configurazione, la trasmissione delle letture dei sensori o il caricamento dei log. I dispositivi restituiscono quindi i risultati al cloud, abilitando il monitoraggio e il controllo remoti.

Spazio dei nomi

Quando crei un comando, specifica il relativo namespace. Per AWS IoT Device Management i comandi, utilizzate lo spazio dei `AWS-IoT` nomi predefinito e fornite un payload o un payloadTemplate. Per AWS IoT FleetWise i comandi, utilizzate lo spazio dei nomi. `AWS-IoT-FleetWise` Per ulteriori informazioni, consulta Comandi remoti nella Guida per gli AWS IoT FleetWise sviluppatori.

Carico utile

Quando create un comando, fornite un payload statico che definisca le azioni che il dispositivo deve eseguire. Il payload può utilizzare qualsiasi formato supportato. Per garantire che i dispositivi interpretino correttamente il payload, consigliamo di specificare il tipo di formato del payload. I dispositivi che utilizzano il MQTT5 protocollo possono seguire lo standard MQTT per identificare il formato. Gli indicatori di formato per JSON o CBOR sono disponibili nell'argomento relativo alla richiesta dei comandi.

Modello di payload

Un modello di payload definisce un payload di comandi con segnaposti che generano payload diversi in fase di esecuzione in base ai valori dei parametri forniti dall'utente. Ad esempio, anziché creare payload separati per valori di temperatura diversi, create un modello con un segnaposto di temperatura e specificate il valore durante l'esecuzione. Ciò elimina il mantenimento di più payload simili.

Dispositivo di destinazione

Per eseguire un comando, specificate un dispositivo di destinazione utilizzando il nome dell'oggetto (per i dispositivi registrati con AWS IoT) o l'ID client MQTT (per i dispositivi non registrati). L'ID client è un identificatore univoco definito nel MQTT protocollo utilizzato per connettere i dispositivi a. AWS IoT Per informazioni dettagliate, vedi Considerazioni sul dispositivo di destinazione.

Argomenti sui comandi

Prima di eseguire un comando, i dispositivi devono sottoscrivere l'argomento relativo alla richiesta dei comandi. Quando si esegue un comando, il payload viene inviato al dispositivo su questo argomento. Dopo l'esecuzione, i dispositivi pubblicano i risultati e lo stato nell'argomento di risposta ai comandi. Per ulteriori informazioni, consulta Argomenti sui comandi.

Esecuzione dei comandi

Un'esecuzione è un'istanza di un comando in esecuzione su un dispositivo di destinazione. Quando si avvia un'esecuzione, il payload viene consegnato al dispositivo e viene generato un ID di esecuzione univoco. Il dispositivo esegue il comando e segnala l'avanzamento a. AWS IoT La logica lato dispositivo determina il comportamento di esecuzione e la segnalazione dello stato su argomenti riservati.

Condizioni relative al valore dei parametri

Quando create comandi con modelli di payload, definite le condizioni di valore per convalidare i valori dei parametri prima dell'esecuzione. Le condizioni di valore garantiscono che i parametri soddisfino i requisiti, impedendo esecuzioni non valide.

Operatori supportati per tipo CommandParameterValue

Tipi numerici (INTEGER, LONG, DOUBLE, UNSIGNEDLONG)

- `EQUALS`- Il valore deve essere uguale al numero specificato
- `NOT_EQUALS`- Il valore non deve essere uguale al numero specificato
- `GREATER_THAN`- Il valore deve essere maggiore del numero specificato
- `GREATER_THAN_EQUALS`- Il valore deve essere maggiore o uguale al numero specificato
- `LESS_THAN`- Il valore deve essere inferiore al numero specificato
- `LESS_THAN_EQUALS`- Il valore deve essere inferiore o uguale al numero specificato
- `IN_RANGE`- Il valore deve rientrare nell'intervallo specificato (incluso)
- `NOT_IN_RANGE`- Il valore deve essere al di fuori dell'intervallo specificato (incluso)
- `IN_SET`- Il valore deve corrispondere a uno dei numeri specificati
- `NOT_IN_SET`- Il valore non deve corrispondere a nessuno dei numeri specificati

Tipo di stringa (STRING)

- `EQUALS`- Il valore deve essere uguale alla stringa specificata
- `NOT_EQUALS`- Il valore non deve essere uguale alla stringa specificata
- `IN_SET`- Il valore deve corrispondere a una delle stringhe specificate
- `NOT_IN_SET`- Il valore non deve corrispondere a nessuna delle stringhe specificate

Tipo booleano

- Le condizioni relative ai valori non sono supportate

Tipo binario

- Le condizioni relative ai valori non sono supportate

Esempio: comando di controllo della temperatura

```
{
  "commandId": "SetTemperature",
  "namespace": "AWS-IoT",
  "payloadTemplate": "{\"temperature\":
 \"${aws:iot:commandexecution::parameter:temperature}\"}",
  "parameters": [
    {
      "name": "temperature",
      "type": "INTEGER",
      "valueConditions": [
        {
          "comparisonOperator": "IN_RANGE",
          "operand": {
            "numberRange": {
              "min": "60",
              "max": "80"
            }
          }
        }
      ]
    }
  ]
}
```

In questo esempio, il `temperature` parametro deve essere compreso tra 60 e 80 (inclusi). Le richieste di esecuzione con valori al di fuori di questo intervallo non vengono convalidate.

> **Note**
>
> Le condizioni di valore vengono valutate al richiamo dell'API. StartCommandExecution Le convalide non riuscite restituiscono un errore e impediscono la creazione dell'esecuzione.

Valore dei parametri, priorità e valutazione.

Quando si avviano le esecuzioni dei comandi con modelli di payload, i valori dei parametri vengono risolti utilizzando la seguente priorità:

1. Parametri della richiesta di esecuzione: i valori forniti nella `StartCommandExecution` richiesta hanno la massima priorità
2. Valori predefiniti del comando: se non viene fornito un parametro nella richiesta di esecuzione, `defaultValue` viene utilizzato quello del parametro
3. Nessun valore: se non viene fornito nessuno dei due, l'esecuzione ha esito negativo come parametro richiesto per generare la richiesta di esecuzione

Le condizioni di valore vengono valutate in base al valore finale del parametro derivato in precedenza, sulla priorità e prima della creazione dell'esecuzione. Se la convalida fallisce, la richiesta di esecuzione restituisce un errore.

Esempio: SetTemperature comando con **`defaultValue`**

```
{
  "parameters": [
    {
      "name": "temperature",
      "type": "INTEGER",
      "defaultValue": {"I": 72},
      "valueConditions": [
        {
          "comparisonOperator": "IN_RANGE",
          "operand": {"numberRange": {"min": "60", "max": "80"}}
        }
      ]
    }
  ]
}
```

All'avvio dell'esecuzione:

- Se si fornisce `"temperature": {"I": 75}` nella richiesta, viene utilizzato 75
- Se si omette il parametro di temperatura, viene utilizzato il valore predefinito 72
- Entrambi i valori vengono convalidati rispetto alla condizione dell'intervallo [60,80]

Stati dei comandi

I comandi Account AWS possono trovarsi in uno dei tre stati seguenti: Disponibile, Obsoleto o In attesa di eliminazione.

Disponibilità

Una volta completata la creazione, un comando si trova nello stato Disponibile e può essere eseguito sui dispositivi.

Deprecated

Contrassegna i comandi come obsoleti quando non sono più necessari. I comandi obsoleti non possono avviare nuove esecuzioni, ma le esecuzioni in sospeso continuano fino al completamento. Per abilitare nuove esecuzioni, ripristina lo stato Disponibile del comando.

In attesa di eliminazione

Quando contrassegni un comando per l'eliminazione, questo viene eliminato automaticamente se è obsoleto per un periodo superiore al timeout massimo (impostazione predefinita: 12 ore). Questa azione è permanente. Se non è obsoleto o obsoleto per un periodo inferiore al timeout, il comando entra nello stato In attesa di eliminazione e viene rimosso dopo la scadenza del timeout.

Stato di esecuzione del comando

Quando si avvia un'esecuzione su un dispositivo di destinazione, il dispositivo entra in `CREATED` stato e può passare ad altri stati in base ai report del dispositivo. È possibile recuperare informazioni sullo stato e tenere traccia delle esecuzioni.

> **Note**
>
> È possibile eseguire più comandi contemporaneamente su un dispositivo. Utilizza il controllo della concorrenza per limitare le esecuzioni per dispositivo e prevenire il sovraccarico. Per il numero massimo di esecuzioni simultanee per dispositivo, consulta le quote dei comandi.AWS IoT Device Management

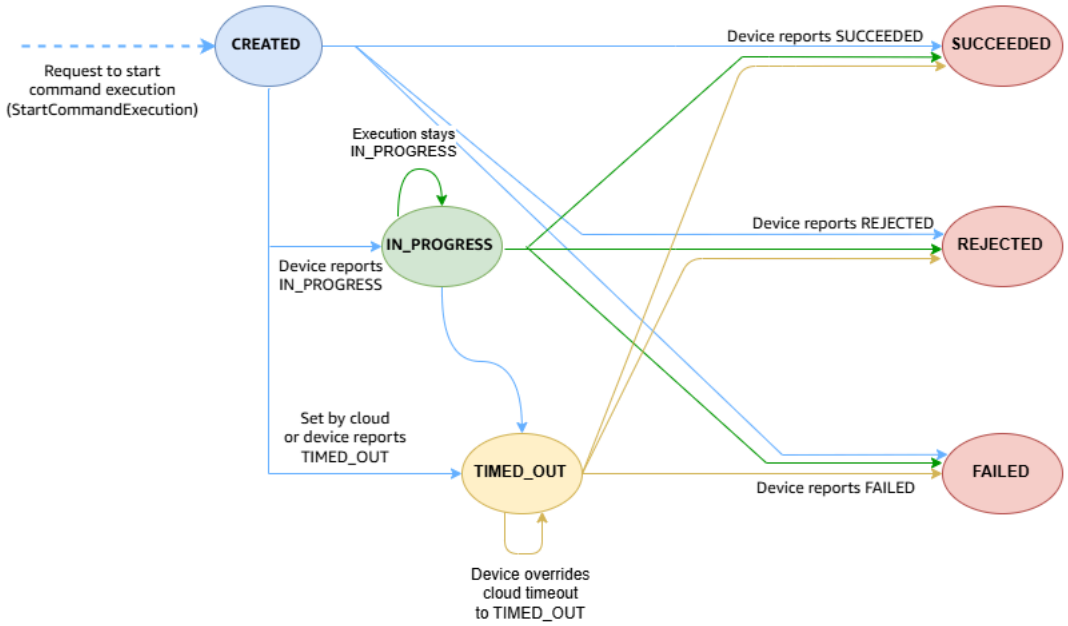
La tabella seguente mostra gli stati di esecuzione e le relative transizioni in base all'avanzamento dell'esecuzione.

Stato e origine dell'esecuzione del comando

Stato di esecuzione del comando	Avviato dal dispositivo/cloud?	Esecuzione del terminale?	Transizioni di stato consentite
`CREATED`	Cloud	No	• IN_PROGRESS • RIUSCITA • NON RIUSCITO • REJECTED • TIMED_OUT
`IN_PROGRESS`	Dispositivo	No	• IN_PROGRESS • RIUSCITA • NON RIUSCITO • REJECTED • TIME_OUT
`TIMED_OUT`	Dispositivo e cloud	No	• RIUSCITA • NON RIUSCITO • REJECTED • TIME_OUT
`SUCCEEDED`	Dispositivo	Sì	Non applicabile
`FAILED`	Dispositivo	Sì	Non applicabile
`REJECTED`	Dispositivo	Sì	Non applicabile

I dispositivi possono pubblicare aggiornamenti sullo stato e sui risultati in qualsiasi momento utilizzando i comandi (argomenti MQTT riservati). Per fornire un contesto aggiuntivo, i dispositivi possono utilizzare `reasonDescription` i campi `reasonCode` e nell'`statusReason`oggetto.

Il diagramma seguente mostra le transizioni dello stato di esecuzione.



> **Note**
>
> Quando non AWS IoT rileva alcuna risposta del dispositivo entro il periodo di timeout, viene impostato `TIMED_OUT` uno stato temporaneo che consente nuovi tentativi e modifiche dello stato. Se il dispositivo segnala esplicitamente`TIMED_OUT`, questo diventa uno stato del terminale senza ulteriori transizioni. Per ulteriori informazioni, consulta Esecuzioni di comandi non terminali.

Le sezioni seguenti descrivono le esecuzioni terminali e non terminali e i relativi stati.

Argomenti

- Esecuzioni di comandi non terminali
- Esecuzioni di comandi da terminale

Esecuzioni di comandi non terminali

Un'esecuzione non è terminale se può accettare aggiornamenti dai dispositivi. Le esecuzioni non terminali sono considerate attive. I seguenti stati non sono terminali:

- created

Quando si avvia un'esecuzione dalla AWS IoT console o si utilizza l'`StartCommandExecution`API, le richieste riuscite modificano lo stato in. `CREATED` Da questo stato, le esecuzioni possono passare a qualsiasi altro stato non terminale o terminale.

- IN_PROGRESS

  Dopo aver ricevuto il payload, i dispositivi possono iniziare a eseguire istruzioni ed eseguire azioni specifiche. Durante l'esecuzione, i dispositivi possono pubblicare risposte all'argomento relativo alla risposta ai comandi e aggiornare lo stato a`IN_PROGRESS`. A partire da`IN_PROGRESS`, le esecuzioni possono passare a qualsiasi stato terminale o non terminale tranne. `CREATED`

  > **Note**
  >
  > L'`UpdateCommandExecution`API può essere richiamata più volte con status. `IN_PROGRESS` Specificate ulteriori dettagli di esecuzione utilizzando l'`statusReason`oggetto.

- TIMED_OUT

  Sia il cloud che il dispositivo possono attivare questo stato. `IN_PROGRESS`Lo stato delle esecuzioni può cambiare in `CREATED` `TIMED_OUT` per i seguenti motivi:

  - Dopo l'invio del comando, viene avviato un timer. Se il dispositivo non risponde entro la durata specificata, il cloud cambia stato in`TIMED_OUT`. In questo caso, l'esecuzione non è terminale.
  - Il dispositivo può sostituire lo stato di qualsiasi terminale o segnalare un timeout e impostare lo stato su. `TIMED_OUT` In questo caso, lo stato rimane`TIMED_OUT`, ma i campi `StatusReason` dell'oggetto cambiano in base alle informazioni sul dispositivo. L'esecuzione diventa terminale.

  Per ulteriori informazioni, consulta Valore del timeout e stato di `TIMED_OUT` esecuzione.

Esecuzioni di comandi da terminale

Un'esecuzione diventa terminale quando non accetta più aggiornamenti dai dispositivi. I seguenti stati sono terminali. Le esecuzioni possono passare allo stato di terminale da qualsiasi stato non terminale:`CREATED`,, o. `IN_PROGRESS` `TIMED_OUT`

- RIUSCITA

Se il dispositivo completa con successo l'esecuzione, può pubblicare una risposta all'argomento relativo alla risposta ai comandi e aggiornare lo stato su. `SUCCEEDED`

- NON RIUSCITO

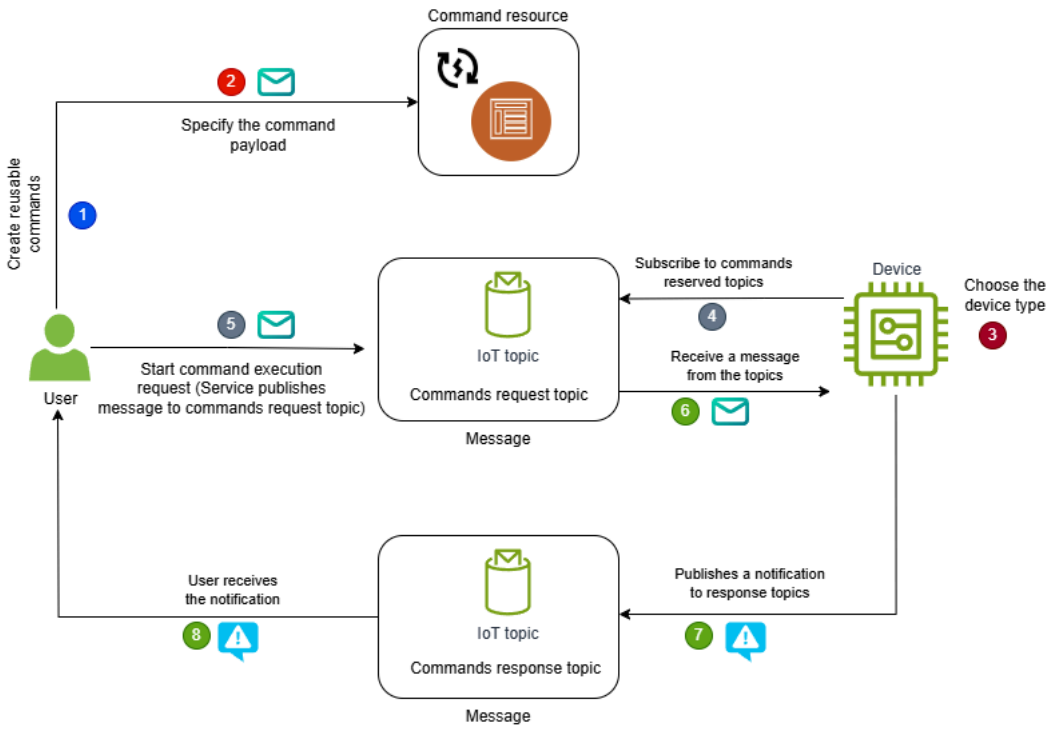
  Quando un dispositivo non riesce a completare l'esecuzione, può pubblicare una risposta all'argomento di risposta ai comandi e aggiornare lo stato a`FAILED`. Utilizzate i `reasonDescription` campi `reasonCode` e nell'`statusReason`oggetto, o nei CloudWatch registri, per risolvere gli errori.

- REJECTED

  Quando un dispositivo riceve una richiesta non valida o incompatibile, può richiamare l'API con status. `UpdateCommandExecution` `REJECTED` Utilizza i `reasonDescription` campi `reasonCode` e nell'`statusReason`oggetto, o nei CloudWatch registri, per risolvere i problemi.

Flusso di lavoro dei comandi di alto livello

Questo flusso di lavoro mostra come i dispositivi interagiscono con AWS IoT Device Management i comandi. Tutte le richieste API HTTP utilizzano le credenziali Sigv4 per la firma.



Panoramica del flusso di lavoro

- Crea e gestisci i comandi
- Scegliete il dispositivo di destinazione per i vostri comandi e iscrivetevi agli argomenti MQTT
- Avvia e monitora l'esecuzione dei comandi per il dispositivo di destinazione
- (Facoltativo) Abilita le notifiche per gli eventi dei comandi

Crea e gestisci i comandi

Per creare e gestire i comandi per i tuoi dispositivi, procedi nel seguente modo.

1. Crea una risorsa di comando

   Crea un comando dal Command Hub o utilizzando l'`CreateCommand`API.
2. Specificare il payload

   Fornisci un payload in qualsiasi formato. Specificate il tipo di contenuto per garantire la corretta interpretazione del dispositivo.

   Per i comandi dinamici con modelli di payload, il payload finale viene generato in fase di esecuzione utilizzando i parametri forniti. I modelli supportano solo il formato JSON, ma il payload generato può essere inviato come JSON o CBOR.
3. (Facoltativo) Gestisci i comandi creati

   Aggiorna il nome visualizzato e la descrizione dopo la creazione. Contrassegna i comandi come obsoleti quando non sono più necessari o eliminali definitivamente. Per modificare le informazioni sul payload, crea un nuovo comando.

Scegliete il dispositivo di destinazione per i vostri comandi e iscrivetevi agli argomenti MQTT

Scegli il tuo dispositivo di destinazione e configura gli argomenti MQTT per ricevere comandi e pubblicare risposte.

1. Scegli il dispositivo di destinazione per il tuo comando

   Scegli un dispositivo di destinazione per ricevere ed eseguire il comando. Usa un nome oggetto per i dispositivi registrati o un ID client per i dispositivi non registrati. Per ulteriori informazioni, consulta Considerazioni sul dispositivo di destinazione.

2. Configura la politica del AWS IoT dispositivo

   Configura una policy IAM che concede le autorizzazioni per ricevere esecuzioni e pubblicare aggiornamenti. Vedi alcuni esempi di Policy IAM di esempio policy.

3. Stabilire una connessione MQTT

   Connect i dispositivi al broker di messaggi e sottoscrivi gli argomenti di richiesta e risposta. I dispositivi richiedono `iot:Connect` l'autorizzazione. Trova l'endpoint del tuo piano dati utilizzando il comando `DescribeEndpoint` API o `describe-endpoint` CLI:

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

   L'esecuzione di questo comando restituisce l'endpoint del piano dati specifico dell'account come illustrato di seguito.

   ```
   account-specific-prefix.iot.region.amazonaws.com
   ```

4. Iscriviti agli argomenti relativi ai comandi

   Sottoscrivi l'argomento relativo alla richiesta dei comandi. Quando si avvia un'esecuzione, il broker di messaggi pubblica il payload relativo a questo argomento. Il dispositivo riceve ed elabora il comando.

   (Facoltativo) Iscriviti agli argomenti di risposta (`acceptedorejected`) per ricevere la conferma se il servizio cloud ha accettato o rifiutato la risposta del dispositivo.

   In questo esempio, sostituisci:

   - *<device>*con `thing` o `client` a seconda che il dispositivo scelto come target sia stato registrato come oggetto IoT o specificato come client MQTT.
   - *<DeviceID>*con l'identificatore univoco del dispositivo bersaglio. Questo ID può essere l'ID univoco del client MQTT o il nome di un oggetto.

   > **Note**
   >
   > Se il tipo di payload non è JSON o CBOR, il *<PayloadFormat>* campo potrebbe non essere presente nell'argomento relativo alla richiesta dei comandi. Per ottenere il formato del payload, si consiglia di utilizzare MQTT5 per ottenere le informazioni sul formato

dalle intestazioni dei messaggi MQTT. Per ulteriori informazioni, consulta Argomenti sui comandi.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
$aws/commands/<devices>/<DeviceID>/executions/+/response/accepted/<PayloadFormat>
$aws/commands/<devices>/<DeviceID>/executions/+/response/rejected/<PayloadFormat>
```

Avvia e monitora l'esecuzione dei comandi per il dispositivo di destinazione

Dopo aver creato i comandi e specificato le destinazioni per il comando, è possibile avviare l'esecuzione sul dispositivo di destinazione effettuando le seguenti operazioni.

1. Avvia l'esecuzione del comando sul dispositivo di destinazione

   Avvia l'esecuzione da Command Hub o utilizza l'`StartCommandExecution`API con l'endpoint specifico del tuo account. Utilizzalo `iot:Data-ATS` per dual-stack (IPv4/) o solo per. IPv6 `iot:Jobs` IPv4

   L'API pubblica il payload nell'argomento di richiesta dei comandi.

   > **Note**
   >
   > Se il dispositivo è offline e utilizza sessioni permanenti MQTT, il comando attende il broker di messaggi. Quando il dispositivo si riconnette prima del timeout, può elaborare il comando e pubblicare i risultati. Se il timeout scade, il timeout di esecuzione scade e il payload verrà scartato.

2. Aggiorna il risultato dell'esecuzione del comando

   Il dispositivo riceve il payload, elabora il comando, esegue le azioni specificate e pubblica i risultati nell'argomento di risposta ai comandi utilizzando l'API basata su `UpdateCommandExecution` MQTT. Se è abbonato agli argomenti accettati e rifiutati, il dispositivo riceve la conferma se il servizio cloud ha accettato o rifiutato la risposta.

   A seconda di ciò che hai specificato nell'argomento della richiesta, *<devices>* possono essere oggetti o client e *<DeviceID>* può essere il nome dell' AWS IoT oggetto o l'ID del client MQTT.

> **Note**
>
> Nell'argomento di risposta ai comandi *<PayloadFormat>* possono essere solo JSON o CBOR.

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/
response/<PayloadFormat>
```

3. (Facoltativo) Recupera il risultato dell'esecuzione del comando

   Recupera i risultati dell'esecuzione dalla AWS IoT console o utilizzando. `GetCommandExecution` Il dispositivo deve pubblicare i risultati nell'argomento di risposta ai comandi per le informazioni più recenti. Visualizza dettagli aggiuntivi tra cui l'ora dell'ultimo aggiornamento, il risultato e l'ora di completamento.

(Facoltativo) Abilita le notifiche per gli eventi dei comandi

Iscriviti agli eventi Commands per ricevere notifiche quando lo stato di esecuzione cambia. Per informazioni dettagliate sugli eventi di esecuzione dei comandi, inclusi il formato dei messaggi di evento e gli attributi del payload, vedereEventi di esecuzione dei comandi.

1. Creazione di una regola dell'argomento

   Iscrivetevi all'argomento Commands events per le notifiche di modifica dello stato. Crea una regola tematica per indirizzare i dati del dispositivo ad altri AWS IoT servizi come AWS Lambda Amazon SQS e AWS Step Functions utilizzando la AWS IoT console o. Creare una AWS IoT regola

   In questo esempio, *<CommandID>* sostituiscilo con l'identificatore del comando per il quale desideri ricevere notifiche e *<CommandExecutionStatus>* con lo stato dell'esecuzione del comando.

```
$aws/events/commandExecution/<CommandID>/<CommandExecutionStatus>
```

> **Note**
>
> Per ricevere notifiche per tutti i comandi e gli stati di esecuzione dei comandi, è possibile utilizzare caratteri jolly e sottoscrivere il seguente argomento.

```
$aws/events/commandExecution/+/#
```

2. Ricevi ed elabora gli eventi dei comandi

   Gestisci i comandi, invia notifiche push e crea applicazioni utilizzando gli eventi sottoscritti.

Il codice seguente mostra un payload di esempio per le notifiche di comandi ed eventi che riceverai.

```
{
    "executionId": "2bd65c51-4cfd-49e4-9310-d5cbfdbc8554",
    "status":"FAILED",
    "statusReason": {
         "reasonCode": "DEVICE_TOO_BUSY",
         "reasonDescription": ""
    },
    "eventType": "COMMAND_EXECUTION",
    "commandArn":"arn:aws:iot:us-east-1:123456789012:command/0b9d9ddf-
e873-43a9-8e2c-9fe004a90086",
    "targetArn":"arn:aws:iot:us-east-1:123456789012:thing/5006c3fc-
de96-4def-8427-7eee36c6f2bd",
    "timestamp":1717708862107
}
```

Crea e gestisci i comandi

Usa AWS IoT Device Management i comandi per configurare azioni remote riutilizzabili o inviare istruzioni immediate ai dispositivi. Crea e gestisci i comandi dalla AWS IoT console o utilizzando. AWS CLI

Crea e gestisci le operazioni dei comandi

- Crea una risorsa di comando
- Recuperare informazioni su un comando

- Elenca i comandi nel tuo Account AWS
- Aggiornare una risorsa di comando
- Deprecare o ripristinare una risorsa di comando
- Eliminare una risorsa di comando

Crea una risorsa di comando

Fornisci le seguenti informazioni durante la creazione di un comando:

- Informazioni generali

  Fornisci un ID di comando univoco per identificare il comando quando lo esegui sui dispositivi di destinazione. Specificate facoltativamente un nome visualizzato, una descrizione e tag per la gestione.

- Carico utile

  Per i comandi statici, fornite un payload che definisca le azioni del dispositivo. Specificate il tipo di formato Payload per una corretta interpretazione del dispositivo.

  Per i comandi dinamici, vedete l'attributo del modello Payload.

- Modello di payload

  Per i comandi dinamici, fornisci un PayloadTemplate con segnaposto e parametri. Fornisci e condizioni opzionali. `defaultValue` AWS IoT Device Management I comandi sostituiscono i segnaposto in fase di esecuzione. I parametri mancanti utilizzano il loro DefaultValue. Tutti i valori devono soddisfare condizioni definite.

  Sono supportati i seguenti tipi di segnaposto con distinzione tra maiuscole e minuscole:

  - `${aws:iot:commandexecution::parameter:`*`parameter1`*`}`— Un segnaposto per il valore di un parametro con il nome. `parameter1`
  - `${aws:iot:commandexecution::executionTimeoutSec}`— Un segnaposto per il parametro di timeout di esecuzione del comando fornito in fase di esecuzione.

Argomenti relativi al payload e ai comandi

Comandi riservati Gli argomenti utilizzano un formato basato sul tipo di formato Payload.

- Per i `application/json` nostri tipi di `application/cbor` contenuto, usa questo argomento della richiesta:

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
```

- Per altri tipi di contenuto o formati non specificati, usa questo argomento della richiesta. Il formato Payload viene visualizzato nell'intestazione del messaggio MQTT.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request
```

L'argomento di risposta ai comandi utilizza `json` o `cbor` formatta indipendentemente dal tipo di payload. *<PayloadFormat>*deve essere `json` o`cbor`:

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/response/<PayloadFormat>
```

Crea una risorsa di comando (console)

Le sezioni seguenti descrivono le considerazioni sul formato Payload e la creazione di comandi dalla console.

Argomenti

- Formato statico del payload dei comandi
- Formato dinamico del payload dei comandi
- Come creare un comando (console)

Formato statico del payload dei comandi

Il payload supporta qualsiasi formato fino a 32 KB. Specificate il tipo di formato Payload per un'interpretazione sicura e corretta del dispositivo.

Specificate il tipo di formato Payload utilizzando il `type/subtype` formato (ad esempio, `application/json` o`application/cbor`). Default: `application/octet-stream`. Vedi Tipi MIME comuni per i formati supportati.

Formato dinamico del payload dei comandi

Il PayloadTemplate deve essere un JSON valido con almeno un segnaposto, fino a 32 KB.

Per il `AwsJsonSubstitution` preprocessore, AWS IoT Device Management Commands invia notifiche in formato JSON o CBOR in base alla configurazione del preprocessore.

Come creare un comando (console)

Per creare un comando dalla console, vai su Command Hub e segui questi passaggi:

1. Scegli il comando Crea.
2. Specificate un ID di comando univoco.
3. (Facoltativo) Specificate il nome visualizzato, la descrizione e i tag.
4. Carica il file Payload contenente le azioni del dispositivo. Specificate il tipo di formato Payload per una corretta interpretazione del dispositivo.
5. (Facoltativo) Per i modelli di payload JSON con segnaposto, i parametri vengono precompilati nella tabella in linea per la modifica.
6. (Facoltativo) Configura il tipo di valore del parametro (obbligatorio), il valore predefinito e le condizioni.
7. Scegliete il comando Crea.

Creare una risorsa di comando (CLI)

Usa il comando `CreateCommand`API o `create-command`CLI per creare un comando.

Argomenti

- Payload del comando
- Policy IAM di esempio
- Esempio di creazione di comandi statici
- Esempio di creazione dinamica di comandi

Payload del comando

Fornisci un modello di payload o payload statico. I payload statici sono codificati in base 64. Per i modelli di payload, il payload finale viene generato in fase di esecuzione utilizzando i valori dei parametri. I dispositivi elaborano il payload ed eseguono azioni specificate. Specificate il tipo di contenuto Payload per la corretta ricezione del dispositivo.

> **Note**
>
> I payload non possono essere modificati dopo la creazione del comando. Crea un nuovo comando per modificare il payload.

Policy IAM di esempio

Prima di utilizzare questa operazione API, assicurati che la tua policy IAM ti autorizzi a eseguire questa azione sul dispositivo. L'esempio seguente mostra una policy IAM che consente l'autorizzazione dell'utente a eseguire l'`CreateCommand`azione.

In questo esempio, sostituisci:

- *`region`*con il tuo Regione AWS, ad esempio*`us-east-1`*.
- *`account-id`*con il tuo Account AWS numero, ad esempio*`123456789012`*.
- *`command-id`*con un identificatore univoco per l'ID del AWS IoT comando, ad esempio*`LockDoor`*. Se desideri inviare più di un comando, puoi specificare questi comandi nella sezione Resource della policy IAM.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": "iot:CreateCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Esempio di creazione di comandi statici

L'esempio seguente mostra come creare un comando statico. A seconda dell'applicazione, sostituisci:

- *`<command-id>`*con un identificatore univoco per il comando. Ad esempio, per bloccare la porta di casa, puoi specificare*`LockDoor`*. Ti consigliamo di utilizzare l'UUID. Puoi anche usare caratteri alfanumerici, «-» e «_».

- (Facoltativo) *<display-name>* e*<description>*, che sono campi facoltativi che è possibile utilizzare per fornire un nome descrittivo e una descrizione significativa per il comando, ad esempio. *Lock the doors of my home*
- `namespace`, che è possibile utilizzare per specificare lo spazio dei nomi del comando. Deve esserlo. `AWS-IoT` Per informazioni sull'utilizzo di questa funzionalità per AWS IoT FleetWise, consulta Comandi remoti
- `payload`contiene informazioni sul payload che si desidera utilizzare durante l'esecuzione del comando e sul tipo di contenuto.

```
aws iot create-command \
    --command-id <command-id> \
    --display-name <display-name> \
    --description <description> \
    --namespace AWS-IoT \
    --payload
 '{"content":"eyAibWVzc2FnZSI6ICJIZWxsbyBJb1QiIH0=","contentType":"application/json"}'
```

L'esecuzione di questo comando genera una risposta che contiene l'ID e l'ARN (Amazon resource name) del comando. Ad esempio, se hai specificato il *LockDoor* comando durante la creazione, di seguito viene mostrato un esempio di output dell'esecuzione del comando.

```
{
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/LockDoor"
}
```

Esempio di creazione dinamica di comandi

L'esempio seguente mostra come creare un comando dinamico. A seconda dell'applicazione, sostituisci:

- *<command-id>*con un identificatore univoco per il comando. Ad esempio, per impostare lo stato di alimentazione della luce, è possibile specificare*Light_Power_Status*. Ti consigliamo di utilizzare l'UUID. Puoi anche usare caratteri alfanumerici, «-» e «_».
- (Facoltativo) *<display-name>* e*<description>*, che sono campi facoltativi che è possibile utilizzare per fornire un nome descrittivo e una descrizione significativa per il comando, ad esempio. *Turn a light ON or OFF*

- `namespace`, che è possibile utilizzare per specificare lo spazio dei nomi del comando. Deve esserlo. `AWS-IoT` Per informazioni sull'utilizzo di questa funzionalità per AWS IoT FleetWise, consulta Comandi remoti
- `payloadTemplate`contiene il modello di playload in formato JSON con segnaposto.
- `preprocessor`contiene la configurazione per il preprocessore che determina come il PayloadTemplate deve essere elaborato.
- `mandatoryParameter`contiene i parametri corrispondenti ai segnaposto nel PayloadTemplate, il loro tipo, i valori predefiniti e le condizioni.

```
aws iot create-command \
    --command-id Light_Power_Status \
    --description "Turn a light ON or OFF" \
    --namespace AWS-IoT \
    --payload-
template '{"powerStatus":"${aws:iot:commandexecution::parameter:powerStatus}"}' \
    --preprocessor awsJsonSubstitution={outputFormat=JSON} \
    --mandatory-parameters "name=powerStatus, defaultValue={S=OFF},
 valueConditions=[{comparisonOperator=IN_SET, operand={strings=['ON','OFF']}}]"
```

L'esecuzione di questo comando genera una risposta che contiene l'ID e l'ARN (Amazon resource name) del comando. Ad esempio, se hai specificato il *`Light_Power_Status`* comando durante la creazione, di seguito viene mostrato un esempio di output dell'esecuzione del comando.

```
{
    "commandId": "Light_Power_Status",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/Light_Power_Status"
}
```

Recuperare informazioni su un comando

Dopo aver creato un comando, è possibile recuperare informazioni su di esso dalla AWS IoT console e utilizzare il. AWS CLIÈ possibile ottenere le seguenti informazioni.

- L'ID del comando, il nome della risorsa Amazon (ARN), qualsiasi nome visualizzato e descrizione che hai specificato per il comando.
- Lo stato del comando, che indica se un comando è disponibile per l'esecuzione sul dispositivo di destinazione o se è obsoleto o eliminato.

- Il payload o il payloadTemplate che hai fornito.
- Il preprocessore che hai fornito.
- I parametri obbligatori che hai fornito.
- L'ora in cui il comando è stato creato e aggiornato l'ultima volta.

Recupera una risorsa di comando (console)

Per recuperare un comando dalla console, vai al Command Hub della AWS IoT console, quindi scegli il comando che hai creato per visualizzarne i dettagli.

Oltre ai dettagli del comando, puoi visualizzare la cronologia dei comandi, che fornisce informazioni sulle esecuzioni del comando sul dispositivo di destinazione. Dopo aver eseguito questo comando sul dispositivo, puoi trovare informazioni sulle esecuzioni in questa scheda.

Recupera una risorsa di comando (CLI)

Utilizzate l'operazione API del piano di controllo `GetCommand`HTTP o il `get-command` AWS CLI comando per recuperare informazioni su una risorsa di comando. Devi aver già creato il comando utilizzando la richiesta `CreateCommand` API o la `create-command` CLI.

Policy IAM di esempio

Prima di utilizzare questa operazione API, assicurati che la tua policy IAM ti autorizzi a eseguire questa azione sul dispositivo. L'esempio seguente mostra una policy IAM che consente l'autorizzazione dell'utente a eseguire l'`GetCommand`azione.

In questo esempio, sostituisci:

- *`region`*con il tuo Regione AWS, ad esempio`us-east-1`.
- *`account-id`*con il tuo Account AWS numero, ad esempio*`123456789023`*.
- *`command-id`*con il tuo identificatore di comando AWS IoT univoco, ad esempio *`LockDoor`* o*`Light_Power_Status`*. Se desideri recuperare più di un comando, puoi specificare questi comandi nella sezione Resource della policy IAM.

```
{
    "Version":"2012-10-17",
```

```
    "Statement": {
        "Action": "iot:GetCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Recupera un esempio di comando ()AWS CLI

L'esempio seguente mostra come recuperare informazioni su un comando utilizzando il. `get-command` AWS CLI A seconda dell'applicazione, *<command-id>* sostituiscilo con l'identificatore del comando per il quale desideri recuperare le informazioni. È possibile ottenere queste informazioni dalla risposta della `create-command` CLI.

```
aws iot get-command --command-id <command-id>
```

L'esecuzione di questo comando genera una risposta che contiene informazioni sul comando, sul payload e sull'ora in cui è stato creato e aggiornato l'ultima volta. Fornisce inoltre informazioni che indicano se un comando è obsoleto o viene eliminato.

Ad esempio, il codice seguente mostra una risposta di esempio.

```
{
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:<region>:<account>:command/LockDoor",
    "namespace": "AWS-IoT",
    "payload":{
        "content": "eyAibWVzc2FnZSI6ICJIZWxsbyBJb1QiIH0=",
        "contentType": "application/json"
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "lastUpdatedAt": "2024-03-23T00:50:10.095000-07:00",
    "deprecated": false,
    "pendingDeletion": false
}
```

Elenca i comandi nel tuo Account AWS

Dopo aver creato i comandi, puoi visualizzare i comandi che hai creato nel tuo account. Nell'elenco puoi trovare informazioni su:

- L'ID del comando e l'eventuale nome visualizzato specificato per i comandi.
- Il nome della risorsa Amazon (ARN) dei comandi.
- Lo stato del comando che indica se i comandi sono disponibili per l'esecuzione sul dispositivo di destinazione o se sono obsoleti.

> **Note**
>
> L'elenco non mostra quelli che vengono eliminati dal tuo account. Se i comandi sono in attesa di eliminazione, puoi comunque visualizzarne i dettagli utilizzando il relativo ID di comando.

- L'ora in cui i comandi sono stati creati e aggiornati l'ultima volta.

Elenca i comandi nel tuo account (console)

Nella AWS IoT console, puoi trovare l'elenco dei comandi che hai creato e i relativi dettagli accedendo al Command Hub.

Elenca i comandi nel tuo account (CLI)

Per elencare i comandi che hai creato, utilizza l'operazione `ListCommands`API o la `list-commands`CLI.

Policy IAM di esempio

Prima di utilizzare questa operazione API, assicurati che la tua policy IAM ti autorizzi a eseguire questa azione sul dispositivo. L'esempio seguente mostra una policy IAM che consente l'autorizzazione dell'utente a eseguire l'`ListCommands`azione.

In questo esempio, sostituisci:

- *`region`*con il tuo Regione AWS, ad esempio`us-east-1`.
- *`account-id`*con il tuo Account AWS numero, ad esempio*`123456789012`*.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": "iot:ListCommands",
```

```
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/*"
    }
}
```

Elenca i comandi nell'esempio del tuo account

Il comando seguente mostra come elencare i comandi nel tuo account.

```
aws iot list-commands --namespace "AWS-IoT"
```

L'esecuzione di questo comando genera una risposta che contiene un elenco di comandi creati dall'utente, l'ora in cui i comandi sono stati creati e l'ultimo aggiornamento. Fornisce inoltre informazioni sullo stato del comando, che indicano se un comando è obsoleto o è disponibile per l'esecuzione sul dispositivo di destinazione. Per ulteriori informazioni sui diversi stati e sul motivo dello stato, vedere. Stato di esecuzione del comando

Aggiornare una risorsa di comando

Dopo aver creato un comando, è possibile aggiornare il nome visualizzato e la descrizione del comando.

> **Note**
>
> Il payload del comando non può essere aggiornato. Per aggiornare queste informazioni o utilizzare un payload modificato, è necessario creare un nuovo comando.

Aggiorna una risorsa di comando (console)

Per aggiornare un comando dalla console, vai al Command Hub della AWS IoT console ed esegui i seguenti passaggi.

1. Per aggiornare una risorsa di comando esistente, scegli il comando che desideri aggiornare, quindi in Azioni scegli Modifica.
2. Specificate il nome visualizzato e la descrizione che desiderate utilizzare e tutte le coppie nome-valore come tag per il comando.
3. Scegliete Modifica per salvare il comando con le nuove impostazioni.

Aggiornare una risorsa di comando (CLI)

Utilizza l'operazione API del piano di `UpdateCommand`controllo o `update-command` AWS CLI per aggiornare una risorsa di comando. Utilizzando questa API, puoi:

- Modificare il nome visualizzato e la descrizione di un comando creato.
- Deprecate una risorsa di comando o ripristinate un comando che è già stato obsoleto.

Policy IAM di esempio

Prima di utilizzare questa operazione API, assicurati che la tua policy IAM ti autorizzi a eseguire questa azione sul dispositivo. L'esempio seguente mostra una policy IAM che consente l'autorizzazione dell'utente a eseguire l'`UpdateCommand`azione.

In questo esempio, sostituisci:

- *`region`*con il tuo Regione AWS, ad esempio`us-east-1`.
- *`account-id`*con il tuo Account AWS numero, ad esempio*`123456789012`*.
- *`command-id`*con il tuo identificatore di comando AWS IoT univoco, ad esempio*`LockDoor`*. Se desideri recuperare più di un comando, puoi specificare questi comandi nella sezione Resource della policy IAM.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": "iot:UpdateCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Aggiorna le informazioni su un comando examples ()AWS CLI

L'esempio seguente mostra come aggiornare le informazioni su un comando utilizzando il `update-command` AWS CLI comando. Per informazioni su come utilizzare questa API per deprecare o ripristinare una risorsa di comando, consulta. Aggiornare una risorsa di comando (CLI)

L'esempio mostra come aggiornare il nome visualizzato e la descrizione di un comando. A seconda dell'applicazione, *<command-id>* sostituiscilo con l'identificatore del comando per il quale desideri recuperare le informazioni.

```
aws iot update-command \
    --command-id <command-id>
    --displayname <display-name> \
    --description <description>
```

L'esecuzione di questo comando genera una risposta che contiene le informazioni aggiornate sul comando e l'ora dell'ultimo aggiornamento. Il codice seguente mostra un esempio di richiesta e risposta per l'aggiornamento del nome visualizzato e della descrizione di un comando che spegne l'AC.

```
aws iot update-command \
    --command-id <LockDoor> \
    --displayname <Secondary lock door> \
    --description <Locks doors to my home>
```

L'esecuzione di questo comando genera la seguente risposta.

```
{
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/LockDoor",
    "displayName": "Secondary lock door",
    "description": "Locks doors to my home",
    "lastUpdatedAt": "2024-05-09T23:15:53.899000-07:00"
}
```

Deprecare o ripristinare una risorsa di comando

Dopo aver creato un comando, se non si desidera più continuare a utilizzarlo, è possibile contrassegnarlo come obsoleto. Quando si depreca un comando, tutte le esecuzioni di comandi in sospeso continueranno a essere eseguite sul dispositivo di destinazione fino a quando non raggiungono lo stato di terminale. Una volta che un comando è diventato obsoleto, se si desidera utilizzarlo, ad esempio per inviare una nuova esecuzione di comando al dispositivo di destinazione, è necessario ripristinarlo.

> **Note**
>
> Non è possibile modificare un comando obsoleto o eseguire nuove esecuzioni per esso. Per eseguire nuovi comandi sul dispositivo, è necessario ripristinarlo in modo che lo stato del comando passi a Disponibile.

Per ulteriori informazioni sulla deprecazione e il ripristino di un comando e sulle relative considerazioni, vedere. Deprecare una risorsa di comando

Eliminare una risorsa di comando

Se non desideri più utilizzare un comando, puoi rimuoverlo definitivamente dal tuo account. Se l'operazione di eliminazione ha esito positivo:

- Se il comando è obsoleto per una durata superiore al timeout massimo di 12 ore, il comando verrà eliminato immediatamente.
- Se il comando non è obsoleto o lo è stato per una durata inferiore al timeout massimo, il comando sarà in uno stato. `pending deletion` Verrà rimosso automaticamente dal tuo account dopo il timeout massimo di 12 ore.

> **Note**
>
> Il comando potrebbe essere eliminato anche se sono presenti esecuzioni di comandi in sospeso. Il comando sarà in sospeso di eliminazione e verrà rimosso automaticamente dal tuo account.

Eliminare una risorsa di comando (console)

Per eliminare un comando dalla console, accedi al Command Hub della AWS IoT console ed esegui i seguenti passaggi.

1. Scegli il comando che desideri eliminare, quindi in Azioni scegli Elimina.
2. Conferma di voler eliminare il comando, quindi scegli Elimina.

Il comando verrà contrassegnato per l'eliminazione e rimosso definitivamente dal tuo account dopo 12 ore.

Eliminare una risorsa di comando (CLI)

Utilizzate l'operazione API del piano di controllo `DeleteCommand` HTTP o il `delete-command` AWS CLI comando per eliminare una risorsa di comando. Se l'operazione di eliminazione ha esito positivo, vedrai un HTTP `statusCode` di 204 o 202 e il comando verrà eliminato automaticamente dal tuo account dopo la durata massima del timeout di 12 ore. Nel caso dello stato 204, indica che il comando è stato eliminato.

Policy IAM di esempio

Prima di utilizzare questa operazione API, assicurati che la tua policy IAM ti autorizzi a eseguire questa azione sul dispositivo. L'esempio seguente mostra una policy IAM che consente l'autorizzazione dell'utente a eseguire l'`DeleteCommand`azione.

In questo esempio, sostituisci:

- *`region`*con il tuo Regione AWS, ad esempio`us-east-1`.
- *`account-id`*con il tuo Account AWS numero, ad esempio*`123456789012`*.
- *`command-id`*con il tuo identificatore di comando AWS IoT univoco, ad esempio*`LockDoor`*. Se desideri recuperare più di un comando, puoi specificare questi comandi nella sezione Resource della policy IAM.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": "iot:DeleteCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Eliminare un esempio di comando ()AWS CLI

Gli esempi seguenti mostrano come eliminare un comando utilizzando il `delete-command` AWS CLI comando. A seconda dell'applicazione, *<command-id>* sostituiscilo con l'identificatore del comando che stai eliminando.

```
aws iot delete-command --command-id <command-id>
```

Se la richiesta API ha esito positivo, il comando genera un codice di stato di 202 o 204. Puoi utilizzare l'`GetCommand`API per verificare che il comando non esista più nel tuo account.

Avvia e monitora le esecuzioni di comandi

Dopo aver creato un comando, avvia un'esecuzione sul dispositivo di destinazione. Il dispositivo aggiorna i risultati e pubblica lo stato negli argomenti riservati MQTT. Recupera e monitora lo stato di esecuzione dal tuo account.

Avvia e monitora i comandi utilizzando la AWS IoT console o AWS CLI.

Avvia e monitora le operazioni dei comandi

- Avvia l'esecuzione di un comando
- Aggiorna il risultato dell'esecuzione di un comando
- Recupera l'esecuzione di un comando
- Visualizzazione degli aggiornamenti dei comandi utilizzando il client di test MQTT
- Elenca le esecuzioni dei comandi nel tuo Account AWS
- Eliminare l'esecuzione di un comando

Avvia l'esecuzione di un comando

> **Important**
>
> L'utente è l'unico responsabile della distribuzione dei comandi in modo sicuro e conforme alle leggi applicabili.

Prima di iniziare un'esecuzione, assicurati che:

- Hai creato un comando nello spazio dei AWS IoT nomi con le informazioni sul payload. All'avvio di Execution, il dispositivo elabora le istruzioni di Payload ed esegue azioni specificate. Vedi Crea una risorsa di comando per la creazione di comandi.
- Il dispositivo è abbonato agli argomenti riservati di MQTT per i comandi. All'avvio di Execution, le informazioni sul payload vengono pubblicate in questo argomento di richiesta MQTT riservato:

  *<devices>*possono essere client Things o MQTT. *<DeviceID>*è il nome dell'oggetto o l'ID del client. *<PayloadFormat>*Valori supportati: JSON e CBOR. Per ulteriori informazioni, consulta Argomenti sui comandi.

  ```
  $aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
  ```

  Per formati diversi da JSON/CBOR, usa questo formato Commands Topic*<PayloadFormat>*:

  ```
  $aws/commands/<devices>/<DeviceID>/executions/+/request
  ```

Considerazioni sul dispositivo di destinazione

Specificare il dispositivo di destinazione per ricevere ed eseguire il comando. Usa un nome oggetto per i dispositivi registrati o un ID client per i dispositivi non registrati. Dopo aver ricevuto il payload, il dispositivo esegue il comando ed esegue le azioni specificate.

AWS IoT cosa

I dispositivi di destinazione possono essere oggetti registrati nel AWS IoT registro. Le cose semplificano la ricerca e la gestione dei dispositivi.

Registra i dispositivi come oggetti dalla pagina del dispositivo Connect o utilizzando `CreateThing`. Trova oggetti esistenti da Thing Hub o li utilizza `DescribeThing`. Per i dettagli della registrazione, vedere Gestione degli elementi con il registro.

ID client

Per i dispositivi non registrati, usa il Client ID.

L'ID client è un identificatore univoco assegnato ai dispositivi. Definito nel protocollo MQTT, contiene caratteri alfanumerici, caratteri di sottolineatura o trattini. Ogni dispositivo a cui si connette AWS IoT necessita di un ID client univoco.

> **Note**
>
> - Per gli oggetti registrati, l'ID client può corrispondere al nome dell'oggetto.
> - Quando si sceglie come target un ID client specifico, i dispositivi devono connettersi AWS IoT utilizzando tale ID client per ricevere il payload.

Il Client ID è l'ID client MQTT utilizzato dai dispositivi per la connessione a. AWS IoT Core AWS IoT utilizza questo ID per identificare i dispositivi e gestire connessioni e abbonamenti.

Considerazioni sul timeout dell'esecuzione dei comandi

Il timeout specifica la durata (in secondi) con cui i dispositivi forniscono i risultati di esecuzione.

Dopo aver creato un'esecuzione, viene avviato un timer. Se il dispositivo va offline o non riporta i risultati entro il timeout, l'Esecuzione scade con lo stato`TIMED_OUT`.

Impostazione predefinita: 10 secondi. Massimo: 12 ore.

Valore del timeout e stato di **`TIMED_OUT`** esecuzione

Sia il cloud che il dispositivo possono segnalare il timeout.

Dopo aver inviato il comando, si avvia un timer. Se nessuna risposta del dispositivo arriva entro il timeout, il cloud imposta lo stato di esecuzione su `TIMED_OUT` con codice motivo. `$NO_RESPONSE_FROM_DEVICE`

Ciò si verifica quando:

- Il dispositivo è andato offline durante l'esecuzione.
- Il dispositivo non è riuscito a completare l'esecuzione entro il timeout.
- Il dispositivo non è riuscito a segnalare lo stato entro il timeout.

In questo caso, quando lo stato di esecuzione di `TIMED_OUT` viene segnalato dal cloud, l'esecuzione del comando non è terminale. Il dispositivo può pubblicare una risposta che sostituisce lo stato di uno qualsiasi degli stati del terminale:`SUCCEEDED`, o. `FAILED` `REJECTED` L'esecuzione del comando diventa quindi terminale e non accetta ulteriori aggiornamenti.

Il dispositivo può anche aggiornare uno `TIMED_OUT` stato avviato dal cloud segnalando che si è verificato un timeout durante l'esecuzione del comando. In questo caso, lo stato di esecuzione del

comando rimane invariato`TIMED_OUT`, ma l'`statusReason`oggetto viene aggiornato in base alle informazioni riportate dal dispositivo. L'esecuzione del comando diventa quindi terminale e non vengono accettati ulteriori aggiornamenti.

Utilizzo di sessioni persistenti MQTT

È possibile configurare le sessioni persistenti MQTT da utilizzare con la funzionalità dei AWS IoT Device Management comandi. Questa funzionalità è particolarmente utile in casi come quando il dispositivo va offline e si desidera assicurarsi che il dispositivo riceva ancora il comando quando torna online prima della durata del timeout ed esegua le istruzioni specificate.

Per impostazione predefinita, la scadenza della sessione persistente MQTT è impostata su 60 minuti. Se il timeout di esecuzione dei comandi è configurato su un valore superiore a tale durata, le esecuzioni di comandi che durano più di 60 minuti possono essere rifiutate dal broker di messaggi e avere esito negativo. Per eseguire comandi che durano più di 60 minuti, puoi richiedere un aumento del tempo di scadenza della sessione persistente.

> **Note**
>
> Per assicurarti di utilizzare correttamente la funzionalità delle sessioni persistenti MQTT, imposta il flag Clean Start su zero. Per ulteriori informazioni, consulta Sessioni persistenti MQTT.

Avvia l'esecuzione di un comando (console)

Per iniziare a eseguire il comando dalla console, vai alla pagina Command Hub della AWS IoT console ed esegui i seguenti passaggi.

1. Per eseguire il comando che hai creato, scegli Esegui comando.
2. Esamina le informazioni sul comando che hai creato, inclusi gli argomenti riservati MQTT e i parametri, se applicabile.

   Per i comandi dinamici, inserite i valori dei parametri o lasciateli con i valori predefiniti. Per i parametri che non hanno un valore predefinito, è necessario fornire un valore da inviare come parte di questa esecuzione.
3. Specificare il dispositivo di destinazione per ricevere ed eseguire il comando. Il dispositivo può essere specificato come AWS IoT oggetto se è stato registrato AWS IoT o utilizzando l'ID client

se il dispositivo non è ancora stato registrato. Per ulteriori informazioni, consulta Considerazioni sul dispositivo di destinazione

4. (Facoltativo) Configurate un valore di timeout per il comando che determini la durata per la quale desiderate che il comando venga eseguito prima del timeout. Se il comando deve essere eseguito per più di 60 minuti, potrebbe essere necessario aumentare il tempo di scadenza delle sessioni persistenti MQTT. Per ulteriori informazioni, consulta Considerazioni sul timeout dell'esecuzione dei comandi.
5. Seleziona Esegui comando.

Avviare l'esecuzione di un comando ()AWS CLI

Utilizzate l'operazione API del piano dati `StartCommandExecution`HTTP per avviare l'esecuzione di un comando. La richiesta e la risposta dell'API sono correlate dall'ID di esecuzione del comando. Una volta completata l'esecuzione del comando, il dispositivo può segnalare lo stato e il risultato dell'esecuzione al cloud pubblicando un messaggio nell'argomento di risposta ai comandi. Per un codice di risposta personalizzato, i codici applicativi di tua proprietà possono elaborare il messaggio di risposta e pubblicare il risultato su AWS IoT.

Se i tuoi dispositivi hanno sottoscritto l'argomento relativo alla richiesta dei comandi, l'`StartCommandExecution`API pubblicherà il messaggio di payload sull'argomento. Il payload può utilizzare qualsiasi formato di tua scelta. Per ulteriori informazioni, consulta Payload del comando.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
```

Se il formato del payload non è JSON o CBOR, di seguito viene illustrato il formato dell'argomento di richiesta dei comandi.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request
```

Policy IAM di esempio

Prima di utilizzare questa operazione API, assicurati che la tua policy IAM ti autorizzi a eseguire questa azione sul dispositivo. L'esempio seguente mostra una policy IAM che consente l'autorizzazione dell'utente a eseguire l'`StartCommandExecution`azione.

In questo esempio, sostituisci:

- *`region`*con il tuo Regione AWS, ad esempio`us-east-1`.

- *account-id*con il tuo Account AWS numero, ad esempio*123456789012*.
- *command-id*con un identificatore univoco per il AWS IoT comando, ad esempio*LockDoor*. Se desideri inviare più di un comando, puoi specificare questi comandi nella policy IAM.
- *devices*con uno dei due `thing` o `client` a seconda che i dispositivi siano stati registrati come AWS IoT oggetti o siano specificati come client MQTT.
- *device-id*con il tuo AWS IoT `thing-name` o. `client-id`

```
{
  "Effect": "Allow",
  "Action": [
      "iot:StartCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
  ]
}
```

Per visualizzare un elenco delle chiavi di condizione supportate`StartCommandExecution`, consulta Condition Keys for AWS IoT nella IAM User Guide.

Ottieni un endpoint del piano dati specifico dell'account

Prima di eseguire il comando API, è necessario ottenere l'URL dell'endpoint specifico dell'account per l'endpoint. Se utilizzi endpoint dual-stack (and), usa il. IPv4 IPv6 `iot:Data-ATS` L'`iot:Jobs`endpoint è solo per. IPv4 Ad esempio, se si esegue questo comando:

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

Restituisce l'URL dell'endpoint specifico dell'account, come mostrato nella risposta di esempio riportata di seguito.

```
{
    "endpointAddress":
    "<account-specific-prefix>-ats.iot.<region>.api.com"
}
```

Avvia un esempio di esecuzione di comando ()AWS CLI

L'esempio seguente mostra come iniziare l'esecuzione di un comando utilizzando il `start-command-execution` AWS CLI comando.

In questo esempio, sostituisci:

- *<command-arn>*con l'ARN per il comando che si desidera eseguire. È possibile ottenere queste informazioni dalla risposta del comando `create-command` CLI. Ad esempio, se stai eseguendo il comando per cambiare la modalità del volante, usa`arn:aws:iot:`*region*`:`*account-id*`:command/`*SetComfortSteeringMode*.
- *<target-arn>*con il Thing ARN per il dispositivo di destinazione, che può essere un oggetto IoT o un client MQTT, per il quale si desidera eseguire il comando. Ad esempio, se stai eseguendo il comando per il dispositivo di destinazione`myRegisteredThing`, usa. `arn:aws:iot:`*region*`:`*account-id*`:thing/`*myRegisteredThing*
- *<endpoint-url>*con l'endpoint specifico dell'account in cui hai ottenutoOttieni un endpoint del piano dati specifico dell'account, preceduto da. `https://` Ad esempio, `https://`*123456789012abcd*`.jobs.iot.`*us-east-1*`.amazonaws.com`.
- (Facoltativo) È inoltre possibile specificare un parametro aggiuntivo quando si esegue `executionTimeoutSeconds` l'operazione API. `StartCommandExecution` Questo campo opzionale specifica il tempo in secondi entro il quale il dispositivo deve completare l'esecuzione del comando. Per impostazione predefinita, il valore è 10 secondi. Quando lo stato di esecuzione del comando è impostato su`CREATED`, viene avviato un timer. Se il risultato dell'esecuzione del comando non viene ricevuto prima della scadenza del timer, lo stato cambia automaticamente in`TIMED_OUT`.
- ```
  aws iot-jobs-data start-command-execution \
      --command-arn <command-arn>  \
      --target-arn <target-arn> \
      --endpoint <endpoint-url> \
      --execution-timeout-seconds 900
  ```
- (Facoltativo) Per i comandi dinamici, specificate i parametri e i relativi valori da utilizzare per la sostituzione. È necessario fornire un valore per i parametri che non hanno un valore DefaultValue impostato al momento della creazione del comando. Se un parametro ha un DefaultValue, il valore del parametro fornito qui ha la precedenza. Per i parametri per cui è impostato ValueConditions, il valore del parametro fornito qui deve soddisfare la condizione.

  Basato su un esempio di comando `Light_Power_Status` dinamico:
  ```

- ```
  aws iot-jobs-data start-command-execution \
      --command-arn arn:aws:iot:us-east-1:123456789012:command/Light_Power_Status  \
      --target-arn arn:aws:iot:us-east-1:123456789012:thing/exampleThing \
      --endpoint <endpoint-url> \
      --execution-timeout-seconds 900 \
      --parameters "powerStatus={S=ON}"
  ```

L'esecuzione di questo comando restituisce un ID di esecuzione del comando. È possibile utilizzare questo ID per interrogare lo stato di esecuzione del comando, i dettagli e la cronologia di esecuzione dei comandi.

> **Note**
>
> Se il comando è obsoleto, la richiesta `StartCommandExecution` API avrà esito negativo con un'eccezione di convalida. Per correggere questo errore, ripristina prima il comando utilizzando l'`UpdateCommand`API, quindi esegui la richiesta. `StartCommandExecution`

```
{
    "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542"
}
```

Aggiorna il risultato dell'esecuzione di un comando

Utilizzate l'operazione API del piano dati `UpdateCommandExecution` MQTT per aggiornare lo stato o il risultato dell'esecuzione di un comando.

> **Note**
>
> Prima di utilizzare questa API:
>
> - Il dispositivo deve aver stabilito una connessione MQTT e aver sottoscritto gli argomenti di richiesta e risposta dei comandi. Per ulteriori informazioni, consulta Flusso di lavoro dei comandi di alto livello.
> - È necessario aver già eseguito questo comando utilizzando l'operazione `StartCommandExecution` API.
```

Policy IAM di esempio

Prima di utilizzare questa operazione API, assicurati che la tua policy IAM autorizzi il tuo dispositivo a eseguire queste azioni. Di seguito è riportato un esempio di policy che autorizza il dispositivo a eseguire l'azione. Per ulteriori esempi di policy IAM che consentono l'autorizzazione dell'utente a eseguire l'azione `UpdateCommandExecution` MQTT, vedere. Esempi di policy di connessione e pubblicazione

In questo esempio, sostituisci:

- *`Region`*con il tuo Regione AWS, ad esempio`us-east-1`.
- *`AccountID`*con il tuo Account AWS numero, ad esempio*`123456789012`*.
- *`ThingName`*con il nome dell' AWS IoT oggetto a cui intendi eseguire il comando, ad esempio*`myRegisteredThing`*.
- *`commands-request-topic`*e *`commands-response-topic`* con i nomi degli argomenti di richiesta e risposta dei AWS IoT comandi. Per ulteriori informazioni, consulta Flusso di lavoro dei comandi di alto livello.

Esempio di policy IAM per l'ID client MQTT

Il codice seguente mostra un esempio di policy del dispositivo quando si utilizza l'ID client MQTT.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
```

```
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/request",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/request/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/request",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/request/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/${iot:ClientId}"
    }
  ]
}
```

Esempio di policy IAM per l'IoT

Il codice seguente mostra un esempio di policy relativa ai dispositivi quando si utilizza un AWS IoT oggetto.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/request",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/request/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/request",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/rejected",
```

```
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/request/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/${iot:ClientId}"
    }
  ]
}
```

Come usare l'**`UpdateCommandExecution`**API

Dopo aver ricevuto l'esecuzione del comando nell'argomento della richiesta, il dispositivo elabora il comando. Utilizza quindi l'`UpdateCommandExecution`API per aggiornare lo stato e il risultato dell'esecuzione del comando in base al seguente argomento di risposta.

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/response/<PayloadFormat>
```

In questo esempio, *`<DeviceID>`* è l'identificatore univoco del dispositivo di destinazione ed *`<execution-id>`* è l'identificatore dell'esecuzione del comando sul dispositivo di destinazione. *`<PayloadFormat>`*Possono essere JSON o CBOR.

> **Note**
>
> Se non hai registrato il dispositivo con AWS IoT, puoi utilizzare l'ID cliente come identificatore anziché il nome dell'oggetto.

```
$aws/commands/clients/<ClientID>/executions/<ExecutionId>/response/<PayloadFormat>
```

Il dispositivo ha segnalato gli aggiornamenti dello stato di esecuzione

I tuoi dispositivi possono utilizzare l'API per segnalare uno dei seguenti aggiornamenti di stato all'esecuzione del comando. Per ulteriori informazioni su questi stati, consultaStato di esecuzione del comando.

- `IN_PROGRESS`: Quando il dispositivo inizia a eseguire il comando, può aggiornare lo stato a`IN_PROGRESS`.
- `SUCCEEDED`: Quando il dispositivo elabora correttamente il comando e completa l'esecuzione, può pubblicare un messaggio nell'argomento di risposta come`SUCCEEDED`.
- `FAILED`: Se il dispositivo non è riuscito a eseguire il comando, può pubblicare un messaggio nell'argomento di risposta come`FAILED`.
- `REJECTED`: Se il dispositivo non accetta il comando, può pubblicare un messaggio nell'argomento di risposta come`REJECTED`.
- `TIMED_OUT`: Lo stato di esecuzione del comando può cambiare in `TIMED_OUT` causa di uno dei seguenti motivi.
  - Il risultato dell'esecuzione del comando non è stato ricevuto. Ciò può accadere perché l'esecuzione non è stata completata entro la durata specificata o se il dispositivo non è riuscito a pubblicare le informazioni sullo stato nell'argomento di risposta.
  - Il dispositivo segnala che si è verificato un timeout durante il tentativo di esecuzione del comando.

Per ulteriori informazioni sullo `TIMED_OUT` stato, vedereValore del timeout e stato di `TIMED_OUT` esecuzione.

Considerazioni sull'utilizzo dell'API **`UpdateCommandExecution`**

Di seguito sono riportate alcune considerazioni importanti sull'utilizzo dell'`UpdateCommandExecution`API.

- I tuoi dispositivi possono utilizzare un `statusReason` oggetto opzionale per fornire informazioni aggiuntive sull'esecuzione. Se i dispositivi forniscono questo oggetto, il `reasonCode` campo dell'oggetto è obbligatorio, ma il `reasonDescription` campo è facoltativo.
- Quando i dispositivi utilizzano l'`statusReason`oggetto, `reasonCode` devono utilizzare lo schema `[A-Z0-9_-]+` e non superare i 64 caratteri di lunghezza. Se lo fornisci`reasonDescription`, assicurati che non superi i 1.024 caratteri di lunghezza. Può utilizzare qualsiasi carattere tranne i caratteri di controllo come le nuove righe.

- I dispositivi possono utilizzare un `result` oggetto opzionale per fornire informazioni sul risultato dell'esecuzione del comando, ad esempio il valore restituito da una chiamata di funzione remota. Se fornite il`result`, deve essere necessario inserire almeno una voce.
- Nel `result` campo, si specificano le voci come coppie chiave-valore. Per ogni voce, è necessario specificare le informazioni sul tipo di dati come stringa, booleana o binaria. Un tipo di dati stringa deve utilizzare la chiaves, un tipo di dati booleano utilizza la chiave b e un tipo di dati binario deve utilizzare la chiave. `bin` Assicurati che queste chiavi siano minuscole.
- Se riscontri un errore durante l'esecuzione dell'`UpdateCommandExecution`API, puoi visualizzare l'errore nel gruppo di `AWSIoTLogsV2` log di Amazon CloudWatch. Per informazioni sull'abilitazione della registrazione e sulla visualizzazione dei log, consulta. Configurare la registrazione AWS IoT

`UpdateCommandExecution`Esempio di API

Il codice seguente mostra un esempio di come il dispositivo può utilizzare l'`UpdateCommandExecution`API per segnalare lo stato di esecuzione, il `statusReason` campo per fornire informazioni aggiuntive sullo stato e il campo dei risultati per fornire informazioni sul risultato dell'esecuzione, ad esempio la percentuale della batteria dell'auto in questo caso.

```
{
  "status": "IN_PROGRESS",
  "statusReason": {
    "reasonCode": "200",
    "reasonDescription": "Execution_in_progress"
  },
  "result": {
        "car_battery": {
            "s": "car battery at 50 percent"
        }
    }
}
```

Recupera l'esecuzione di un comando

Dopo aver eseguito un comando, è possibile recuperare informazioni sull'esecuzione del comando dalla AWS IoT console e utilizzare il. AWS CLIÈ possibile ottenere le seguenti informazioni.

> **Note**
>
> Per recuperare lo stato di esecuzione del comando più recente, il dispositivo deve pubblicare le informazioni sullo stato nell'argomento della risposta utilizzando l'API `UpdateCommandExecution` MQTT, come descritto di seguito. Fino a quando il dispositivo non pubblicherà questo argomento, l'`GetCommandExecution`API riporterà lo stato come o. `CREATED` `TIMED_OUT`

Ogni esecuzione di comando creata avrà:

- Un ID di esecuzione, che è un identificatore univoco dell'esecuzione del comando.
- Lo stato dell'esecuzione del comando. Quando si esegue il comando sul dispositivo di destinazione, l'esecuzione del comando entra in uno stato `CREATED`. Può quindi passare ad altri stati di esecuzione dei comandi come descritto di seguito.
- Il risultato dell'esecuzione del comando.
- L'ID di comando univoco e il dispositivo di destinazione per il quale sono state create le esecuzioni.
- La data di inizio, che indica l'ora in cui è stata creata l'esecuzione del comando.

Recupera l'esecuzione di un comando (console)

È possibile recuperare l'esecuzione di un comando dalla console utilizzando uno dei seguenti metodi.

- Dalla pagina Command Hub

  Vai alla pagina Command Hub della AWS IoT console ed esegui questi passaggi.

  1. Scegli il comando per il quale hai creato un'esecuzione sul dispositivo di destinazione.
  2. Nella pagina dei dettagli del comando, nella scheda Cronologia dei comandi, vedrai le esecuzioni che hai creato. Scegli l'esecuzione per la quale desideri recuperare le informazioni.
  3. Se i tuoi dispositivi hanno utilizzato l'`UpdateCommandExecution`API per fornire le informazioni sui risultati, puoi trovare queste informazioni nella scheda Risultati di questa pagina.
- Dalla pagina Thing hub

  Se si è scelto un AWS IoT oggetto come dispositivo di destinazione durante l'esecuzione del comando, è possibile visualizzare i dettagli di esecuzione dalla pagina Thing hub.

1. Vai alla pagina Thing Hub nella AWS IoT console e scegli l'oggetto per cui hai creato l'esecuzione del comando.
2. Nella pagina dei dettagli dell'oggetto, nella cronologia dei comandi, vedrai le esecuzioni che hai creato. Scegli l'esecuzione per la quale desideri recuperare le informazioni.
3. Se i tuoi dispositivi hanno utilizzato l'`UpdateCommandExecution`API per fornire le informazioni sui risultati, puoi trovare queste informazioni nella scheda Risultati di questa pagina.

Recupera l'esecuzione di un comando (CLI)

Utilizzate l'operazione API HTTP del piano di `GetCommandExecution` AWS IoT Core controllo per recuperare informazioni sull'esecuzione di un comando. È necessario aver già eseguito questo comando utilizzando l'operazione `StartCommandExecution` API.

Policy IAM di esempio

Prima di utilizzare questa operazione API, assicurati che la tua policy IAM ti autorizzi a eseguire questa azione sul dispositivo. L'esempio seguente mostra una policy IAM che consente l'autorizzazione dell'utente a eseguire l'`GetCommandExecution`azione.

In questo esempio, sostituisci:

- *`region`*con il tuo Regione AWS, ad esempio`us-east-1`.
- *`account-id`*con il tuo Account AWS numero, ad esempio*`123456789012`*.
- *`command-id`*con il tuo identificatore di AWS IoT comando univoco, ad esempio*`LockDoor`*.
- *`devices`*con uno dei due `thing` o `client` a seconda che i dispositivi siano stati registrati come AWS IoT oggetti o siano specificati come client MQTT.
- *`device-id`*con il tuo AWS IoT `thing-name` o. `client-id`

```
{
  "Effect": "Allow",
  "Action": [
      "iot:GetCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
```

```
  ]
}
```

Recupera un esempio di esecuzione di comando

L'esempio seguente mostra come recuperare informazioni su un comando che è stato eseguito utilizzando il `start-command-execution` AWS CLI comando. L'esempio seguente mostra come recuperare informazioni su un comando che è stato eseguito per disattivare la modalità volante.

In questo esempio, sostituisci:

- *<execution-id>*con l'identificatore per l'esecuzione del comando per il quale si desidera recuperare le informazioni.
- *<target-arn>*con l'Amazon Resource Number (ARN) del dispositivo a cui stai indirizzando l'esecuzione. È possibile ottenere queste informazioni dalla risposta del comando `start-command-execution` CLI.
- Facoltativamente, se i dispositivi hanno utilizzato l'`UpdateCommandExection`API per fornire il risultato dell'esecuzione, è possibile specificare se includere il risultato dell'esecuzione del comando nella risposta dell'API che utilizza l'`GetCommandExecution`API`GetCommandExecution`.

```
aws iot get-command-execution
    --execution-id <execution-id> \
    --target-arn <target-arn> \
    --include-result
```

L'esecuzione di questo comando genera una risposta che contiene informazioni sull'ARN dell'esecuzione del comando, sullo stato dell'esecuzione e sull'ora in cui è iniziata l'esecuzione e quando è stata completata. Fornisce inoltre un `statusReason` oggetto che contiene informazioni aggiuntive sullo stato. Per ulteriori informazioni sui diversi stati e sul motivo dello stato, vedereStato di esecuzione del comando.

Il codice seguente mostra un esempio di risposta dalla richiesta API.

> **Note**
>
> Il `completedAt` campo nella risposta di esecuzione corrisponde all'ora in cui il dispositivo segnala lo stato di un terminale al cloud. Nel caso dello `TIMED_OUT` status, questo campo

verrà impostato solo quando il dispositivo segnala un timeout. Quando lo `TIMED_OUT` stato è impostato dal cloud, lo `TIMED_OUT` stato non viene aggiornato. Per ulteriori informazioni sul comportamento del timeout, consultaConsiderazioni sul timeout dell'esecuzione dei comandi.

```
{
    "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/LockDoor",
    "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/myRegisteredThing",
    "status": "SUCCEEDED",
    "statusReason": {
        "reasonCode": "DEVICE_SUCCESSFULLY_EXECUTED",
        "reasonDescription": "SUCCESS"
    },
    "result": {
        "sn": { "s": "ABC-001" },
        "digital": { "b": true }
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "completedAt": "2024-03-23T00:50:10.095000-07:00"
}
```

Visualizzazione degli aggiornamenti dei comandi utilizzando il client di test MQTT

È possibile utilizzare il client di test MQTT per visualizzare lo scambio di messaggi su MQTT quando si utilizza la funzionalità dei comandi. Dopo che il dispositivo ha stabilito una connessione MQTT con AWS IoT, è possibile creare un comando, specificare il payload e quindi eseguirlo sul dispositivo. Quando si esegue il comando, se il dispositivo ha sottoscritto l'argomento di richiesta riservata MQTT per i comandi, viene visualizzato il messaggio di payload pubblicato su questo argomento.

Il dispositivo riceve quindi le istruzioni di payload ed esegue le operazioni specificate sul dispositivo. AWS IoT Utilizza quindi l'`UpdateCommandExecution`API per pubblicare il risultato dell'esecuzione del comando e le informazioni sullo stato negli argomenti di risposta riservati MQTT per i comandi. AWS IoT Device Management ascolta gli aggiornamenti sugli argomenti di risposta e archivia le informazioni aggiornate e pubblica i log su Amazon. AWS CloudTrail CloudWatch È quindi possibile recuperare le informazioni più recenti sull'esecuzione dei comandi dalla console o utilizzando l'API. `GetCommandExecution`

I passaggi seguenti mostrano come utilizzare il client di test MQTT per osservare i messaggi.

1. Aprire il client di test MQTT nella AWS IoT console.
2. Nella scheda Iscriviti, inserisci il seguente argomento e poi scegli Iscriviti, *<thingId>* dov'è il nome del dispositivo con AWS IoT cui ti sei registrato.

   > **Note**
   >
   > Puoi trovare il nome dell'oggetto per il tuo dispositivo nella pagina Thing Hub della AWS IoT console. Se non hai registrato il dispositivo come oggetto, puoi registrarlo al momento della connessione AWS IoT dalla pagina Connect device.

   ```
   $aws/commands/things/<thingId>/executions/+/request
   ```

3. (Facoltativo) Nella scheda Iscriviti, puoi anche inserire i seguenti argomenti e scegliere Iscriviti.

   ```
   $aws/commands/things/+/executions/+/response/accepted/json
   $aws/commands/things/+/executions/+/response/rejected/json
   ```

4. Quando si avvia l'esecuzione di un comando, il payload del messaggio verrà inviato al dispositivo utilizzando l'argomento di richiesta a cui il dispositivo è abbonato,. `$aws/commands/things/<thingId>/executions/+/request` Nel client di test MQTT, dovresti vedere il payload del comando che contiene le istruzioni per l'elaborazione del comando da parte del dispositivo.
5. Dopo l'avvio dell'esecuzione del comando, il dispositivo può pubblicare aggiornamenti di stato al seguente argomento di risposta riservato MQTT per i comandi.

   ```
   $aws/commands/<devices>/<device-id>/executions/<executionId>/response/json
   ```

   Ad esempio, prendete in considerazione un comando che avete eseguito per accendere l'aria condizionata dell'auto per ridurre la temperatura al valore desiderato. Il codice JSON seguente mostra un messaggio di esempio pubblicato dal veicolo nell'argomento di risposta, che mostra che non è riuscito a eseguire il comando.

   ```
   {
     "deviceId": "My_Car",
     "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542",
   ```

```
  "status": "FAILED",
  "statusReason": {
    "reasonCode": "CAR_LOW_ON_BATTERY",
    "reasonDescription": "Car battery is lower than 5 percent"
  }
}
```

In questo caso, puoi caricare la batteria dell'auto e quindi eseguire nuovamente il comando.

Elenca le esecuzioni dei comandi nel tuo Account AWS

Dopo aver eseguito un comando, è possibile recuperare informazioni sull'esecuzione del comando dalla AWS IoT console e utilizzare il. AWS CLIÈ possibile ottenere le seguenti informazioni.

- Un ID di esecuzione, che è un identificatore univoco dell'esecuzione del comando.
- Lo stato dell'esecuzione del comando. Quando si esegue il comando sul dispositivo di destinazione, l'esecuzione del comando entra in uno stato `CREATED`. Può quindi passare ad altri stati di esecuzione dei comandi come descritto di seguito.
- L'ID di comando univoco e il dispositivo di destinazione per il quale sono state create le esecuzioni.
- La data di inizio, che indica l'ora in cui è stata creata l'esecuzione del comando.

Elenca le esecuzioni dei comandi nel tuo account (console)

Puoi vedere tutte le esecuzioni dei comandi dalla console utilizzando uno dei seguenti metodi.

- Dalla pagina Command hub

  Vai alla pagina Command Hub della AWS IoT console ed esegui questi passaggi.

  1. Scegli il comando per il quale hai creato un'esecuzione sul dispositivo di destinazione.
  2. Nella pagina dei dettagli del comando, vai alla scheda Cronologia dei comandi e vedrai un elenco delle esecuzioni che hai creato.
- Dalla pagina Thing hub

  Se si è scelto un AWS IoT oggetto come dispositivo di destinazione durante l'esecuzione del comando e si sono create più esecuzioni di comandi per un singolo dispositivo, è possibile visualizzare le esecuzioni per il dispositivo dalla pagina Thing hub.

1. Vai alla pagina Thing Hub nella AWS IoT console e scegli l'oggetto per cui hai creato le esecuzioni.
2. Nella pagina dei dettagli dell'oggetto, nella cronologia dei comandi, vedrai un elenco di esecuzioni che hai creato per il dispositivo.

Elenca le esecuzioni dei comandi nel tuo account (CLI)

Utilizza l'operazione API HTTP del piano di `ListCommandExecutions` AWS IoT Core controllo per elencare tutte le esecuzioni di comandi nel tuo account.

Policy IAM di esempio

Prima di utilizzare questa operazione API, assicurati che la tua policy IAM ti autorizzi a eseguire questa azione sul dispositivo. L'esempio seguente mostra una policy IAM che consente l'autorizzazione dell'utente a eseguire l'`ListCommandExecutions`azione.

In questo esempio, sostituisci:

- *`region`*con il tuo Regione AWS, ad esempio`us-east-1`.
- *`account-id`*con il tuo Account AWS numero, ad esempio*`123456789012`*.
- *`command-id`*con il tuo identificatore di AWS IoT comando univoco, ad esempio*`LockDoor`*.

```
{
  "Effect": "Allow",
  "Action": "iot:ListCommandExecutions",
  "Resource": *
}
```

Elenca un esempio di esecuzione di comandi

L'esempio seguente mostra come elencare le esecuzioni di comandi in. Account AWS

Quando si esegue il comando, è necessario specificare se filtrare l'elenco per visualizzare solo le esecuzioni di comandi create per un particolare dispositivo utilizzando il`targetArn`, o le esecuzioni per un particolare comando specificato utilizzando. `commandArn`

In questo esempio, sostituisci:

- *<target-arn>*con l'Amazon Resource Number (ARN) del dispositivo a cui stai indirizzando l'esecuzione, ad esempio. `arn:aws:iot:`*us-east-1*`:`*123456789012*`:thing/`*b8e4157c98f332cffb37627f*
- *<target-arn>*con l'Amazon Resource Number (ARN) del dispositivo a cui stai indirizzando l'esecuzione, ad esempio. `arn:aws:iot:`*us-east-1*`:`*123456789012*`:thing/`*b8e4157c98f332cffb37627f*
- *<after>*con il tempo dopo il quale desideri elencare le esecuzioni che sono state create, ad esempio. `2024-11-01T03:00`

```
aws iot list-command-executions \
--target-arn <target-arn> \
--started-time-filter '{after=<after>}' \
--sort-order "ASCENDING"
```

L'esecuzione di questo comando genera una risposta che contiene un elenco di esecuzioni di comandi create e l'ora in cui le esecuzioni hanno iniziato l'esecuzione e quando sono state completate. Fornisce inoltre informazioni sullo stato e l'`statusReason`oggetto che contiene informazioni aggiuntive sullo stato.

```
{
    "commandExecutions": [
        {
            "commandArn": "arn:aws:iot:us-east-1:123456789012:command/TestMe002",
            "executionId": "b2b654ca-1a71-427f-9669-e74ae9d92d24",
            "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/
b8e4157c98f332cffb37627f",
            "status": "TIMED_OUT",
            "createdAt": "2024-11-24T14:39:25.791000-08:00",
            "startedAt": "2024-11-24T14:39:25.791000-08:00"
        },
        {
            "commandArn": "arn:aws:iot:us-east-1:123456789012:command/TestMe002",
            "executionId": "34bf015f-ef0f-4453-acd0-9cca2d42a48f",
            "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/
b8e4157c98f332cffb37627f",
            "status": "IN_PROGRESS",
            "createdAt": "2024-11-24T14:05:36.021000-08:00",
            "startedAt": "2024-11-24T14:05:36.021000-08:00"
        }
```

```
    ]
}
```

Per ulteriori informazioni sui diversi stati e sul motivo dello stato, vedereStato di esecuzione del comando.

Eliminare l'esecuzione di un comando

Se non desideri più utilizzare l'esecuzione di un comando, puoi rimuoverla definitivamente dal tuo account.

> **Note**
>
> - L'esecuzione di un comando può essere eliminata solo se è entrato in uno stato di terminale`SUCCEEDED`, ad esempio`FAILED`, o`REJECTED`.
> - Questa operazione può essere eseguita solo utilizzando l' AWS IoT Core API o il AWS CLI. Non è disponibile dalla console.

Policy IAM di esempio

Prima di utilizzare questa operazione API, assicurati che la tua policy IAM autorizzi il tuo dispositivo a eseguire queste azioni. Di seguito è riportato un esempio di policy che autorizza il dispositivo a eseguire l'azione.

In questo esempio, sostituisci:

- *`Region`*con il tuo Regione AWS, ad esempio`us-east-1`.
- *`AccountID`*con il tuo Account AWS numero, ad esempio*`123456789012`*.
- *`CommandID`*con l'identificatore del comando di cui si desidera eliminare l'esecuzione.
- *`devices`*con uno dei due `thing` o `client` a seconda che i dispositivi siano stati registrati come AWS IoT oggetti o siano specificati come client MQTT.
- *`device-id`*con il tuo AWS IoT `thing-name` o. `client-id`

```
{
  "Effect": "Allow",
  "Action": [
```

```
        "iot:DeleteCommandExecution"
    ],
    "Resource": [
        "arn:aws:iot:region:account-id:command/command-id",
        "arn:aws:iot:region:account-id:devices/device-id"
    ]
}
```

Eliminare un esempio di esecuzione di comando

L'esempio seguente mostra come eliminare un comando utilizzando il `delete-command` AWS CLI comando. A seconda dell'applicazione, *<execution-id>* sostituiscilo con l'identificatore per l'esecuzione del comando che stai eliminando e poi *<target-arn>* con l'ARN del dispositivo di destinazione.

```
aws iot delete-command-execution \
--execution-id <execution-id> \
--target-arn <target-arn>
```

Se la richiesta API ha esito positivo, l'esecuzione del comando genera un codice di stato 200. Puoi utilizzare l'`GetCommandExecution`API per verificare che l'esecuzione del comando non esista più nel tuo account.

Deprecare una risorsa di comando

Deprecate i comandi per indicare che sono obsoleti e non devono essere utilizzati. Ad esempio, i comandi obsoleti non vengono più mantenuti attivamente o quando si creano comandi più recenti con lo stesso ID ma payload diversi.

Considerazioni chiave

Considerazioni importanti sulla deprecazione dei comandi:

- La deprecazione di un comando non lo elimina. È possibile recuperare il comando utilizzando il relativo ID e ripristinarlo per riutilizzarlo.
- Il tentativo di avviare nuove esecuzioni su comandi obsoleti genera un errore che impedisce l'uso di comandi obsoleti.
- Per eseguire un comando obsoleto, è necessario innanzitutto ripristinarlo. Dopo il ripristino, il comando diventa disponibile per l'uso e l'esecuzione regolari sui dispositivi di destinazione.

- Se si rende obsoleto un comando mentre le esecuzioni sono in corso, queste continuano a funzionare fino al completamento. È comunque possibile recuperare lo stato di esecuzione.

Deprecare una risorsa di comando (console)

Per rendere obsoleto un comando dalla console, accedi al Command Hub della AWS IoT console ed esegui i seguenti passaggi.

1. Scegli il comando che desideri rendere obsoleto, quindi in Azioni scegli Deprecate.
2. Conferma che desideri rendere obsoleto il comando, quindi scegli Deprecate.

Deprecare una risorsa di comando (CLI)

Contrassegna i comandi come obsoleti utilizzando la CLI. `update-command` È necessario rendere obsoleto un comando prima dell'eliminazione. Per utilizzare un comando obsoleto, ripristinalo prima.

```
aws iot update-command \
    --command-id <command-id> \
    --deprecated
```

Ad esempio, se avete reso obsoleto il *ACSwitch* comando aggiornato nell'esempio precedente, il codice seguente mostra un esempio di output dell'esecuzione del comando.

```
{
    "commandId": "turnOffAc",
    "deprecated": true,
    "lastUpdatedAt": "2024-05-09T23:16:51.370000-07:00"
}
```

Controlla l'ora e lo stato di deprecazione

Utilizza l'`GetCommand`API per determinare se un comando è obsoleto e quando è stato deprecato l'ultima volta.

```
aws iot get-command --command-id <turnOffAC>
```

Questo comando genera una risposta contenente informazioni sul comando, inclusi i timestamp di creazione e deprecazione relativi all'ultimo campo aggiornato. Ciò consente di determinare la durata

del comando e se eliminarlo o riutilizzarlo. Di seguito viene illustrato un esempio di risposta per il *turnOffAc* comando:

```
{
    "commandId": "turnOffAC",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/turnOffAC",
    "namespace": "AWS-IoT",
    "payload": {
        "content": "testPayload.json",
        "contentType": "application/json"
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "lastUpdatedAt": "2024-05-09T23:16:51.370000-07:00",
    "deprecated": false
}
```

Ripristina una risorsa di comando

Per utilizzare o inviare il `ACSwitch` comando al dispositivo, ripristinalo prima.

Per ripristinare un comando dalla console, vai al Command Hub della AWS IoT console, scegli il comando che desideri ripristinare, quindi in Azioni scegli Ripristina.

Per ripristinare un comando utilizzando l' AWS IoT Core API o il AWS CLI, utilizza l'operazione `UpdateCommand` API o la `update-command` CLI. Il codice seguente mostra un esempio di richiesta e risposta.

```
aws iot update-command \
    --command-id <command-id>
    --no-deprecated
```

Il codice seguente mostra un output di esempio.

```
{
    "commandId": "ACSwitch",
    "deprecated": false,
    "lastUpdatedAt": "2024-05-09T23:17:21.954000-07:00"
}
```

AWS IoT tunneling sicuro

Quando i dispositivi vengono implementati dietro firewall con restrizioni in siti remoti, è necessario disporre di un modo per accedere a tali dispositivi per la risoluzione dei problemi, gli aggiornamenti della configurazione e altre attività operative. Utilizza il tunneling sicuro per stabilire una comunicazione bidirezionale con i dispositivi remoti tramite una connessione sicura gestita da. AWS IoT Il tunneling sicuro non richiede aggiornamenti alle regole del firewall in ingresso esistente, pertanto è possibile mantenere lo stesso livello di sicurezza fornito dalle regole firewall in un sito remoto.

Ad esempio, un dispositivo sensore situato in una fabbrica che si trova a poche centinaia di miglia di distanza sta avendo problemi a misurare la temperatura di fabbrica. È possibile utilizzare il tunneling sicuro per aprire e avviare rapidamente una sessione su quel dispositivo sensore. Dopo aver identificato il problema (ad esempio, un file di configurazione non valido), è possibile reimpostare il file e riavviare il dispositivo sensore attraverso la stessa sessione. Rispetto a una risoluzione dei problemi più tradizionale (ad esempio, l'invio di un tecnico in fabbrica per indagare sul dispositivo sensore), il tunneling sicuro riduce la risposta agli incidenti, i tempi di recupero e i costi operativi.

Che cos'è il tunneling sicuro?

Utilizza il tunneling sicuro per accedere ai dispositivi implementati dietro firewall con restrizioni di porte nei siti remoti. Puoi connetterti al dispositivo di destinazione dal computer portatile o desktop come dispositivo di origine utilizzando il Cloud AWS. L'origine e la destinazione comunicano tra loro utilizzando un proxy locale open source che viene eseguito su ciascun dispositivo. Il proxy locale comunica con il utilizzando una porta aperta consentita Cloud AWS dal firewall, in genere 443. I dati trasmessi attraverso il tunnel vengono crittografati tramite Transport Layer Security (TLS).

Argomenti

- Concetti di tunneling sicuri
- Come funziona il tunneling sicuro
- Ciclo di vita sicuro del tunnel

Concetti di tunneling sicuri

Quando stabilisce la comunicazione con i dispositivi remoti, il tunneling sicuro utilizza i seguenti termini. Per ulteriori informazioni sul funzionamento del tunneling sicuro, consulta la sezione Come funziona il tunneling sicuro.

Token di accesso client (CAT)

Una coppia di token generati dal tunneling sicuro quando viene creato un nuovo tunnel. Il CAT viene utilizzato dai dispositivi di origine e di destinazione per connettersi al servizio di tunneling sicuro. Il CAT può essere utilizzato una sola volta per la connessione al tunnel. Per riconnetterti al tunnel, ruota i token di accesso del client utilizzando l'operazione RotateTunnelAccessTokenAPI o il comando rotate-tunnel-access-tokenCLI.

Token client

Un valore unico generato dal client che il tunneling AWS IoT sicuro può utilizzare per tutti i tentativi di connessione successivi allo stesso tunnel. Questo campo è facoltativo. Se il token client non viene fornito, il token di accesso client (CAT) può essere utilizzato una sola volta per lo stesso tunnel. I tentativi di connessione successivi che utilizzano lo stesso CAT vengono rifiutati. Per ulteriori informazioni sull'utilizzo dei token client, consulta l'implementazione di riferimento del proxy locale in. GitHub

Applicazione di destinazione

L'applicazione che viene eseguita sul dispositivo di destinazione. Ad esempio, l'applicazione di destinazione può essere un daemon SSH per stabilire una sessione SSH utilizzando il tunneling sicuro.

Dispositivo di destinazione

Il dispositivo remoto a cui si desidera accedere.

Agente del dispositivo

Un'applicazione IoT che si connette al gateway del AWS IoT dispositivo e ascolta le nuove notifiche del tunnel tramite MQTT. Per ulteriori informazioni, consulta Snippet dell'agente IoT.

Proxy locale

Un proxy software che viene eseguito sui dispositivi di origine e di destinazione e inoltra un flusso dei dati tra il tunneling sicuro e l'applicazione del dispositivo. Il proxy locale può essere eseguito in modalità sorgente o in modalità di destinazione. Per ulteriori informazioni, consulta Proxy locale.

Dispositivo di origine

Il dispositivo utilizzato da un operatore per avviare una sessione sul dispositivo di destinazione, in genere un computer portatile o desktop.

Tunnel

Un percorso logico AWS IoT che consente la comunicazione bidirezionale tra un dispositivo di origine e un dispositivo di destinazione.

Come funziona il tunneling sicuro

Di seguito viene illustrato il modo in cui il tunneling sicuro stabilisce una connessione tra il dispositivo di origine e quello di destinazione. Per informazioni sui diversi termini, come ad esempio il token di accesso client (CAT), consulta la sezione Concetti di tunneling sicuri.

1. Apertura di un tunnel

   Per aprire un tunnel per l'avvio di una sessione con il dispositivo di destinazione remoto, è possibile utilizzare il Console di gestione AWS comando AWS CLI open-tunnel o l'API. OpenTunnel

2. Eseguire il download della coppia di token di accesso client

   Dopo avere aperto un tunnel, puoi effettuare il download del token di accesso client (CAT) per l'origine e la destinazione e salvarlo sul dispositivo di origine. Il CAT deve essere recuperato e salvato in questa fase prima di avviare il proxy locale.

3. Avvio del proxy locale nella modalità di destinazione

   L'agente IoT che è stato installato ed è in esecuzione sul dispositivo di destinazione verrà sottoscritto all'argomento MQTT riservato `$aws/things/thing-name/tunnels/notify` e riceverà il CAT. *`thing-name`*Ecco il nome dell' AWS IoT oggetto che crei per la tua destinazione. Per ulteriori informazioni, consulta Argomenti di tunneling sicuro.

   L'agente IoT utilizza quindi il CAT per avviare il proxy locale in modalità di destinazione e impostare una connessione sul lato di destinazione del tunnel. Per ulteriori informazioni, consulta Snippet dell'agente IoT.

4. Avvio del proxy locale in modalità di origine

Dopo l'apertura del tunnel, AWS IoT Device Management fornisce il CAT del codice sorgente che è possibile scaricare sul dispositivo sorgente. Puoi utilizzare il CAT per avviare il proxy locale nella modalità di origine, che quindi collega il lato di origine del tunnel. Per ulteriori informazioni sul proxy locale, consulta la sezione Proxy locale.

5. Apertura di una sessione SSH

   Poiché entrambi i lati del tunnel sono collegati, puoi avviare una sessione SSH utilizzando il proxy locale sul lato di origine.

Per ulteriori informazioni su come utilizzare per Console di gestione AWS aprire un tunnel e avviare una sessione SSH, vedereApertura di un tunnel e avvio di una sessione SSH su un dispositivo remoto.

Il seguente video descrive come funziona il tunneling sicuro e ti guiderà attraverso il processo di configurazione di una sessione SSH su un dispositivo Raspberry Pi.

Ciclo di vita sicuro del tunnel

I tunnel possono avere lo stato `OPEN` o `CLOSED`. Le connessioni al tunnel possono avere lo stato `CONNECTED` o `DISCONNECTED`. Di seguito viene illustrato il funzionamento dei diversi stati del tunnel e della connessione.

1. Quando si apre un tunnel, ha uno stato di `OPEN`. Lo stato della connessione di origine e destinazione del tunnel è impostato su `DISCONNECTED`.
2. Quando un dispositivo (di origine o destinazione) si connette al tunnel, lo stato della connessione corrispondente cambia in `CONNECTED`.
3. Quando un dispositivo si disconnette dal tunnel mentre lo stato della connessione rimane `OPEN`, lo stato della connessione corrispondente torna a `DISCONNECTED`. Un dispositivo può collegarsi e scollegarsi da un tunnel ripetutamente finché il tunnel rimane `OPEN`.

> **Note**
>
> I token di accesso client (CAT) possono essere utilizzati una sola volta per connettersi a un tunnel. Per riconnetterti al tunnel, ruota i token di accesso del client utilizzando l'operazione RotateTunnelAccessTokenAPI o il comando rotate-tunnel-access-tokenCLI. Per alcuni

esempi, consulta Risoluzione dei problemi di connettività del tunneling AWS IoT sicuro mediante la rotazione dei token di accesso del client.

4. Quando chiami `CloseTunnel` o il tunnel rimane `OPEN` per più tempo del valore `MaxLifetimeTimeout`, lo stato del tunnel diventa `CLOSED`. È possibile configurare `MaxLifetimeTimeout` quando si chiama `OpenTunnel`. `MaxLifetimeTimeout` viene impostato su 12 ore se non si specifica un valore.

> **Note**
>
> Un tunnel non può essere riaperto quando lo stato è `CLOSED`.

5. Mentre il tunnel è visibile, puoi chiamare `DescribeTunnel` e `ListTunnels` per visualizzare i metadati del tunnel. Il tunnel può essere visibile nella AWS IoT console per almeno tre ore prima di essere eliminato.

AWS IoT tutorial di tunneling sicuro

AWS IoT il tunneling sicuro aiuta i clienti a stabilire una comunicazione bidirezionale con i dispositivi remoti protetti da un firewall tramite una connessione sicura gestita da. AWS IoT

Per una dimostrazione del tunneling AWS IoT sicuro, usa la nostra demo di tunneling sicuro su.AWS IoT GitHub

I tutorial seguenti ti aiutano a imparare a utilizzare il tunneling sicuro. Imparerai a:

1. Creare un tunnel sicuro utilizzando i metodi di configurazione rapida e manuale per accedere al dispositivo remoto.
2. Configurare il proxy locale quando si utilizza il metodo di configurazione manuale e connettersi al tunnel per accedere al dispositivo di destinazione.
3. Accedere tramite SSH al dispositivo remoto da un browser senza dover configurare il proxy locale.
4. Converti un tunnel creato utilizzando AWS CLI o utilizzando il metodo di configurazione manuale per utilizzare il metodo di configurazione rapida.

Tutorial in questa sezione

I tutorial in questa sezione si concentrano sulla creazione di un tunnel utilizzando Console di gestione AWS e l' AWS IoT API Reference. Nella AWS IoT console, puoi creare un tunnel dalla pagina dell'hub Tunnels o dalla pagina dei dettagli di un oggetto che hai creato. Per ulteriori informazioni, consulta Metodi di creazione di tunnel nella AWS IoT console.

Di seguito sono illustrati i tutorial in questa sezione:

- Apertura di un tunnel e utilizzo di SSH basato su browser per accedere al dispositivo remoto

  In questo tutorial viene illustrato come aprire un tunnel dalla pagina Hub dei tunnel mediante il metodo di configurazione rapida. Imparerai anche come utilizzare SSH basato su browser per accedere al dispositivo remoto utilizzando un'interfaccia a riga di comando contestuale all'interno della console. AWS IoT

- Apertura di un tunnel utilizzando la configurazione manuale e connessione al dispositivo remoto

  In questo tutorial viene illustrato come aprire un tunnel dalla pagina Hub dei tunnel mediante il metodo di configurazione manuale. Verrà inoltre descritto come configurare e avviare il proxy locale da un terminale nel dispositivo di origine e connettersi al tunnel.

- Apertura di un tunnel per dispositivo remoto e utilizzo di SSH basato su browser

  In questo tutorial viene illustrato come aprire un tunnel dalla pagina dei dettagli di un oggetto creato in precedenza. Verrà descritto come creare un nuovo tunnel e utilizzare un tunnel esistente. Il tunnel esistente corrisponde al tunnel aperto più recente che è stato creato per il dispositivo. È anche possibile utilizzare l'SSH basato su browser per accedere al dispositivo remoto.

AWS IoT tutorial di tunneling sicuro

- Apertura di un tunnel e avvio di una sessione SSH su un dispositivo remoto
- Apertura di un tunnel per dispositivo remoto e utilizzo di SSH basato su browser

Apertura di un tunnel e avvio di una sessione SSH su un dispositivo remoto

In questi tutorial, verrà descritto come accedere in remoto a un dispositivo protetto da un firewall. Non è possibile avviare una sessione SSH diretta nel dispositivo perché il firewall blocca tutto il traffico in

entrata. Nei tutorial viene illustrato come aprire un tunnel e utilizzarlo per avviare una sessione SSH a un dispositivo remoto.

Prerequisiti per i tutorial

I prerequisiti per l'esecuzione del tutorial possono variare a seconda che si utilizzino i metodi di configurazione manuale o rapida per aprire un tunnel e accedere al dispositivo remoto.

> **Note**
>
> Per entrambi i metodi di configurazione, è necessario consentire il traffico in uscita sulla porta 443.

- Per informazioni sui prerequisiti per il tutorial sul metodo di configurazione rapida, consulta Prerequisiti per il metodo di configurazione rapida.
- Per informazioni sui prerequisiti per il tutorial sul metodo di configurazione manuale, consulta Prerequisiti per il metodo di configurazione manuale. Se si utilizza questo metodo di configurazione, è necessario configurare il proxy locale sul dispositivo di origine. Per scaricare il codice sorgente del proxy locale, consulta Local proxy reference implementation su. GitHub

Metodi di configurazione del tunnel

In questi tutorial, verranno fornite informazioni sui metodi di configurazione manuale e rapida per aprire un tunnel e connettersi al dispositivo remoto. Nella tabella seguente viene illustrata la differenza tra i metodi di configurazione. Dopo aver creato il tunnel, è possibile utilizzare un'interfaccia a riga di comando integrata nel browser per effettuare l'accesso SSH al dispositivo remoto. Se i token vengono smarriti o il tunnel si disconnette, è possibile inviare nuovi token di accesso per riconnettersi al tunnel.

Metodi di configurazione rapida e manuale

Criteri	Configurazione rapida	Configurazione manuale
Creazione del tunnel	Creare un nuovo tunnel con configura zioni predefinite, modificabili. Per accedere al dispositivo remoto, è	Creare un tunnel specificando manualmen te le configurazioni del tunnel. È possibile utilizzare questo metodo per connettersi al

Criteri	Configurazione rapida	Configurazione manuale
	possibile utilizzare solo SSH come servizio di destinazione.	dispositivo remoto utilizzando servizi diversi da SSH.
Token di accesso	Il token di accesso di destinazione verrà inviato automaticamente al dispositivo nell'argomento MQTT riservato, se un nome oggetto viene specificato durante la creazione del tunnel. Non è necessario scaricare o gestire il token sul dispositivo di origine.	È necessario scaricare o gestire manualmente il token sul dispositivo di origine. Il token di accesso di destinazione viene inviato automaticamente al dispositivo remoto nell'argomento MQTT riservato, se un nome oggetto viene specificato durante la creazione del tunnel.
Proxy locale	Un proxy locale basato sul Web viene configurato automaticamente per l'interazione con il dispositivo. Non è necessario configurare manualmente il proxy locale.	È necessario configurare manualmente e avviare il proxy locale. Per configurare il proxy locale, puoi utilizzare il AWS IoT Device Client o scaricare l'implementazione di riferimento del proxy locale su GitHub.

Metodi di creazione di tunnel nella AWS IoT console

I tutorial in questa sezione mostrano come creare un tunnel utilizzando Console di gestione AWS e l'OpenTunnelAPI. Se si configura la destinazione durante la creazione di un tunnel, il tunneling AWS IoT sicuro invia il token di accesso del client di destinazione al dispositivo remoto tramite MQTT e l'argomento MQTT riservato,). `$aws/things/RemoteDeviceA/tunnels/notify` Alla ricezione del messaggio MQTT, l'agente IoT sul dispositivo remoto avvia il proxy locale in modalità di destinazione. Per ulteriori informazioni, consulta Argomenti riservati.

> **Note**
>
> Puoi omettere la configurazione di destinazione se desideri consegnare il token di accesso client di destinazione al dispositivo remoto tramite un altro metodo. Per ulteriori informazioni, consulta Configurazione di un dispositivo remoto e utilizzo dell'agente IoT.

Nella AWS IoT console, è possibile creare un tunnel utilizzando uno dei seguenti metodi. Per informazioni sui tutorial che illustrano come creare un tunnel utilizzando questi metodi, consulta Tutorial in questa sezione.

- Hub dei tunnel

  Quando si crea il tunnel, è possibile specificare se utilizzare i metodi di configurazione rapida o manuale per crearlo e fornire i dettagli di configurazione del tunnel opzionali. Questi includono anche il nome del dispositivo di destinazione e il servizio che si desidera utilizzare per la connessione al dispositivo. Dopo aver creato un tunnel, è possibile utilizzare SSH all'interno del browser o aprire un terminale all'esterno della AWS IoT console per accedere al dispositivo remoto.
- Pagina dei dettagli dell'oggetto

  Quando si crea il tunnel, sarà anche possibile specificare se utilizzare il tunnel aperto più recente o creare un nuovo tunnel per il dispositivo, oltre a selezionare i metodi di configurazione e fornire i dettagli di configurazione del tunnel opzionali. Non è possibile modificare i dettagli di configurazione di un tunnel esistente. È possibile utilizzare il metodo di configurazione rapida per ruotare i token di accesso ed effettuare l'accesso SSH al dispositivo remoto all'interno del browser. Per aprire un tunnel utilizzando questo metodo, è necessario aver creato un oggetto IoT (ad esempio`RemoteDeviceA`) nel AWS IoT registro. Per ulteriori informazioni, consulta Registrare un dispositivo nel AWS IoT registro.

Tutorial in questa sezione

- Apertura di un tunnel e utilizzo di SSH basato su browser per accedere al dispositivo remoto
- Apertura di un tunnel utilizzando la configurazione manuale e connessione al dispositivo remoto

Apertura di un tunnel e utilizzo di SSH basato su browser per accedere al dispositivo remoto

È possibile utilizzare il metodo di configurazione rapida o manuale per creare un tunnel. In questo tutorial viene illustrato come aprire un tunnel utilizzando il metodo di configurazione rapida e utilizzare l'SSH basato su browser per connettersi al dispositivo remoto. Per un esempio che mostra come aprire un tunnel utilizzando il metodo di configurazione manuale, consulta Apertura di un tunnel utilizzando la configurazione manuale e connessione al dispositivo remoto.

Utilizzando il metodo di configurazione rapida, è possibile creare un nuovo tunnel con configurazioni predefinite che è possibile modificare. Un proxy locale basato sul Web viene configurato

automaticamente e il token di accesso viene distribuito automaticamente al dispositivo di destinazione remoto mediante MQTT. Dopo aver creato un tunnel, è possibile iniziare a interagire con il dispositivo remoto utilizzando un'interfaccia a riga di comando all'interno della console.

Con il metodo di configurazione rapida, è necessario utilizzare SSH come il servizio di destinazione per accedere al dispositivo remoto. Per ulteriori informazioni sui diversi metodi di configurazione, consulta Metodi di configurazione del tunnel.

Prerequisiti per il metodo di configurazione rapida

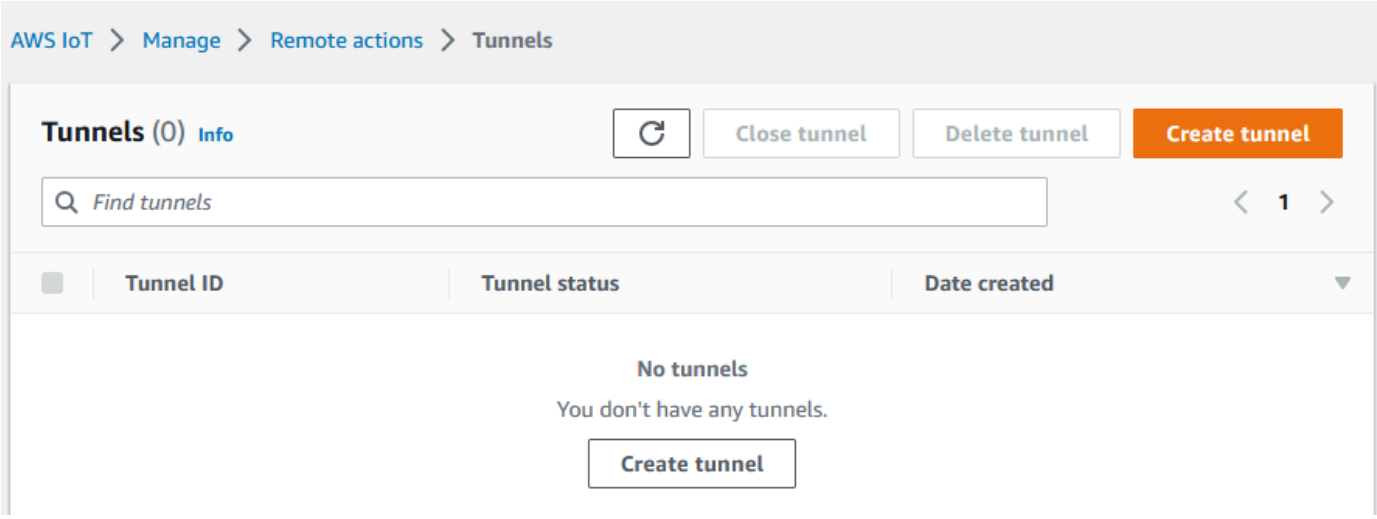
- I firewall dietro cui si trova il dispositivo remoto devono consentire il traffico in uscita sulla porta 443. Il tunnel creato utilizzerà questa porta per connettersi al dispositivo remoto.
- Sul dispositivo remoto è in esecuzione un agente per dispositivi IoT (vediSnippet dell'agente IoT) che si connette al gateway del AWS IoT dispositivo ed è configurato con un abbonamento tematico MQTT. Per ulteriori informazioni, consulta Connettere un dispositivo al gateway del AWS IoT dispositivo.
- È necessario disporre di un daemon SSH in esecuzione sul dispositivo remoto.

Apertura di un tunnel

È possibile aprire un tunnel sicuro utilizzando il Console di gestione AWS, l' AWS IoT API Reference o il AWS CLI. Facoltativamente, è possibile configurare un nome di destinazione, ma non è richiesto per questo tutorial. Se si configura la destinazione, il tunneling sicuro distribuirà automaticamente il token di accesso al dispositivo remoto tramite MQTT. Per ulteriori informazioni, consulta Metodi di creazione di tunnel nella AWS IoT console.

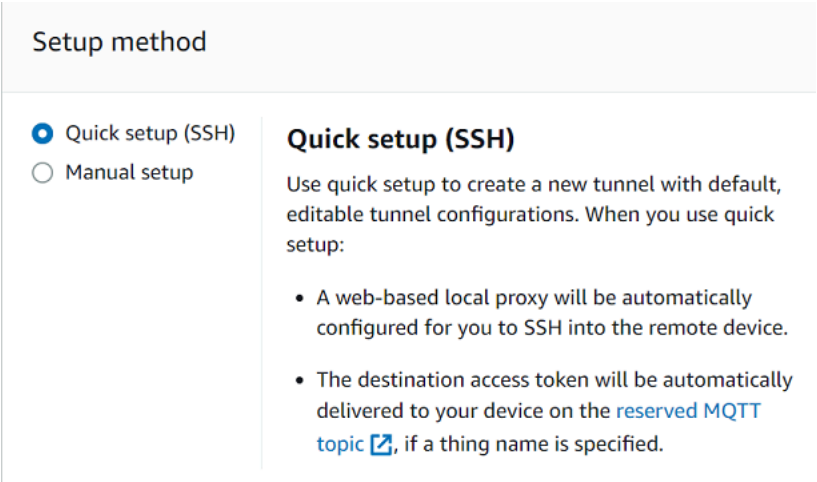
Per aprire un tunnel mediante la console

1. Vai a Hub dei tunnel della console AWS IoT e scegli Create tunnel (Crea tunnel).



2. Per questo tutorial, scegli Quick setup (Configurazione rapida) come metodo di creazione del tunnel, quindi seleziona Next (Avanti).

> **Note**
>
> Se si crea un tunnel sicuro dalla pagina dei dettagli di un oggetto creato in precedenza, è possibile scegliere se creare un nuovo tunnel o utilizzarne uno esistente. Per ulteriori informazioni, consulta Apertura di un tunnel per dispositivo remoto e utilizzo di SSH basato su browser.

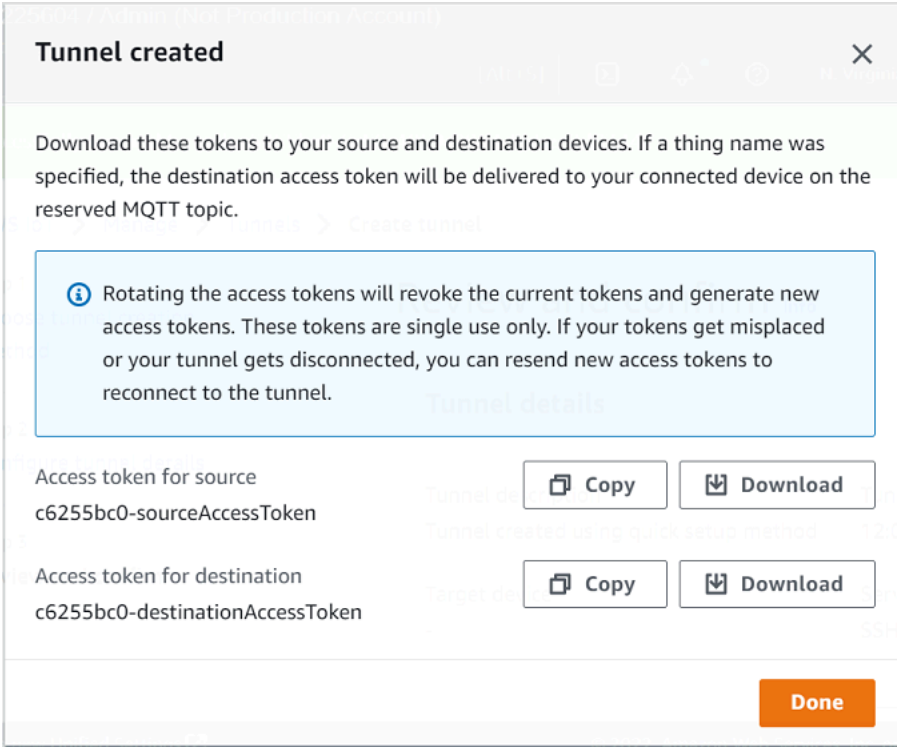


3. Rivedi e conferma i dettagli di configurazione del tunnel. Per creare un tunnel, scegli Confirm and create (Conferma e crea). Se desideri modificare questi dettagli, scegli Previous (Indietro) per tornare alla pagina precedente, quindi conferma e crea il tunnel.

> **Note**
>
> Quando si utilizza la configurazione rapida, il nome del servizio non può essere modificato. È necessario utilizzare SSH come Service (Servizio).

4. Per creare il tunnel, scegli Done (Fine).

   Per questo tutorial, non è necessario scaricare i token di accesso di origine o destinazione. Questi token possono essere utilizzati una sola volta per connettersi al tunnel. Se il tunnel si disconnette, è possibile generare e inviare nuovi token al dispositivo remoto per riconnettersi al tunnel. Per ulteriori informazioni, consulta Nuovo invio dei token di accesso al tunnel.



Per aprire un tunnel mediante l'API

Per aprire un nuovo tunnel, puoi utilizzare l'operazione API. OpenTunnel

> **Note**
>
> È possibile creare un tunnel utilizzando il metodo di configurazione rapida solo dalla console AWS IoT . Quando utilizzi l' AWS IoT API Reference API o il AWS CLI, utilizzerà il metodo di configurazione manuale. È possibile aprire il tunnel esistente creato in precedenza e quindi

modificare il metodo di configurazione del tunnel per utilizzare la configurazione rapida. Per ulteriori informazioni, consulta Apertura di un tunnel esistente e utilizzo di SSH basato su browser.

Di seguito viene illustrato un esempio di come eseguire questa operazione API. Facoltativamente, se si desidera specificare il nome dell'oggetto e il servizio di destinazione, utilizzare il parametro `DestinationConfig`. Per un esempio che mostra come utilizzare questo parametro, consulta Apertura di un nuovo tunnel per il dispositivo remoto.

```
aws iotsecuretunneling open-tunnel
```

L'esecuzione di questo comando crea un nuovo tunnel e fornisce i token di accesso di origine e destinazione.

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

Utilizzo di SSH basato su browser

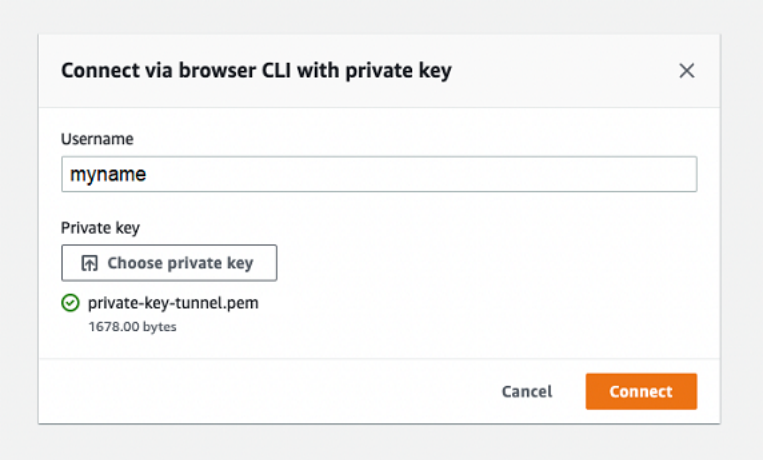
Dopo aver creato un tunnel utilizzando il metodo di configurazione rapida e dopo che il dispositivo di destinazione si è connesso al tunnel, è possibile accedere al dispositivo remoto utilizzando un SSH basato su browser. Utilizzando l'SSH basato su browser, è possibile comunicare direttamente con il dispositivo remoto inserendo i comandi in un'interfaccia a riga di comando contestuale all'interno della console. Questa funzionalità semplifica l'interazione con il dispositivo remoto perché non è necessario aprire un terminale all'esterno della console o configurare il proxy locale.

Per utilizzare l'SSH basato su browser

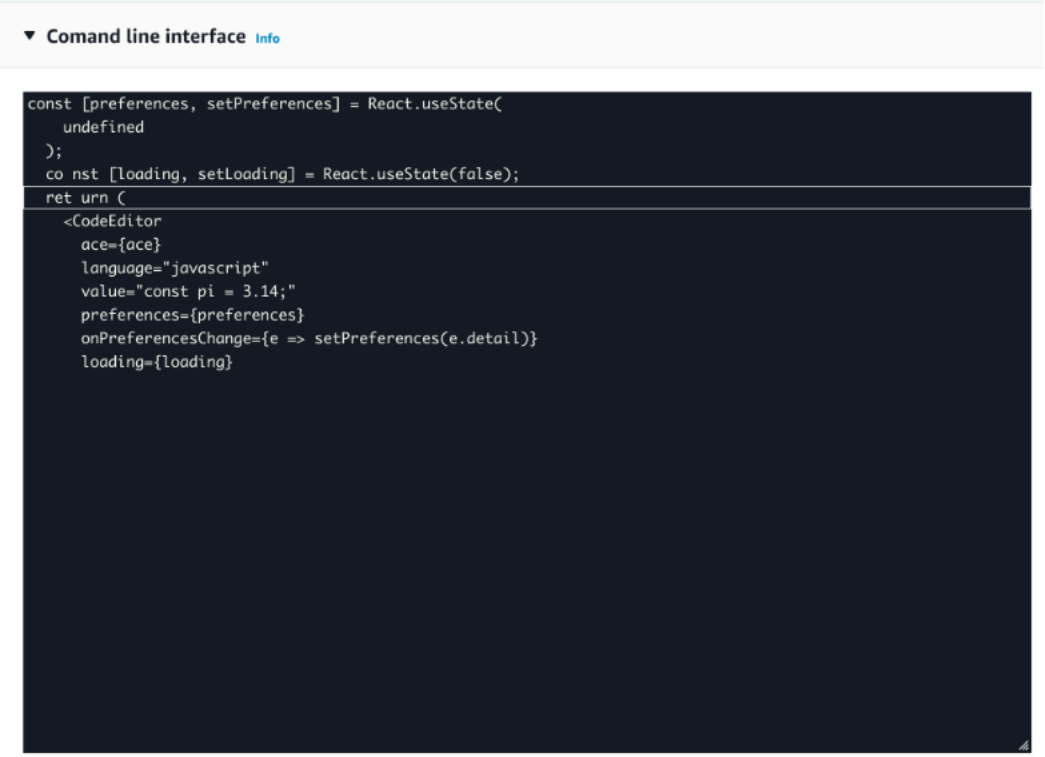
1. Passa a Hub dei tunnel della console AWS IoT e scegli il tunnel creato per visualizzare i relativi dettagli.
2. Espandi la sezione Secure Shell (SSH) e scegli Connect (Connetti).
3. Scegli se eseguire l'autenticazione alla connessione SSH fornendo nome utente e password oppure, per un'autenticazione più sicura, utilizzare la chiave privata del tuo dispositivo. Se esegui

l'autenticazione utilizzando la chiave privata, tieni presente che solo le chiavi RSA in formato PEM (256 e 512) funzionano con la console SSH. AWS IoT Secure Tunneling

- Per connetterti utilizzando il nome utente e la password, scegli Use password (Usa password). Puoi quindi inserire nome utente e password e iniziare a utilizzare la CLI integrata nel browser.
- Per connetterti utilizzando la chiave privata del dispositivo di destinazione, scegli Use private key (Usa chiave privata). Specifica il nome utente e carica il file della chiave privata del dispositivo, quindi scegli Connect (Connetti) per iniziare a utilizzare la CLI integrata nel browser.



Dopo aver effettuato l'autenticazione alla connessione SSH, puoi iniziare rapidamente a inserire i comandi e interagire con il dispositivo utilizzando la CLI del browser, poiché il proxy locale è già stato configurato automaticamente.



Se la CLI del browser rimane aperta dopo la durata del tunnel, potrebbe scadere, causando la disconnessione dell'interfaccia a riga di comando. Puoi duplicare il tunnel e avviare un'altra sessione per interagire con il dispositivo remoto all'interno della console stessa.

Risoluzione dei problemi relativi all'utilizzo di SSH basato su browser

Di seguito viene illustrato come risolvere alcuni problemi che si potrebbero verificare quando si utilizza l'SSH basato su browser.

- Al posto dell'interfaccia a riga di comando viene visualizzato un errore

  È possibile che l'errore venga visualizzato perché il dispositivo di destinazione si è disconnesso. È possibile scegliere Generate new access tokens (Genera nuovi token di accesso) per generare nuovi token di accesso e inviarli al dispositivo remoto utilizzando MQTT. I nuovi token possono essere usati per riconnettersi al tunnel. La riconnessione al tunnel cancella la cronologia e aggiorna la sessione della riga di comando.
- Un errore di disconnessione del tunnel viene visualizzato durante l'autenticazione tramite chiave privata

  Questo errore potrebbe essere visualizzato perché è possibile che la chiave privata non sia stata accettata dal dispositivo di destinazione. Per risolvere questo errore, controllare il file della chiave

privata caricata per l'autenticazione. Se l'errore persiste, controllare i log del dispositivo. È anche possibile provare a riconnettersi al tunnel inviando nuovi token di accesso al dispositivo remoto.

- Il tunnel è stato chiuso durante l'utilizzo della sessione

  Se il tunnel è stato chiuso perché è rimasto aperto per un periodo di tempo superiore alla durata specificata, è possibile che la sessione della riga di comando venga disconnessa. Un tunnel non può essere riaperto dopo che è stato chiuso. Per riconnettersi, è necessario aprire un altro tunnel al dispositivo.

  È possibile duplicare un tunnel per creare un nuovo tunnel con le stesse configurazioni del tunnel chiuso. È possibile duplicare un tunnel chiuso dalla console. AWS IoT Per duplicare il tunnel, scegliere il tunnel che è stato chiuso per visualizzare i relativi dettagli, quindi selezionare Duplicate tunnel (Duplica tunnel). Specificare la durata del tunnel che si desidera utilizzare, quindi creare il nuovo tunnel.

Pulizia

- Chiusura del tunnel

  Al termine dell'utilizzo, si consiglia di chiudere il tunnel. Un tunnel può chiudersi anche se è rimasto aperto per un periodo di tempo superiore alla durata del tunnel specificata. Un tunnel non può essere riaperto dopo che è stato chiuso. È ancora possibile duplicare un tunnel scegliendo il tunnel chiuso e quindi selezionando Duplicate tunnel (Duplica tunnel). Specificare la durata del tunnel che si desidera utilizzare, quindi creare il nuovo tunnel.

  - Per chiudere un singolo tunnel o più tunnel dalla console AWS IoT , passa all'Hub dei tunnel, scegli i tunnel che desideri chiudere, quindi seleziona Close tunnel (Chiudi tunnel).
  - Per chiudere uno o più tunnel utilizzando l' AWS IoT API Reference API, utilizza l'CloseTunnelAPI.

    ```
    aws iotsecuretunneling close-tunnel \
        --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    ```

- Eliminazione del tunnel

  Puoi eliminare definitivamente un tunnel dal tuo Account AWS.

> **Warning**
>
> Le operazioni di eliminazione sono permanenti e non possono essere annullate.

- Per eliminare un singolo tunnel o più tunnel dalla console AWS IoT , passa all'Hub dei tunnel, scegli i tunnel che desideri eliminare, quindi seleziona Delete tunnel (Elimina tunnel).
- Per eliminare uno o più tunnel utilizzando l' AWS IoT API Reference API, utilizza l'CloseTunnelAPI. Quando utilizzi l'API, imposta il flag `delete` su `true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Apertura di un tunnel utilizzando la configurazione manuale e connessione al dispositivo remoto

Quando si apre un tunnel, è possibile scegliere il metodo di configurazione rapida o manuale per aprire un tunnel nel dispositivo remoto. In questo tutorial viene illustrato come aprire un tunnel utilizzando il metodo di configurazione manuale e configurare e avviare il proxy locale per connettersi al dispositivo remoto.

Quando si utilizza il metodo di configurazione manuale, è necessario specificare manualmente le configurazioni del tunnel durante la creazione. Dopo aver creato il tunnel, puoi usare SSH all'interno del browser o aprire un terminale all'esterno della AWS IoT console. In questo tutorial viene illustrato come utilizzare il terminale esterno alla console per accedere al dispositivo remoto. Verrà inoltre descritto come configurare il proxy locale e connettersi ad esso per interagire con il dispositivo remoto. Per connettersi al proxy locale, è necessario scaricare il token di accesso di origine durante la creazione del tunnel.

Con questo metodo di configurazione, è possibile utilizzare servizi diversi da SSH, come FTP, per connettersi al dispositivo remoto. Per ulteriori informazioni sui diversi metodi di configurazione, consulta Metodi di configurazione del tunnel.

Prerequisiti per il metodo di configurazione manuale

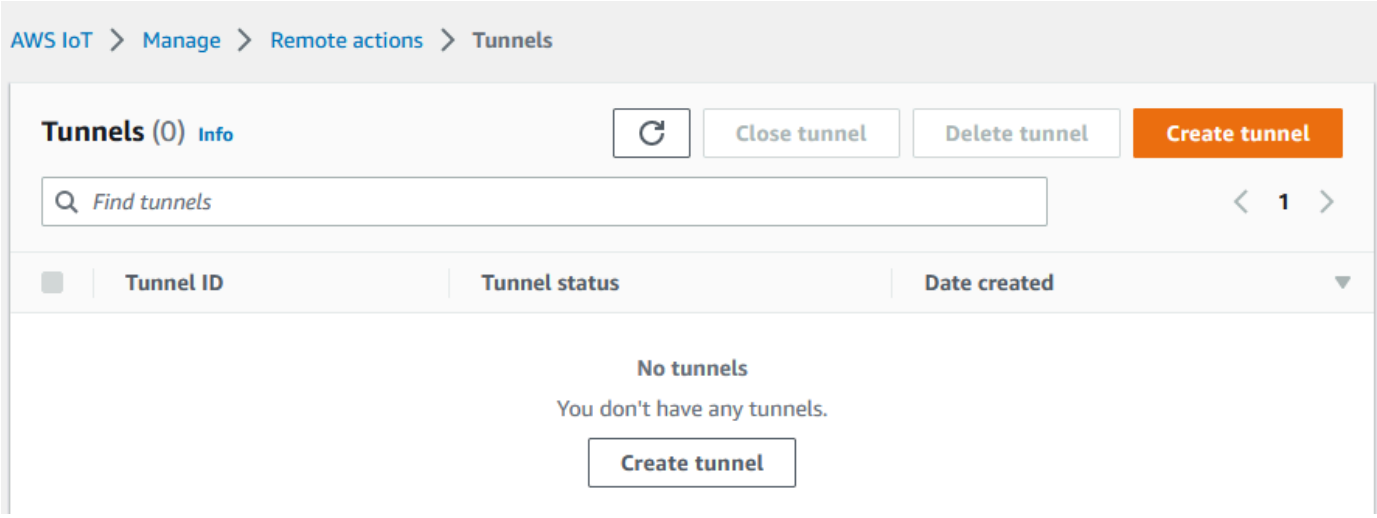
- I firewall dietro cui si trova il dispositivo remoto devono consentire il traffico in uscita sulla porta 443. Il tunnel creato utilizzerà questa porta per connettersi al dispositivo remoto.
- Sul dispositivo remoto è in esecuzione un agente per dispositivi IoT (vediSnippet dell'agente IoT) che si connette al gateway del AWS IoT dispositivo ed è configurato con un abbonamento tematico MQTT. Per ulteriori informazioni, consulta Connettere un dispositivo al gateway del AWS IoT dispositivo.
- È necessario disporre di un daemon SSH in esecuzione sul dispositivo remoto.
- Hai scaricato il codice sorgente del proxy locale da GitHube lo hai creato per la piattaforma di tua scelta. Faremo riferimento al file eseguibile proxy locale costruito come `localproxy` in questo tutorial.

Apertura di un tunnel

Puoi aprire un tunnel sicuro utilizzando Console di gestione AWS, AWS IoT API Reference o AWS CLI. Facoltativamente, è possibile configurare un nome di destinazione, ma non è richiesto per questo tutorial. Se si configura la destinazione, il tunneling sicuro distribuirà automaticamente il token di accesso al dispositivo remoto tramite MQTT. Per ulteriori informazioni, consulta Metodi di creazione di tunnel nella AWS IoT console.

Per aprire un tunnel tramite la console

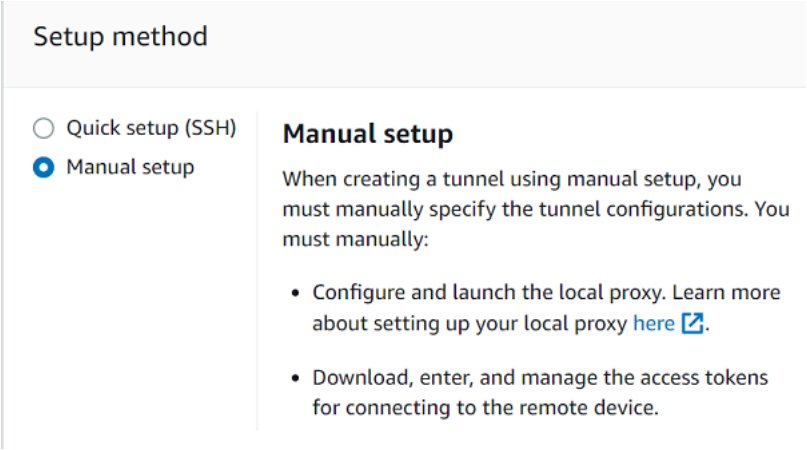
1. Vai a Hub dei tunnel della console AWS IoT e scegli Create tunnel (Crea tunnel).



2. Per questo tutorial, scegli Manual setup (Configurazione manuale) come metodo di creazione del tunnel, quindi seleziona Next (Avanti). Per informazioni sull'utilizzo del metodo Quick setup

(Configurazione rapida) per creare un tunnel, consulta Apertura di un tunnel e utilizzo di SSH basato su browser per accedere al dispositivo remoto.

> **Note**
>
> Se si crea un tunnel sicuro dalla pagina dei dettagli di un oggetto, è possibile scegliere se creare un nuovo tunnel o utilizzarne uno esistente. Per ulteriori informazioni, consulta Apertura di un tunnel per dispositivo remoto e utilizzo di SSH basato su browser.



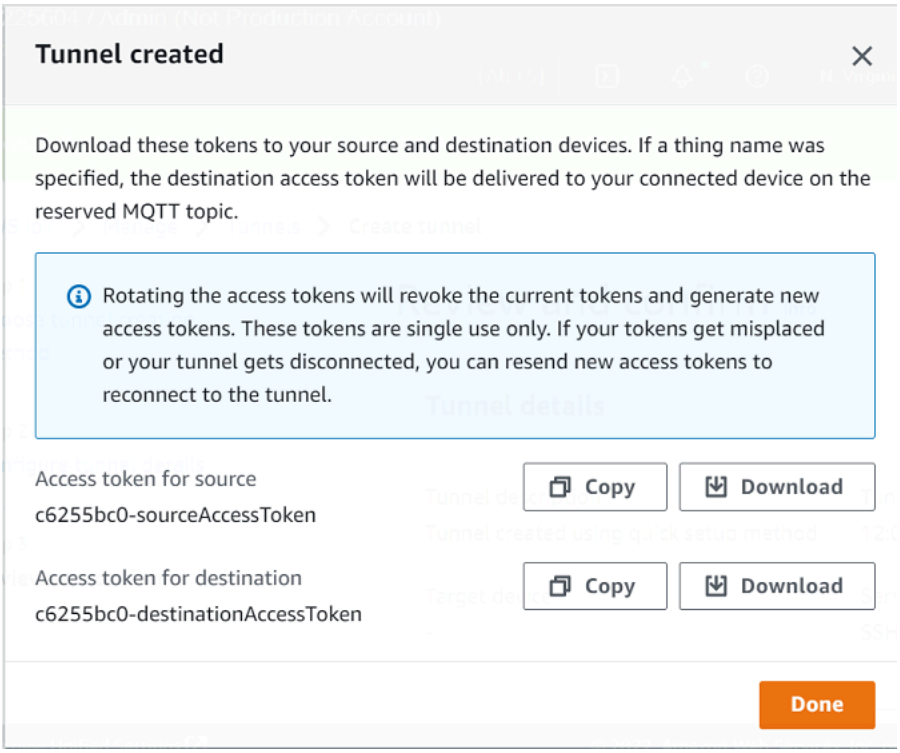
3. (Facoltativo) Immetti le impostazioni di configurazione per il tunnel. Puoi anche ignorare questo passaggio e andare a quello successivo per creare un tunnel.

   Immetti una descrizione del tunnel, la durata del timeout del tunnel e i tag delle risorse come coppie chiave-valore per facilitare l'identificazione della risorsa. Per questo tutorial, puoi ignorare la configurazione della destinazione.

   > **Note**
   >
   > Non ti verrà addebitato alcun costo in base alla durata di apertura di un tunnel.
   > Ti vengono addebitati dei costi solo durante la creazione di un nuovo tunnel. Per informazioni sui prezzi, consulta Tunneling sicuro in Prezzi di AWS IoT Device Management.

4. Scarica i token di accesso client e scegli Done (Fine). I token non saranno disponibili per il download dopo aver scelto Done (Fine).

Questi token possono essere utilizzati una sola volta per connettersi al tunnel. Se i token vengono persi o il tunnel si disconnette, è possibile generare e inviare nuovi token al dispositivo remoto per riconnettersi al tunnel.



Per aprire un tunnel mediante l'API

Per aprire un nuovo tunnel, puoi utilizzare l'operazione API. OpenTunnel Utilizzando l'API è anche possibile specificare configurazioni aggiuntive, come la durata del tunnel e la configurazione della destinazione.

```
aws iotsecuretunneling open-tunnel \
    --region us-east-1 \
    --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
```

L'esecuzione di questo comando crea un nuovo tunnel e fornisce i token di accesso di origine e destinazione.

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
```

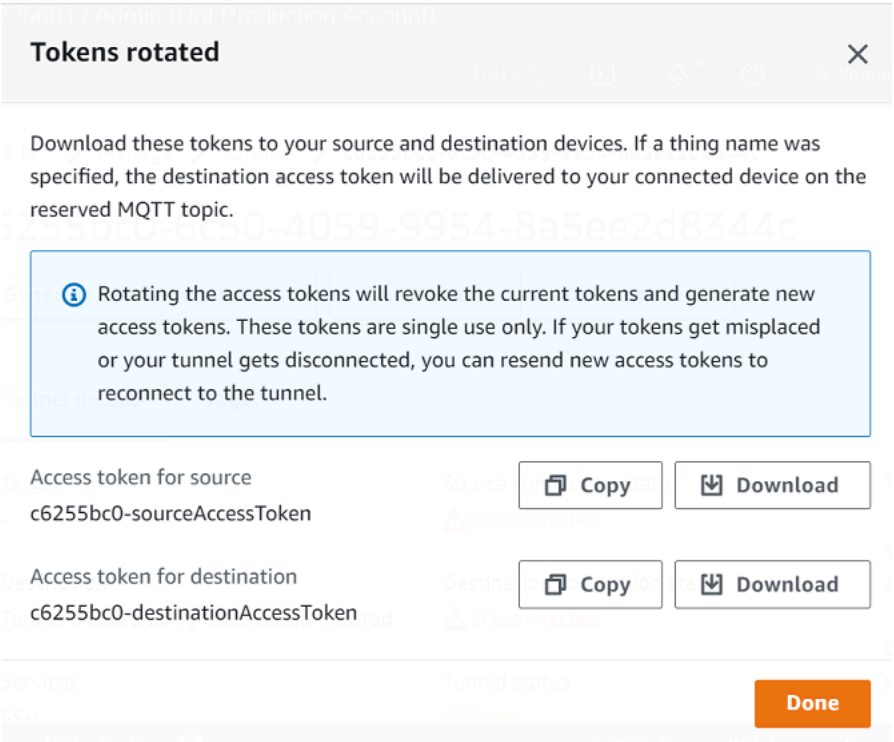
```
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

Nuovo invio dei token di accesso al tunnel

I token ottenuti durante la creazione di un tunnel possono essere utilizzati una sola volta per connettersi al tunnel. Se si smarrisce il token di accesso o il tunnel si disconnette, è possibile inviare nuovamente nuovi token di accesso al dispositivo remoto utilizzando MQTT senza costi aggiuntivi. AWS IoT il tunneling sicuro revocherà i token attuali e restituirà nuovi token di accesso per la riconnessione al tunnel.

Rotazione dei token dalla console

1. Vai all'hub Tunnels della AWS IoT console e scegli il tunnel che hai creato.
2. Nella pagina dei dettagli del tunnel, scegli Generate new access tokens (Genera nuovi token di accesso), quindi scegli Next (Avanti).
3. Scarica i nuovi token di accesso per il tunnel e scegli Done (Fine). Questi token possono essere utilizzati una sola volta. Se smarrisci questi token o il tunnel si disconnette, puoi inviare nuovi token di accesso.



Rotazione dei token di accesso mediante l'API

Per ruotare i token di accesso al tunnel, puoi utilizzare l'operazione RotateTunnelAccessTokenAPI per revocare i token correnti e restituire nuovi token di accesso per la riconnessione al tunnel. Ad esempio, il comando seguente ruota i token di accesso per il dispositivo di destinazione, *RemoteThing1*.

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --tunnel-id <tunnel-id> \
    --client-mode DESTINATION \
    --destination-config thingName=<RemoteThing1>,services=SSH \
    --region <region>
```

L'esecuzione di questo comando genera il nuovo token di accesso come mostrato nel seguente esempio. Il token viene quindi consegnato al dispositivo mediante MQTT per connettersi al tunnel, se l'agente di dispositivo è configurato correttamente.

```
{
    "destinationAccessToken": "destination-access-token",
    "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

Per esempi che mostrano come e quando ruotare i token di accesso, consulta Risoluzione dei problemi di connettività del tunneling AWS IoT sicuro mediante la rotazione dei token di accesso del client.

Configurazione e avvio del proxy locale

Per connettersi al dispositivo remoto, aprire un terminale sul laptop, quindi configurare e avviare il proxy locale. Il proxy locale trasmette i dati inviati dall'applicazione in esecuzione sul dispositivo di origine utilizzando il tunneling sicuro su una connessione sicura. WebSocket È possibile scaricare il sorgente proxy locale da. GitHub

Dopo aver configurato il proxy locale, copiare il token di accesso al client di origine e utilizzarlo per avviare il proxy locale in modalità di origine. Di seguito viene mostrato un comando di esempio per avviare il proxy locale. Nel comando seguente, il proxy locale è configurato per l'ascolto di nuove connessioni sulla porta 5555. In questo comando:

- `-r`specifica la Regione AWS, che deve essere la stessa regione in cui è stato creato il tunnel.
- `-s` specifica la porta a cui il proxy deve connettersi.

- `-t` specifica il testo del token client.

```
./localproxy -r us-east-1 -s 5555 -t source-client-access-token
```

L'esecuzione di questo comando avvierà il proxy locale in modalità di origine. Se si riceve il seguente errore dopo l'esecuzione del comando, impostare il percorso CA. Per informazioni, consulta Secure tunneling local proxy on. GitHub

```
Could not perform SSL handshake with proxy server: certificate verify failed
```

Di seguito è riportato un output di esempio dell'esecuzione del proxy locale nella modalità `source`.

```
...
...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-east-1.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-east-1.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...
```

```
Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Avvia una sessione SSH

Apri un altro terminale e utilizza il comando seguente per avviare una nuova sessione SSH collegandosi al proxy locale sulla porta 5555.

```
ssh username@localhost -p 5555
```

Potrebbe essere richiesta una password per la sessione SSH. Al termine della sessione SSH, digita **`exit`** per chiudere la sessione.

Pulizia

- Chiusura del tunnel

  Al termine dell'utilizzo, si consiglia di chiudere il tunnel. Un tunnel può chiudersi anche se è rimasto aperto per un periodo di tempo superiore alla durata del tunnel specificata. Un tunnel non può essere riaperto dopo che è stato chiuso. È ancora possibile duplicare un tunnel aprendo il tunnel chiuso e quindi selezionando Duplicate tunnel (Duplica tunnel). Specificare la durata del tunnel che si desidera utilizzare, quindi creare il nuovo tunnel.

  - Per chiudere un singolo tunnel o più tunnel dalla console AWS IoT , passa all'Hub dei tunnel, scegli i tunnel che desideri chiudere, quindi seleziona Close tunnel (Chiudi tunnel).
  - Per chiudere uno o più tunnel utilizzando l' AWS IoT API Reference API, utilizza l'CloseTunneloperazione API.

    ```
    aws iotsecuretunneling close-tunnel \
        --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    ```

- Eliminazione del tunnel

  Puoi eliminare definitivamente un tunnel dal tuo Account AWS.

  > **Warning**
  >
  > Le operazioni di eliminazione sono permanenti e non possono essere annullate.

- Per eliminare un singolo tunnel o più tunnel dalla console AWS IoT , passa all'Hub dei tunnel, scegli i tunnel che desideri eliminare, quindi seleziona Delete tunnel (Elimina tunnel).
- Per eliminare uno o più tunnel utilizzando l' AWS IoT API Reference API, utilizza l'operazione CloseTunnelAPI. Quando utilizzi l'API, imposta il flag `delete` su `true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Apertura di un tunnel per dispositivo remoto e utilizzo di SSH basato su browser

Dalla AWS IoT console, puoi creare un tunnel dall'hub Tunnels o dalla pagina dei dettagli di un oggetto IoT che hai creato. Quando si crea un tunnel dall'hub Tunnels (Tunnel), è possibile specificare se creare un tunnel utilizzando la configurazione rapida o manuale. Per un esempio di tutorial, consultare Apertura di un tunnel e avvio di una sessione SSH su un dispositivo remoto.

Quando crei un tunnel dalla pagina dei dettagli dell'oggetto della AWS IoT console, puoi anche specificare se creare un nuovo tunnel o aprire un tunnel esistente per quell'oggetto, come illustrato in questo tutorial. Se si sceglie un tunnel esistente, è possibile accedere al tunnel aperto più recente creato per questo dispositivo. È quindi possibile utilizzare l'interfaccia a riga di comando all'interno del terminale per accedere tramite SSH al dispositivo.

Prerequisiti

- I firewall dietro cui si trova il dispositivo remoto devono consentire il traffico in uscita sulla porta 443. Il tunnel creato utilizzerà questa porta per connettersi al dispositivo remoto.
- È stato creato un oggetto IoT (ad esempio`RemoteDevice1`) nel AWS IoT registro. Questo oggetto corrisponde alla rappresentazione del dispositivo remoto nel cloud. Per ulteriori informazioni, consulta la sezione relativa alla registrazione di un dispositivo nel registro AWS IoT.
- Sul dispositivo remoto è in esecuzione un agente per dispositivi IoT (vediSnippet dell'agente IoT) che si connette al gateway del AWS IoT dispositivo ed è configurato con un abbonamento tematico MQTT. Per ulteriori informazioni, consulta Connettere un dispositivo al gateway del AWS IoT dispositivo.
- È necessario disporre di un daemon SSH in esecuzione sul dispositivo remoto.

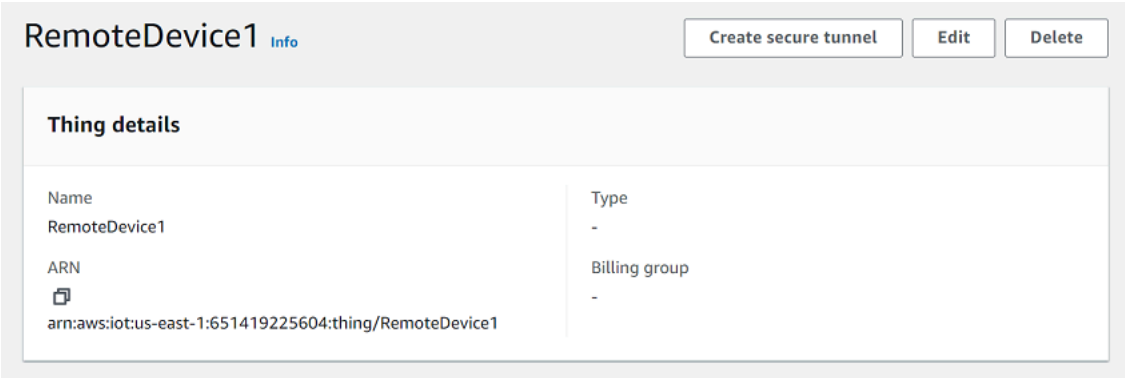
Apertura di un nuovo tunnel per il dispositivo remoto

Supponiamo di voler aprire un tunnel nel dispositivo remoto, `RemoteDevice1`. Innanzitutto, creiamo un oggetto IoT con il nome `RemoteDevice1` nel registro AWS IoT . È quindi possibile creare un tunnel utilizzando Console di gestione AWS l' AWS IoT API Reference API o il AWS CLI.

Configurando una destinazione durante la creazione di un tunnel, il servizio tunneling sicuro fornisce il token di accesso al client di destinazione al dispositivo remoto tramite MQTT e l'argomento MQTT riservato (`$aws/things/RemoteDeviceA/tunnels/notify`). Per ulteriori informazioni, consulta Metodi di creazione di tunnel nella AWS IoT console.

Per creare un tunnel per un dispositivo remoto dalla console

1. Scegli l'oggetto, `RemoteDevice1`, per visualizzare i relativi dettagli, quindi seleziona Create secure tunnel (Crea un tunnel sicuro).



2. Scegli se creare un nuovo tunnel o aprire un tunnel esistente. Per creare un nuovo tunnel, scegli Create new tunnel (Crea un nuovo tunnel). Puoi quindi scegliere se utilizzare il metodo di configurazione rapida o il metodo di configurazione manuale per creare il tunnel. Per ulteriori informazioni, consultare Apertura di un tunnel utilizzando la configurazione manuale e connessione al dispositivo remoto e Apertura di un tunnel e utilizzo di SSH basato su browser per accedere al dispositivo remoto.

Creazione di un tunnel per un dispositivo remoto mediante l'API

Per aprire un nuovo tunnel, puoi utilizzare l'operazione OpenTunnelAPI. Nel codice seguente viene illustrato un esempio di esecuzione di questo comando.

```
aws iotsecuretunneling open-tunnel \
    --region us-east-1 \
    --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
```

```
    --cli-input-json file://input.json
```

Di seguito viene mostrato il contenuto per il file `input.json`. È possibile utilizzare il parametro `destinationConfig` per specificare il nome del dispositivo di destinazione (ad esempio, *RemoteDevice1*) e il servizio che si desidera utilizzare per accedere al dispositivo di destinazione, ad esempio *SSH*. Facoltativamente, è anche possibile specificare parametri aggiuntivi come la descrizione del tunnel e i tag.

Contenuto di input.json

```
{
   "description": "Tunnel to remote device1",
   "destinationConfig": {
      "services": [ "SSH" ],
      "thingName": "RemoteDevice1"
   }
}
```

L'esecuzione di questo comando crea un nuovo tunnel e fornisce i token di accesso di origine e destinazione.

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

Apertura di un tunnel esistente e utilizzo di SSH basato su browser

Supponiamo che tu abbia creato il tunnel per il tuo dispositivo remotoRemoteDevice1, utilizzando il metodo di configurazione manuale o utilizzando l' AWS IoT API Reference API. È quindi possibile aprire il tunnel esistente per il dispositivo e scegliere Quick setup (Configurazione rapida) per utilizzare la funzionalità SSH basata sul browser. Le configurazioni di un tunnel esistente non possono essere modificate, pertanto non è possibile utilizzare il metodo di configurazione manuale.

Per utilizzare la funzionalità SSH basata su browser, non è necessario scaricare il token di accesso di origine o configurare il proxy locale. Un proxy locale basato sul Web verrà configurato automaticamente in modo da poter iniziare a interagire con il dispositivo remoto.

Per utilizzare il metodo di configurazione rapida e l'SSH basato su browser

1. Vai alla pagina dei dettagli dell'oggetto creato in precedenza, `RemoteDevice1`, e scegli Create secure tunnel (Crea un tunnel sicuro).
2. Scegli Use existing tunnel (Utilizza un tunnel esistente) per aprire il tunnel aperto più recente creato per il dispositivo remoto. Le configurazioni del tunnel non possono essere modificate, pertanto non è possibile utilizzare il metodo di configurazione manuale per il tunnel. Per utilizzare il metodo di configurazione rapida, scegli Quick setup (Configurazione rapida).
3. Continua per esaminare e confermare i dettagli di configurazione del tunnel e creare il tunnel. Le configurazioni del tunnel non possono essere modificate.

   Quando crei il tunnel, il tunneling sicuro utilizzerà l'operazione RotateTunnelAccessTokenAPI per revocare i token di accesso originali e generare nuovi token di accesso. Se il dispositivo remoto utilizza MQTT, questi token verranno automaticamente distribuiti al dispositivo remoto nell'argomento MQTT a cui è iscritto. Puoi anche scegliere di scaricare questi token manualmente sul dispositivo di origine.

Dopo aver creato il tunnel, puoi utilizzare l'SSH basato su browser per interagire con il dispositivo remoto direttamente dalla console utilizzando l'interfaccia a riga di comando contestuale. Per utilizzare questa interfaccia a riga di comando, scegli il tunnel per l'oggetto creato in precedenza e, nella pagina dei dettagli, espandi la sezione Command-line interface (Interfaccia a riga di comando). Poiché il proxy locale è già stato configurato automaticamente, puoi iniziare a inserire i comandi per iniziare rapidamente ad accedere e interagire con il tuo dispositivo remoto, `RemoteDevice1`.

Per ulteriori informazioni sul metodo di configurazione rapida e sull'utilizzo di SSH basato su browser, consulta Apertura di un tunnel e utilizzo di SSH basato su browser per accedere al dispositivo remoto.

Pulizia

- Chiusura del tunnel

  Al termine dell'utilizzo, si consiglia di chiudere il tunnel. Un tunnel può chiudersi anche se è rimasto aperto per un periodo di tempo superiore alla durata del tunnel specificata. Un tunnel non può essere riaperto dopo che è stato chiuso. È ancora possibile duplicare un tunnel aprendo il tunnel chiuso e quindi selezionando Duplicate tunnel (Duplica tunnel). Specificare la durata del tunnel che si desidera utilizzare, quindi creare il nuovo tunnel.

- Per chiudere un singolo tunnel o più tunnel dalla console AWS IoT , passa all'Hub dei tunnel, scegli i tunnel che desideri chiudere, quindi seleziona Close tunnel (Chiudi tunnel).
- Per chiudere uno o più tunnel utilizzando l' AWS IoT API Reference API, utilizza l'operazione API. CloseTunnel

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

- Eliminazione del tunnel

  Puoi eliminare definitivamente un tunnel dal tuo Account AWS.

> **Warning**
>
> Le operazioni di eliminazione sono permanenti e non possono essere annullate.

- Per eliminare un singolo tunnel o più tunnel dalla console AWS IoT , passa all'Hub dei tunnel, scegli i tunnel che desideri eliminare, quindi seleziona Delete tunnel (Elimina tunnel).
- Per eliminare uno o più tunnel utilizzando l' AWS IoT API Reference API, utilizza l'operazione CloseTunnelAPI. Quando utilizzi l'API, imposta il flag `delete` su `true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Proxy locale

Il proxy locale trasmette i dati inviati dall'applicazione in esecuzione sul dispositivo di origine utilizzando il tunneling sicuro su una WebSocket connessione sicura. È possibile scaricare il sorgente proxy locale da. GitHub

Il proxy locale può essere eseguito in due modalità: `source` o `destination`. In modalità di origine, il proxy locale viene eseguito sullo stesso dispositivo o rete dell'applicazione client che avvia la connessione TCP. In modalità di destinazione, il proxy locale viene eseguito sul dispositivo remoto, insieme all'applicazione di destinazione. Un singolo tunnel può supportare fino a tre flussi di dati alla volta utilizzando il multiplexing del tunnel. Per ogni flusso di dati, il tunneling sicuro utilizza più

connessioni TCP, il che riduce il rischio di timeout. Per ulteriori informazioni, consulta Flussi di dati multiplex e utilizzo di connessioni TCP simultanee in un tunnel sicuro.

Come utilizzare il proxy locale

Puoi eseguire il proxy locale sui dispositivi di origine e di destinazione per trasmettere i dati agli endpoint di tunneling sicuro. Se i tuoi dispositivi si trovano in una rete che utilizza un proxy web, il proxy web può intercettare le connessioni prima di inoltrarle a Internet. In questo caso, dovrai configurare il proxy locale per utilizzare il proxy Web. Per ulteriori informazioni, consulta Configurare il proxy locale per i dispositivi che utilizzano il proxy web.

Flusso di lavoro del proxy locale

I passaggi seguenti mostrano come avviene l'esecuzione del proxy locale sui dispositivi di origine e destinazione.

1. Connessione del proxy locale al tunneling sicuro

   Il proxy locale stabilisce innanzitutto una connessione al tunneling sicuro. Quando avvii il proxy locale, utilizza i seguenti argomenti:

   - L'`-r`argomento per specificare il modo Regione AWS in cui viene aperto il tunnel.
   - L'argomento `-t` per passare il token di accesso client di origine o di destinazione restituito da `OpenTunnel`.

   > **Note**
   >
   > Non è possibile connettere contemporaneamente due proxy locali che utilizzano lo stesso valore del token di accesso client.

2. Esecuzione di operazioni di origine o destinazione

   Una volta stabilita la WebSocket connessione, il proxy locale esegue azioni in modalità di origine o in modalità destinazione, a seconda della configurazione.

   Per impostazione predefinita, il proxy locale tenta di riconnettersi al tunneling sicuro in caso di errori input/output (I/O) o se la WebSocket connessione viene chiusa in modo imprevisto. Ciò causa la chiusura della connessione TCP. Se si verificano errori di socket TCP, il proxy locale invia un messaggio attraverso il tunnel per notificare all'altro lato di chiudere la connessione TCP. Per impostazione predefinita, il proxy locale utilizza sempre la comunicazione SSL.

3. Arresto del proxy locale

   Dopo aver utilizzato il tunnel, è possibile terminare il processo proxy locale in modo sicuro. Si consiglia di chiudere esplicitamente il tunnel chiamando `CloseTunnel`. I client di tunnel attivi potrebbero non essere chiusi subito dopo la chiamata. `CloseTunnel`

Per ulteriori informazioni su come utilizzare per Console di gestione AWS aprire un tunnel e avviare una sessione SSH, vedereApertura di un tunnel e avvio di una sessione SSH su un dispositivo remoto.

Best practice per i proxy locali

Quando esegui un proxy locale, attieniti alle best practice seguenti:

- Evitare l'uso dell'argomento proxy `-t` locale per passare un token di accesso. Si consiglia di utilizzare la variabile di ambiente `AWSIOT_TUNNEL_ACCESS_TOKEN` per impostare il token di accesso per il proxy locale.
- Eseguire l'eseguibile del proxy locale con privilegi minimi nel sistema operativo o nell'ambiente.
  - Evitare di eseguire il proxy locale come amministratore in Windows.
  - Evitare di eseguire il proxy locale come root su Linux e macOS.
- Si consiglia di eseguire il proxy locale su host separati, container, sandbox, chroot jail o un ambiente virtualizzato.
- Creare il proxy locale con i flag di sicurezza pertinenti, a seconda della toolchain.
- Nei dispositivi con più interfacce di rete, utilizzare l'argomento `-b` per associare il socket TCP all'interfaccia di rete utilizzata per comunicare con l'applicazione di destinazione.

Esempio di comando e output

Di seguito viene illustrato un esempio di comando eseguito su sistema operativo Linux e l'output corrispondente. L'esempio mostra come il proxy locale può essere configurato in entrambe le modalità `source` e `destination`. Il proxy locale aggiorna il protocollo HTTPS per stabilire una connessione di lunga durata, quindi inizia WebSockets a trasmettere i dati attraverso la connessione agli endpoint sicuri del dispositivo di tunneling.

Prima di eseguire questi comandi:

Prima di poter eseguire questi comandi, è necessario aver già aperto un tunnel e aver ottenuto i token di accesso del client per l'origine e la destinazione. Inoltre, devi avere creato il proxy locale come descritto in precedenza. Per creare il proxy locale, apri il codice sorgente del proxy locale nel GitHub repository e segui le istruzioni per la creazione e l'installazione del proxy locale.

> **Note**
>
> I seguenti comandi utilizzati negli esempi utilizzano il flag `verbosity` per illustrare una panoramica dei diversi passaggi descritti in precedenza dopo l'esecuzione del proxy locale. È consigliabile utilizzare questo flag solo per effettuare dei test.

Esecuzione di proxy locale in modalità di origine

I seguenti comandi mostrano come eseguire il proxy locale in modalità sorgente.

Linux/macOS

Su Linux o macOS, esegui i seguenti comandi nel terminale per configurare e avviare il proxy locale sull'origine.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -s 5555 -v 5 -r us-west-2
```

Dove:

- `-s` è la porta di ascolto dell'origine che avvia il proxy locale nella modalità di origine.
- `-v` è la verbosità dell'output che può essere un valore compreso tra zero e sei.
- `-r` è la regione dell'endpoint in cui è aperto il tunnel.

Per ulteriori informazioni sui parametri, consulta la sezione Opzioni impostate utilizzando gli argomenti della riga di comando.

Windows

Su Windows, si configura il proxy locale in modo simile a quello che si esegue per Linux o macOS, ma il modo in cui si definiscono le variabili di ambiente è diverso dalle altre piattaforme. Esegui i seguenti comandi nella finestra `cmd` per configurare e avviare il proxy locale sull'origine.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -s 5555 -v 5 -r us-west-2
```

Dove:

- `-s` è la porta di ascolto dell'origine che avvia il proxy locale nella modalità di origine.
- `-v` è la verbosità dell'output che può essere un valore compreso tra zero e sei.
- `-r` è la regione dell'endpoint in cui è aperto il tunnel.

Per ulteriori informazioni sui parametri, consulta la sezione Opzioni impostate utilizzando gli argomenti della riga di comando.

> **Note**
>
> Quando si utilizza la versione più recente del proxy locale in modalità sorgente, è necessario includere il AWS CLI parametro `--destination-client-type V1` sul dispositivo di origine per garantire la compatibilità con le versioni precedenti. Questo vale quando ci si connette a una di queste modalità di destinazione:
>
> - AWS IoT Client del dispositivo
> - AWS IoT Componente di tunneling sicuro o componente di tunneling AWS IoT Greengrass Version 2 sicuro
> - Qualsiasi codice dimostrativo di AWS IoT Secure Tunneling scritto prima del 2022
> - Versioni 1.X del proxy locale
>
> Questo parametro garantisce una comunicazione corretta tra il proxy di origine aggiornato e i client di destinazione precedenti. Per ulteriori informazioni sulle versioni proxy locali, vedere AWS IoT Secure Tunneling on. GitHub

Di seguito è riportato un esempio di output dell'esecuzione del proxy locale in `source` modalità.

```
...
...

Starting proxy in source mode
```

```
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Esecuzione del proxy locale nella modalità di destinazione

I comandi seguenti mostrano come eseguire il proxy locale in modalità destinazione.

Linux/macOS

Su Linux o macOS, esegui i seguenti comandi nel terminale per configurare e avviare il proxy locale sulla destinazione.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -d 22 -v 5 -r us-west-2
```

Dove:

- `-d` è l'applicazione di destinazione che avvia il proxy locale nella modalità di destinazione.
- `-v` è la verbosità dell'output che può essere un valore compreso tra zero e sei.
- `-r` è la regione dell'endpoint in cui è aperto il tunnel.

Per ulteriori informazioni sui parametri, consulta la sezione Opzioni impostate utilizzando gli argomenti della riga di comando.

Windows

Su Windows, si configura il proxy locale in modo simile a quello che si esegue per Linux o macOS, ma il modo in cui si definiscono le variabili di ambiente è diverso dalle altre piattaforme. Esegui i seguenti comandi nella finestra `cmd` per configurare e avviare il proxy locale sulla destinazione.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -d 22 -v 5 -r us-west-2
```

Dove:

- `-d` è l'applicazione di destinazione che avvia il proxy locale nella modalità di destinazione.
- `-v` è la verbosità dell'output che può essere un valore compreso tra zero e sei.
- `-r` è la regione dell'endpoint in cui è aperto il tunnel.

Per ulteriori informazioni sui parametri, consulta la sezione Opzioni impostate utilizzando gli argomenti della riga di comando.

> **Note**
>
> Quando si utilizza la versione più recente del proxy locale in modalità di destinazione, è necessario includere il AWS CLI parametro `--destination-client-type V1` sul dispositivo di destinazione per garantire la compatibilità con le versioni precedenti. Questo vale quando ci si connette a una di queste modalità di origine:
>
> - Tunneling sicuro basato su browser dalla console. AWS
> - Versioni 1.X del proxy locale

Questo parametro garantisce una comunicazione corretta tra il proxy di destinazione aggiornato e i client di origine precedenti. Per ulteriori informazioni sulle versioni proxy locali, vedere AWS IoT Secure Tunneling on. GitHub

Di seguito è riportato un esempio di output dell'esecuzione del proxy locale in `destination` modalità.

```
...
...

Starting proxy in destination mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
```

Configurare il proxy locale per i dispositivi che utilizzano il proxy web

È possibile utilizzare il proxy locale sui AWS IoT dispositivi per comunicare tramite AWS IoT APIs tunneling sicuro. Il proxy locale trasmette i dati inviati dall'applicazione del dispositivo utilizzando un tunneling sicuro su una connessione sicura. WebSocket Il proxy locale può funzionare in modalità `source` o `destination`. Nella modalità `source`, viene eseguito sullo stesso dispositivo o rete che avvia la connessione TCP. Nella modalità `destination`, il proxy locale viene eseguito sul dispositivo remoto, insieme all'applicazione di destinazione. Per ulteriori informazioni, consulta Proxy locale.

Il proxy locale deve connettersi direttamente a Internet per utilizzare AWS IoT il tunneling sicuro. Per una connessione TCP di lunga durata con tunneling sicuro, il proxy locale aggiorna la richiesta HTTPS per stabilire una WebSockets connessione a uno degli endpoint di connessione del dispositivo di tunneling sicuro.

Se i tuoi dispositivi si trovano in una rete che utilizza un proxy Web, il proxy Web può intercettare le connessioni prima di inoltrarle a Internet. Per stabilire una connessione di lunga durata agli endpoint di connessione del dispositivo di tunneling protetto, configura il proxy locale per l'utilizzo del proxy web come descritto nella sezione specifica websocket.

> **Note**
>
> AWS IoT Client del dispositivo non supporta dispositivi che utilizzano un proxy web. Per lavorare con il proxy web, è necessario utilizzare un proxy locale e configurarlo per il funzionamento con un proxy web come descritto di seguito.

I passaggi seguenti mostrano come il proxy locale funziona con un proxy web.

1. Il proxy locale invia una richiesta HTTP `CONNECT` al proxy web che contiene l'indirizzo remoto del servizio di tunneling protetto, insieme alle informazioni di autenticazione del proxy Web.
2. Il proxy web creerà quindi una connessione di lunga durata agli endpoint di tunneling protetti remoti.
3. La connessione TCP viene stabilita e il proxy locale ora funzionerà sia in modalità di origine che di destinazione per la trasmissione dei dati.

Per completare questa procedura, esegui le fasi seguenti.

- Crea il proxy locale

- Configurazione del proxy Web
- Configura e avvia il proxy locale

Crea il proxy locale

Apri il codice sorgente del proxy locale nel GitHub repository e segui le istruzioni per creare e installare il proxy locale.

Configurazione del proxy Web

Il proxy locale si basa sul meccanismo di tunneling HTTP descritto dalle Specifiche HTTP/1.1. Per rispettare le specifiche, il proxy web deve consentire ai dispositivi di utilizzare il metodo `CONNECT`.

La modalità di configurazione del proxy web dipende dal proxy web utilizzato e dalla versione del proxy web. Per assicurarsi di configurare correttamente il proxy web, controlla la documentazione del proxy web.

Per configurare il proxy web, identifica innanzitutto l'URL del proxy web e verifica se il proxy web supporta il tunneling HTTP. L'URL del proxy web verrà utilizzato in un secondo momento quando si configura e si avvia il proxy locale.

1. Identifica l'URL del proxy web

   L'URL del proxy web sarà nel formato seguente.

   ```
   protocol://web_proxy_host_domain:web_proxy_port
   ```

   AWS IoT il tunneling sicuro supporta solo l'autenticazione di base per il proxy web. Per utilizzare l'autenticazione di base, è necessario specificare **`username`** e **`password`** come parte dell'URL del proxy web. L'URL del proxy web sarà nel formato seguente.

   ```
   protocol://username:password@web_proxy_host_domain:web_proxy_port
   ```

   - *`protocol`*può essere o. `http` `https` Ti consigliamo di utilizzare `https`.
   - *`web_proxy_host_domain`*è l'indirizzo IP del proxy Web o un nome DNS che si risolve nell'indirizzo IP del proxy Web.
   - *`web_proxy_port`*è la porta su cui il proxy web è in ascolto.
   - Il proxy web utilizza questo **`username`** e **`password`** per autenticare la richiesta.

2. Verificare l'URL del proxy web

   Per confermare se il proxy web supporta il tunneling TCP, utilizza un comando `curl` e assicurati di ottenere una risposta `2xx` o `3xx`.

   Ad esempio, se l'URL del proxy web è `https://server.com:1235`, utilizza un flag `proxy-insecure` con il comando `curl` perché il proxy web potrebbe basarsi su un certificato autofirmato.

   ```
   export HTTPS_PROXY=https://server.com:1235
   curl -I https://aws.amazon.com --proxy-insecure
   ```

   Se l'URL del proxy web ha una porta `http` (ad esempio, `http://server.com:1234`), non è necessario utilizzare il flag `proxy-insecure`.

   ```
   export HTTPS_PROXY=http://server.com:1234
   curl -I https://aws.amazon.com
   ```

Configura e avvia il proxy locale

Per configurare il proxy locale affinché utilizzi un proxy web, puoi configurare le variabili di ambiente `HTTPS_PROXY` con i nomi di dominio DNS o gli indirizzi IP e i numeri della porta utilizzati dai server proxy.

Dopo aver configurato il proxy locale, è possibile utilizzare il proxy locale come spiegato in questo documento README.

> **Note**
>
> La dichiarazione di ambiente variabile prevede la distinzione tra lettere maiuscole e minuscole. È consigliabile definire ogni variabile una volta utilizzando tutte le lettere maiuscole o minuscole. Gli esempi seguenti mostrano il nome della variabile di ambiente in lettere maiuscole. Se la stessa variabile viene specificata utilizzando lettere maiuscole e minuscole, la variabile specificata con lettere minuscole ha la precedenza.

I comandi seguenti mostrano come configurare il proxy locale in esecuzione sulla destinazione per utilizzare il proxy web e avviare il proxy locale.

- `AWSIOT_TUNNEL_ACCESS_TOKEN`: questa variabile contiene il token di accesso del client (CAT) per la destinazione.
- `HTTPS_PROXY`: questa variabile contiene l'URL del proxy web o l'indirizzo IP per la configurazione del proxy locale.

I comandi illustrati negli esempi seguenti dipendono dal sistema operativo utilizzato e dal fatto che il proxy web sia in ascolto su una porta HTTP o HTTPS.

Proxy web in ascolto su una porta HTTP

Se il proxy web è in ascolto su una porta HTTP, è possibile fornire l'URL del proxy web o l'indirizzo IP per la variabile `HTTPS_PROXY`.

Linux/macOS

Su Linux o macOS, eseguire i seguenti comandi nel terminale per configurare e avviare il proxy locale sulla destinazione per utilizzare un proxy web in ascolto di una porta HTTP.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22
```

Se è necessario autenticarsi con il proxy, è necessario specificare **`username`** e **`password`** come parte della variabile `HTTPS_PROXY`.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http://username:password@proxy.example.com:1234
./localproxy -r us-east-1 -d 22
```

Windows

Su Windows, si configura il proxy locale in modo simile a quello che si esegue per Linux o macOS, ma il modo in cui si definiscono le variabili di ambiente è diverso dalle altre piattaforme. Esegui i comandi seguenti nella finestra `cmd` per configurare e avviare il proxy locale sulla destinazione per utilizzare un proxy web in ascolto di una porta HTTP.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
```

```
.\localproxy -r us-east-1 -d 22
```

Se è necessario autenticarsi con il proxy, è necessario specificare **`username`** e **`password`** come parte della variabile HTTPS_PROXY.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://username:password@10.15.20.25:1234
.\localproxy -r us-east-1 -d 22
```

Proxy web in ascolto su una porta HTTPS

Esegui i seguenti comandi se il proxy web è in ascolto su una porta HTTPS.

> **Note**
>
> Se utilizzi un certificato autofirmato per il proxy web o se esegui il proxy locale su un sistema operativo che non dispone del supporto OpenSSL nativo e delle configurazioni predefinite, dovrai configurare i certificati proxy web come descritto nella sezione Configurazione dei certificati del repository. GitHub

I comandi seguenti avranno un aspetto simile alla configurazione del proxy web per un proxy HTTP, con l'eccezione che verrà specificato anche il percorso dei file di certificato installati come descritto in precedenza.

Linux/macOS

In Linux o macOS, esegui i seguenti comandi nel terminale per configurare il proxy locale in esecuzione sulla destinazione per utilizzare un proxy web in ascolto di una porta HTTPS.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22 -c /path/to/certs
```

Se è necessario autenticarsi con il proxy, è necessario specificare **`username`** e **`password`** come parte della variabile HTTPS_PROXY.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
```

```
export HTTPS_PROXY=http://username:password@proxy.example.com:1234
./localproxy -r us-east-1 -d 22 -c /path/to/certs
```

Windows

Su Windows, esegui i comandi seguenti nella finestra cmd per configurare e avviare il proxy locale in esecuzione sulla destinazione per utilizzare un proxy web in ascolto di una porta HTTP.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
.\localproxy -r us-east-1 -d 22 -c \path\to\certs
```

Se è necessario autenticarsi con il proxy, è necessario specificare **`username`** e **`password`** come parte della variabile HTTPS_PROXY.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://username:password@10.15.20.25:1234
.\localproxy -r us-east-1 -d 22 -c \path\to\certs
```

Esempio di comando e output

Di seguito viene illustrato un esempio di comando eseguito su sistema operativo Linux e l'output corrispondente. L'esempio mostra un proxy Web che è in ascolto su una porta HTTP e come il proxy locale può essere configurato per l'utilizzo del proxy web sia nella modalità `source` che nella modalità `destination`. Prima di poter eseguire questi comandi, è necessario aver già aperto un tunnel e aver ottenuto i token di accesso del client per l'origine e la destinazione. È inoltre necessario aver creato il proxy locale e aver configurato il proxy web come descritto in precedenza.

Ecco una panoramica dei passaggi dopo aver avviato il proxy locale. Il proxy locale:

- Identifica l'URL del proxy web in modo che possa utilizzare l'URL per connettersi al server proxy.
- Stabilisce una connessione TCP con il proxy web.
- Invia una richiesta HTTP `CONNECT` al proxy web e attende la risposta `HTTP/1.1 200`, che indica che la connessione è stata stabilita.
- Aggiorna il protocollo HTTPS per stabilire una connessione di lunga durata WebSockets .
- Avvia la trasmissione dei dati tramite la connessione agli endpoint del dispositivo di tunneling sicuro.

> **Note**
>
> I seguenti comandi utilizzati negli esempi utilizzano il flag `verbosity` per illustrare una panoramica dei diversi passaggi descritti in precedenza dopo l'esecuzione del proxy locale. È consigliabile utilizzare questo flag solo per effettuare dei test.

Esecuzione di proxy locale in modalità di origine

I comandi seguenti mostrano come eseguire il proxy locale nella modalità di origine.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:username:password@10.15.10.25:1234
./localproxy -s 5555 -v 5 -r us-west-2
```

Di seguito è riportato un output di esempio dell'esecuzione del proxy locale nella modalità `source`.

```
...

Parsed basic auth credentials for the URL
Found Web proxy information in the environment variables, will use it to connect via
 the proxy.

...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved Web proxy IP: 10.10.0.11
Connected successfully with Web Proxy
Successfully sent HTTP CONNECT to the Web proxy
Full response from the Web proxy:
HTTP/1.1 200 Connection established
TCP tunnel established successfully
Connected successfully with proxy server
Successfully completed SSL handshake with proxy server
Web socket session ID: 0a109afffee745f5-00001341-000b8138-cc6c878d80e8adb0-f186064b
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Seting up web socket pings for every 5000 milliseconds
Scheduled next read:
```

```
...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Esecuzione del proxy locale nella modalità di destinazione

I comandi seguenti mostrano come eseguire il proxy locale nella modalità di destinazione.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:username:password@10.15.10.25:1234
./localproxy -d 22 -v 5 -r us-west-2
```

Di seguito è riportato un output di esempio dell'esecuzione del proxy locale nella modalità `destination`.

```
...

Parsed basic auth credentials for the URL
Found Web proxy information in the environment variables, will use it to connect via
 the proxy.

...

Starting proxy in destination mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved Web proxy IP: 10.10.0.1
Connected successfully with Web Proxy
Successfully sent HTTP CONNECT to the Web proxy
Full response from the Web proxy:
HTTP/1.1 200 Connection established
TCP tunnel established successfully
Connected successfully with proxy server
Successfully completed SSL handshake with proxy server
Web socket session ID: 06717bfffed3fd05-00001355-000b8315-da3109a85da804dd-24c3d10d
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Seting up web socket pings for every 5000 milliseconds
Scheduled next read:
```

```
...

Starting web socket read loop continue reading...
```

Flussi di dati multiplex e utilizzo di connessioni TCP simultanee in un tunnel sicuro

Puoi utilizzare più flussi dei dati per tunnel mediante la caratteristica multiplexing di tunneling sicuro. Con il multiplexing, puoi risolvere i problemi dei dispositivi utilizzando più flussi di dati. È inoltre possibile ridurre il carico operativo eliminando la necessità di creare, implementare e avviare più proxy locali o aprire più tunnel sullo stesso dispositivo. Ad esempio, il multiplexing può essere utilizzato nel caso di un browser web che richiede l'invio di più flussi di dati HTTP e SSH.

Per ogni flusso di dati, il tunneling AWS IoT sicuro supporta connessioni TCP simultanee. L'utilizzo di connessioni simultanee riduce il rischio di timeout in caso di richieste multiple da parte del client. Ad esempio, può ridurre il tempo di caricamento quando si accede in remoto a un server Web locale al dispositivo di destinazione.

Nelle sezioni seguenti vengono fornite ulteriori informazioni sul multiplexing e sull'utilizzo di connessioni TCP simultanee e sui diversi casi d'uso.

Argomenti

- Flussi multipli dei dati multiplex in un tunnel sicuro
- Utilizzo di connessioni TCP simultanee in un tunnel sicuro

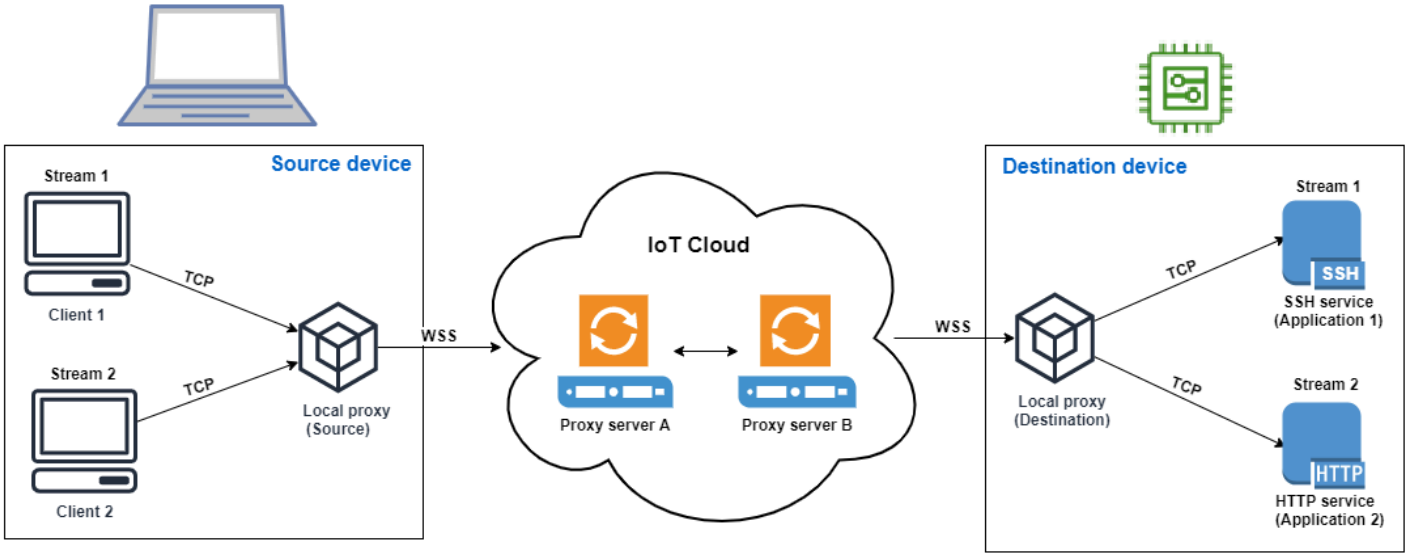
Flussi multipli dei dati multiplex in un tunnel sicuro

È possibile utilizzare la funzionalità di multiplexing per dispositivi che utilizzano più connessioni o porte. Il multiplexing può essere utilizzato anche quando sono necessarie più connessioni a un dispositivo remoto per risolvere eventuali problemi. Ad esempio, può essere utilizzato nel caso di un browser web che richiede l'invio di più flussi di dati HTTP e SSH. I dati delle applicazioni da entrambi gli stream vengono inviati contemporaneamente al dispositivo attraverso il tunnel multiplex.

Esempio di caso d’uso

Ad esempio, potrebbe essere necessario connettersi a un'applicazione web sul dispositivo per modificare alcuni parametri di rete, mentre contemporaneamente emettono comandi shell attraverso

il terminale per verificare che il dispositivo funzioni correttamente con i nuovi parametri di rete. In questo scenario, potrebbe essere necessario connettersi al dispositivo tramite HTTP e SSH e trasferire due flussi di dati paralleli per accedere contemporaneamente all'applicazione Web e al terminale. Con la funzione multiplexing, questi due flussi indipendenti possono essere trasferiti contemporaneamente sullo stesso tunnel.



Come impostare un tunnel multiplex

La procedura seguente illustra come configurare un tunnel multiplex per la risoluzione dei problemi relativi ai dispositivi che utilizzano applicazioni che richiedono connessioni a più porte. Imposterai un tunnel con due flussi multiplex: un flusso HTTP e uno SSH.

1. (Facoltativo) Crea file di configurazione

   Facoltativamente è possibile configurare il dispositivo di origine e di destinazione con i file di configurazione. Usa i file di configurazione se è probabile che le mappature delle porte cambino frequentemente. È possibile saltare questo passaggio se si preferisce specificare la mappatura delle porte tramite CLI o non è necessario avviare il proxy locale su porte di ascolto designate. Per ulteriori informazioni su come utilizzare i file di configurazione, vedi Opzioni impostate tramite --config in. GitHub

   1. Sul tuo dispositivo di origine, nella cartella in cui verrà eseguito il proxy locale, crea una cartella di configurazione chiamata `Config`. All'interno di questa cartella, crea un file chiamato `SSHSource.ini` con il seguente contenuto:

      ```
      HTTP1 = 5555
      ```

```
SSH1 = 3333
```

2. Sul dispositivo di destinazione, nella cartella in cui verrà eseguito il proxy locale, crea una cartella di configurazione chiamata `Config`. All'interno di questa cartella, crea un file chiamato `SSHDestination.ini` con il seguente contenuto:

```
HTTP1 = 80
SSH1 = 22
```

2. Apertura di un tunnel

Aprire un tunnel utilizzando l'operazione `OpenTunnel` API o il comando `open-tunnel` CLI. Configura la destinazione specificando `SSH1` e `HTTP1` come servizi e il nome dell' AWS IoT elemento che corrisponde al tuo dispositivo remoto. Le applicazioni SSH e HTTP sono in esecuzione su questo dispositivo remoto. È necessario aver già creato l'oggetto IoT nel AWS IoT registro. Per ulteriori informazioni, consulta Gestire gli elementi con il registro.

```
aws iotsecuretunneling open-tunnel \
 --destination-config thingName=RemoteDevice1,services=HTTP1,SSH1
```

L'esecuzione di questo comando genera i token di accesso di origine e destinazione che utilizzerai per eseguire il proxy locale.

```
{
 "tunnelId": "b2de92a3-b8ff-46c0-b0f2-afa28b00cecd",
 "tunnelArn": "arn:aws:iot:us-west-2:431600097591:tunnel/b2de92a3-b8ff-46c0-b0f2-
afa28b00cecd",
 "sourceAccessToken": source_client_access_token,
 "destinationAccessToken": destination_client_access_token
}
```

3. Configurazione e avvio del proxy locale

Prima di poter eseguire il proxy locale, configura il AWS IoT Device Client o scarica il codice sorgente del proxy locale da GitHube crealo per la piattaforma che preferisci. È quindi possibile avviare il proxy locale di destinazione e di origine per connettersi al tunnel sicuro. Per ulteriori informazioni sulla configurazione e l'utilizzo del proxy locale, consulta Come utilizzare il proxy locale.

> **Note**
>
> Sul dispositivo di origine, se non si utilizzano file di configurazione o non si specifica la mappatura delle porte utilizzando l'interfaccia a riga di comando, è comunque possibile utilizzare lo stesso comando per eseguire il proxy locale. Il proxy locale raccoglierà le porte disponibili per utilizzare e gestire le mappature al tuo posto.

Start local proxy using configuration files

Esegui i seguenti comandi per eseguire il proxy locale nelle modalità di origine e destinazione utilizzando i file di configurazione.

```
// ----------------- Start the destination local proxy ----------------------
./localproxy -r us-east-1 -m dst -t destination_client_access_token

// ----------------- Start the source local proxy ---------------------------
// You also run the same command below if you want the local proxy to
// choose the mappings for you instead of using configuration files.
./localproxy -r us-east-1 -m src -t source_client_access_token
```

Start local proxy using CLI port mapping

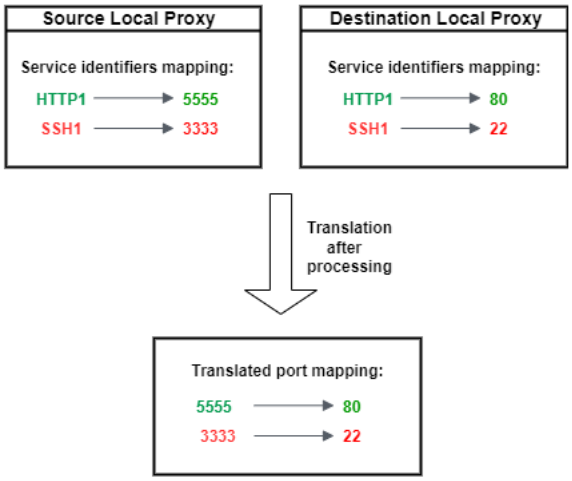
Esegui i seguenti comandi per eseguire il proxy locale nelle modalità di origine e destinazione specificando esplicitamente le mappature delle porte utilizzando l'interfaccia a riga di comando.

```
// ----------------- Start the destination local proxy
 ----------------------------------
./localproxy -r us-east-1 -d HTTP1=80,SSH1=22 -t destination_client_access_token

// ----------------- Start the source local proxy
 ---------------------------------------
./localproxy -r us-east-1 -s HTTP1=5555,SSH1=33 -t source_client_access_token
```

I dati delle applicazioni dalla connessione SSH e HTTP possono ora essere trasferiti contemporaneamente attraverso il tunnel multiplex. Come si può vedere dalla mappa sottostante, l'identificatore del servizio funge da formato leggibile per tradurre la mappatura delle porte tra il

dispositivo di origine e quello di destinazione. Con questa configurazione, il tunneling sicuro inoltra tutto il traffico HTTP in entrata dalla porta *5555* del dispositivo di origine alla porta del dispositivo di destinazione e tutto il traffico SSH in entrata da una porta *3333* all'altra *80* sul dispositivo di destinazione. *22*



Utilizzo di connessioni TCP simultanee in un tunnel sicuro

AWS IoT il tunneling sicuro supporta più di una connessione TCP contemporaneamente per ogni flusso di dati. È possibile utilizzare questa funzionalità quando sono necessarie connessioni simultanee a un dispositivo remoto. L'utilizzo di connessioni TCP simultanee riduce il rischio di timeout in caso di richieste multiple da parte del client. Ad esempio, quando si accede a un server Web su cui sono in esecuzione più componenti, le connessioni TCP simultanee possono ridurre il tempo necessario per caricare il sito.

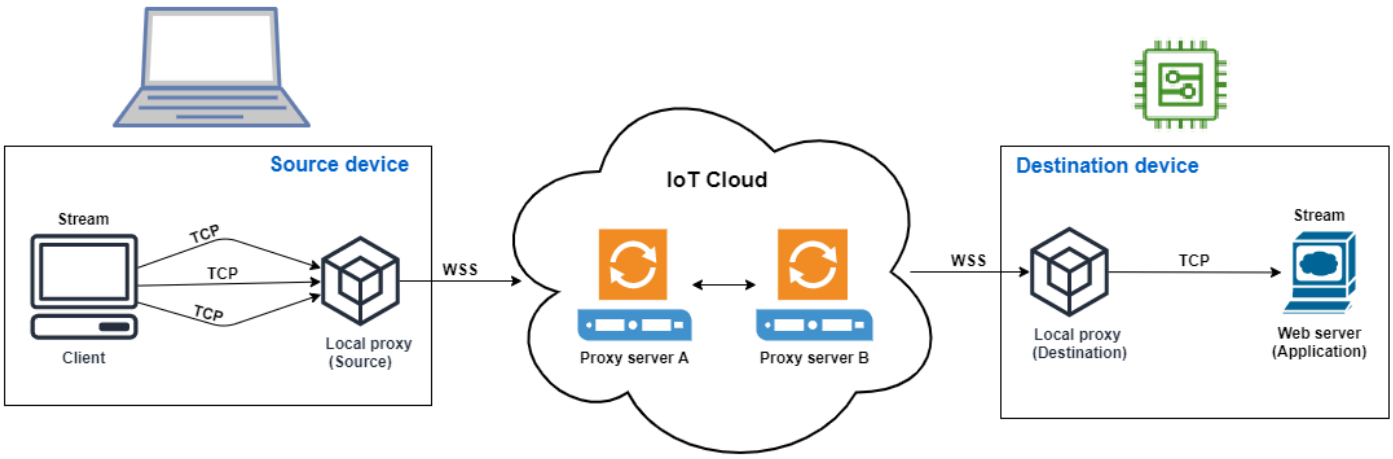
> **Note**
>
> Le connessioni TCP simultanee hanno un limite di larghezza di banda di 800 Kilobyte al secondo per ciascuna. Account AWS AWS IoT il tunneling sicuro può configurare questo limite automaticamente in base al numero di richieste in arrivo.

Esempio di caso d'uso

Supponiamo che sia necessario accedere in remoto a un server Web locale al dispositivo di destinazione e con più componenti in esecuzione su di esso. Con una singola connessione TCP, mentre si tenta di accedere al server Web, il caricamento sequenziale può aumentare il tempo necessario per caricare le risorse sul sito. Le connessioni TCP simultanee possono ridurre il tempo

di caricamento soddisfacendo i requisiti di risorse del sito, riducendo così il tempo di accesso. Il diagramma seguente mostra come sono supportate le connessioni TCP simultanee per il flusso di dati verso l'applicazione del server Web in esecuzione sul dispositivo remoto.

> **Note**
>
> Se si desidera accedere a più applicazioni in esecuzione sul dispositivo remoto utilizzando il tunnel, è possibile utilizzare il multiplexing del tunnel. Per ulteriori informazioni, consulta Flussi multipli dei dati multiplex in un tunnel sicuro.



Come usare connessioni TCP simultanee

La procedura seguente illustra come utilizzare connessioni TCP simultanee per accedere al browser Web sul dispositivo remoto. In caso di più richieste da parte del client, il tunneling AWS IoT sicuro configura automaticamente connessioni TCP simultanee per gestire le richieste, riducendo così i tempi di caricamento.

1. Apertura di un tunnel

   Aprire un tunnel utilizzando l'operazione `OpenTunnel` API o il comando `open-tunnel` CLI. Configura la destinazione specificando `HTTP` come servizio e il nome dell'oggetto AWS IoT che corrisponde al tuo dispositivo remoto. L'applicazione del server Web è in esecuzione su questo dispositivo remoto. È necessario aver già creato l'oggetto IoT nel AWS IoT registro. Per ulteriori informazioni, consulta Gestire gli elementi con il registro.

   ```
   aws iotsecuretunneling open-tunnel \
   ```

```
 --destination-config thingName=RemoteDevice1,services=HTTP
```

L'esecuzione di questo comando genera i token di accesso di origine e destinazione che utilizzerai per eseguire il proxy locale.

```
{
 "tunnelId": "b2de92a3-b8ff-46c0-b0f2-afa28b00cecd",
 "tunnelArn": "arn:aws:iot:us-west-2:431600097591:tunnel/b2de92a3-b8ff-46c0-b0f2-
afa28b00cecd",
 "sourceAccessToken": source_client_access_token,
 "destinationAccessToken": destination_client_access_token
}
```

2. Configurazione e avvio del proxy locale

   Prima di poter eseguire il proxy locale, scarica il codice sorgente del proxy locale da GitHube crealo per la piattaforma che preferisci. È quindi possibile avviare il proxy locale di destinazione e di origine per connettersi al tunnel sicuro e iniziare a utilizzare l'applicazione del server Web remoto.

   > **Note**
   >
   > Affinché il tunneling AWS IoT sicuro utilizzi connessioni TCP simultanee, devi eseguire l'aggiornamento alla versione più recente del proxy locale. Questa funzionalità non è disponibile se si configura il proxy locale utilizzando AWS IoT Device Client.

```
// Start the destination local proxy
./localproxy -r us-east-1 -d HTTP=80 -t destination_client_access_token

// Start the source local proxy
./localproxy -r us-east-1 -s HTTP=5555 -t source_client_access_token
```

   Per ulteriori informazioni sulla configurazione e l'utilizzo del proxy locale, consulta Come utilizzare il proxy locale.

È ora possibile utilizzare il tunnel per accedere all'applicazione del server Web. AWS IoT il tunneling sicuro configurerà e gestirà automaticamente le connessioni TCP simultanee quando ci sono più richieste dal client.

Configurazione di un dispositivo remoto e utilizzo dell'agente IoT

L'agente IoT viene utilizzato per ricevere il messaggio MQTT che include il token di accesso client e avviare un proxy locale sul dispositivo remoto. Se desideri che il tunneling sicuro fornisca il token di accesso client, devi installare ed eseguire l'agente IoT sul dispositivo remoto. L'agente IoT deve sottoscrivere l'argomento MQTT dell'IoT riservato riportato di seguito:

> **Note**
>
> Se si desidera consegnare il token di accesso del client di destinazione al dispositivo remoto tramite metodi diversi dalla sottoscrizione all'argomento MQTT riservato, potrebbero essere necessari un listener CAT (Destination Client Access Token) e un proxy locale. Il listener CAT deve funzionare con il meccanismo di consegna del token di accesso client scelto ed essere in grado di avviare un proxy locale in modalità di destinazione.

Snippet dell'agente IoT

L'agente IoT deve iscriversi al seguente argomento IoT MQTT riservato in modo da poter ricevere il messaggio MQTT e avviare il proxy locale:

`$aws/things/`*`thing-name`*`/tunnels/notify`

`thing-name`Dov'è il nome dell' AWS IoT oggetto associato al dispositivo remoto.

Di seguito è riportato un esempio di payload del messaggio MQTT:

```
{
    "clientAccessToken": "destination-client-access-token",
    "clientMode": "destination",
    "region": "aws-region",
    "services": ["destination-service"]
}
```

Dopo aver ricevuto un messaggio MQTT, l'agente IoT deve avviare un proxy locale sul dispositivo remoto con i parametri appropriati.

Il seguente codice Java dimostra come utilizzare AWS IoT Device SDK e ProcessBuilderdalla libreria Java per creare un semplice agente IoT che funzioni con il tunneling sicuro.

```
// Find the IoT device endpoint for your Account AWS
final String endpoint = iotClient.describeEndpoint(new
 DescribeEndpointRequest().withEndpointType("iot:Data-ATS")).getEndpointAddress();

// Instantiate the IoT Agent with your AWS credentials
final String thingName = "RemoteDeviceA";
final String tunnelNotificationTopic = String.format("$aws/things/%s/tunnels/notify",
 thingName);
final AWSIotMqttClient mqttClient = new AWSIotMqttClient(endpoint, thingName,
                 "your_aws_access_key", "your_aws_secret_key");

try {
    mqttClient.connect();
    final TunnelNotificationListener listener = new
 TunnelNotificationListener(tunnelNotificationTopic);
    mqttClient.subscribe(listener, true);
}
finally {
    mqttClient.disconnect();
}

private static class TunnelNotificationListener extends AWSIotTopic {
    public TunnelNotificationListener(String topic) {
        super(topic);
    }

    @Override
    public void onMessage(AWSIotMessage message) {
        try {
            // Deserialize the MQTT message
            final JSONObject json = new JSONObject(message.getStringPayload());

            final String accessToken = json.getString("clientAccessToken");
            final String region = json.getString("region");

            final String clientMode = json.getString("clientMode");
            if (!clientMode.equals("destination")) {
                throw new RuntimeException("Client mode " + clientMode + " in the MQTT
 message is not expected");
            }

            final JSONArray servicesArray = json.getJSONArray("services");
            if (servicesArray.length() > 1) {
```

```
                throw new RuntimeException("Services in the MQTT message has more than
 1 service");
            }
            final String service = servicesArray.get(0).toString();
            if (!service.equals("SSH")) {
                throw new RuntimeException("Service " + service + " is not supported");
            }

            // Start the destination local proxy in a separate process to connect to
 the SSH Daemon listening port 22
            final ProcessBuilder pb = new ProcessBuilder("localproxy",
                        "-t", accessToken,
                        "-r", region,
                        "-d", "localhost:22");
            pb.start();
        }
        catch (Exception e) {
            log.error("Failed to start the local proxy", e);
        }
    }
}
```

Controllo dell'accesso ai tunnel

Il tunneling sicuro fornisce le operazioni, le risorse e le chiavi di contesto della condizione specifiche del servizio da utilizzare nelle policy di autorizzazione IAM.

Prerequisiti di accesso al tunnel

- Scopri come proteggere le AWS risorse utilizzando le policy IAM.
- Informazioni su come creare e valutare le condizioni IAM.
- Scopri come proteggere AWS le risorse utilizzando i tag delle risorse.

Policy di accesso al tunnel

È necessario utilizzare le seguenti policy per concedere le autorizzazioni a utilizzare l'API di tunneling sicuro. Per ulteriori informazioni sulla AWS IoT sicurezza, consultaGestione delle identità e degli accessi per AWS IoT.

iot: OpenTunnel

L'operazione policy `iot:OpenTunnel` concede un'autorizzazione principale per chiamare OpenTunnel.

Nell'elemento `Resource` dell'istruzione di policy IAM:

- Specifica l'ARN del tunnel con carattere jolly:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`
- Specifica l'ARN di un oggetto per gestire l'autorizzazione `OpenTunnel` per specifici oggetti IoT:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:thing/`*`thing-name`*

Ad esempio, la seguente dichiarazione di policy consente di aprire un tunnel per l'oggetto IoT denominato `TestDevice`.

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

L'operazione policy `iot:OpenTunnel` supporta le seguenti chiavi di condizione:

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `aws:RequestTag/`*`tag-key`*
- `aws:SecureTransport`
- `aws:TagKeys`

La seguente istruzione della policy consente di aprire un tunnel per l'oggetto, se l'oggetto appartiene a un gruppo di oggetti con un nome che inizia con `TestGroup` e il servizio di destinazione configurato nel tunnel è SSH.

```
{
```

```
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "ForAnyValue:StringLike": {
            "iot:ThingGroupArn": [
                "arn:aws:iot:aws-region:aws-account-id:thinggroup/TestGroup*"
            ]
        },
        "ForAllValues:StringEquals": {
            "iot:TunnelDestinationService": [
                "SSH"
            ]
        }
    }
}
```

È inoltre possibile utilizzare i tag delle risorse per controllare le autorizzazioni per aprire i tunnel. Ad esempio, la seguente dichiarazione policy consente di aprire un tunnel se la chiave tag `Owner` è presente con un valore di `Admin` e non vengono specificati altri tag. Per informazioni generali sull'utilizzo dei tag, consulta Taggare le tue risorse AWS IoT.

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/Owner": "Admin"
        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "Owner"
        }
    }
}
```

IoT: RotateTunnelAccessToken

L'operazione policy `iot:RotateTunnelAccessToken` concede un'autorizzazione principale per chiamare RotateTunnelAccessToken.

Nell'elemento `Resource` dell'istruzione di policy IAM:

- Specifica l'ARN del tunnel completo:

  `arn:aws:iot:`*`aws-region`*`: `*`aws-account-id`*`:tunnel/`*`tunnel-id`*

  Puoi anche utilizzare l'ARN del tunnel con carattere jolly:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`

- Specifica l'ARN di un oggetto per gestire l'autorizzazione `RotateTunnelAccessToken` per specifici oggetti IoT:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:thing/`*`thing-name`*

Ad esempio, la seguente istruzione di policy consente di ruotare il token di accesso di origine di un tunnel o il token di accesso di destinazione di un tunnel per l'oggetto IoT denominato `TestDevice`.

```
{
    "Effect": "Allow",
    "Action": "iot:RotateTunnelAccessToken",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

L'operazione policy `iot:RotateTunnelAccessToken` supporta le seguenti chiavi di condizione:

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `iot:ClientMode`
- `aws:SecureTransport`

La seguente istruzione di policy consente di ruotare il token di accesso di destinazione sull'oggetto se l'oggetto appartiene a un gruppo di oggetti con un nome che inizia con `TestGroup`, il servizio di destinazione configurato nel tunnel è SSH e il client è in modalità `DESTINATION`.

```
{
    "Effect": "Allow",
    "Action": "iot:RotateTunnelAccessToken",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "ForAnyValue:StringLike": {
            "iot:ThingGroupArn": [
                "arn:aws:iot:aws-region:aws-account-id:thinggroup/TestGroup*"
            ]
        },
        "ForAllValues:StringEquals": {
            "iot:TunnelDestinationService": [
                "SSH"
            ],
            "iot:ClientMode": "DESTINATION"
        }
    }
}
```

IoT: DescribeTunnel

L'operazione policy `iot:DescribeTunnel` concede un'autorizzazione principale per chiamare DescribeTunnel.

Nell'elemento `Resource` dell'istruzione di policy IAM, specifica l'ARN del tunnel completo:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

Puoi anche utilizzare l'ARN del carattere jolly:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

L'operazione policy `iot:DescribeTunnel` supporta le seguenti chiavi di condizione:

- `aws:ResourceTag/tag-key`
- `aws:SecureTransport`

La seguente dichiarazione policy consente di chiamare `DescribeTunnel` se il tunnel richiesto è contrassegnato con la chiave `Owner` con un valore pari a `Admin`.

```
{
    "Effect": "Allow",
    "Action": "iot:DescribeTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/Owner": "Admin"
        }
    }
}
```

IoT: ListTunnels

L'operazione policy `iot:ListTunnels` concede un'autorizzazione principale per chiamare ListTunnels.

Nell'elemento `Resource` dell'istruzione di policy IAM:

- Specifica l'ARN del tunnel con carattere jolly:

  `arn:aws:iot:aws-region:aws-account-id:tunnel/*`
- Specifica l'ARN di un oggetto per gestire l'autorizzazione `ListTunnels` per gli oggetti IoT selezionati:

  `arn:aws:iot:aws-region:aws-account-id:thing/thing-name`

L'operazione di policy `iot:ListTunnels` supporta la chiave di condizione `aws:SecureTransport`.

La seguente dichiarazione policy consente di elencare i tunnel per l'oggetto denominato `TestDevice`.

```
{
    "Effect": "Allow",
    "Action": "iot:ListTunnels",
    "Resource": [
```

```
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

IoT: ListTagsForResource

L'operazione policy `iot:ListTagsForResource` concede un'autorizzazione principale per chiamare `ListTagsForResource`.

Nell'elemento `Resource` dell'istruzione di policy IAM, specifica l'ARN del tunnel completo:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

Puoi anche utilizzare l'ARN del tunnel con carattere jolly:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

L'operazione di policy `iot:ListTagsForResource` supporta la chiave di condizione `aws:SecureTransport`.

IoT: CloseTunnel

L'operazione policy `iot:CloseTunnel` concede un'autorizzazione principale per chiamare CloseTunnel.

Nell'elemento `Resource` dell'istruzione di policy IAM, specifica l'ARN del tunnel completo:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

Puoi anche utilizzare l'ARN del tunnel con carattere jolly:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

L'operazione policy `iot:CloseTunnel` supporta le seguenti chiavi di condizione:

- `iot:Delete`
- `aws:ResourceTag/tag-key`
- `aws:SecureTransport`

La seguente dichiarazione policy consente di chiamare `CloseTunnel` se il parametro `Delete` della richiesta è `false` e il tunnel richiesto viene taggato con la chiave `Owner` con un valore pari a `QATeam`.

```
{
    "Effect": "Allow",
    "Action": "iot:CloseTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "Bool": {
            "iot:Delete": "false"
        },
        "StringEquals": {
            "aws:ResourceTag/Owner": "QATeam"
        }
    }
}
```

IoT: TagResource

L'operazione policy `iot:TagResource` concede un'autorizzazione principale per chiamare `TagResource`.

Nell'elemento `Resource` dell'istruzione di policy IAM, specifica l'ARN del tunnel completo:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

Puoi anche utilizzare l'ARN del tunnel con carattere jolly:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

L'operazione di policy `iot:TagResource` supporta la chiave di condizione `aws:SecureTransport`.

IoT: UntagResource

L'operazione policy `iot:UntagResource` concede un'autorizzazione principale per chiamare `UntagResource`.

Nell'elemento `Resource` dell'istruzione di policy IAM, specifica l'ARN del tunnel completo:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

Puoi anche utilizzare l'ARN del tunnel con carattere jolly:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

L'operazione di policy `iot:UntagResource` supporta la chiave di condizione `aws:SecureTransport`.

Risoluzione dei problemi di connettività del tunneling AWS IoT sicuro mediante la rotazione dei token di accesso del client

Quando si utilizza il tunneling AWS IoT sicuro, è possibile che si verifichino problemi di connettività anche se il tunnel è aperto. Le sezioni seguenti illustrano alcuni possibili problemi e come è possibile risolverli con la rotazione dei token di accesso client. Per ruotare il client access token (CAT), usa l'RotateTunnelAccessTokenAPI o il. rotate-tunnel-access-token AWS CLI A seconda che si verifichi un errore con l'utilizzo del client in modalità di origine o di destinazione, è possibile ruotare il CAT in modalità di origine o destinazione o entrambe.

> **Note**
>
> - Se non sei sicuro se il CAT debba essere ruotato sull'origine o sulla destinazione, puoi ruotare il CAT sia sull'origine che sulla destinazione impostando `ClientMode` su ALL quando usi l'API `RotateTunnelAccessToken`.
> - La rotazione del CAT non estende la durata del tunnel. Ad esempio, supponiamo che la durata del tunnel è di 12 ore e il tunnel è già aperto da 4 ore. Quando ruoti i token di accesso, i nuovi token generati possono essere utilizzati solo per le restanti 8 ore.

Argomenti

- Errore del token di accesso client non valido
- Errore di mancata corrispondenza del token client
- Problemi di connettività del dispositivo remoto

Errore del token di accesso client non valido

Quando si utilizza il tunneling AWS IoT sicuro, è possibile che si verifichi un errore di connessione quando si utilizza lo stesso token di accesso client (CAT) per riconnettersi allo stesso tunnel. In questo caso, il proxy locale non può connettersi al server proxy di tunneling sicuro. Se utilizzi un client in modalità di origine, potresti visualizzare il seguente messaggio di errore:

```
Invalid access token: The access token was previously used and cannot be used again
```

L'errore si verifica perché il token di accesso client (CAT) può essere utilizzato solo una volta dal proxy locale e quindi diventa non valido. Per risolvere questo errore, ruota il token di accesso client in modalità `SOURCE` per generare un nuovo CAT per l'origine. Per un esempio che mostra come ruotare il CAT di origine, consulta Esempio di rotazione del CAT di origine.

Errore di mancata corrispondenza del token client

> **Note**
>
> Non è consigliabile utilizzare i token client per riutilizzare il CAT. Ti consigliamo di utilizzare l'API `RotateTunnelAccessToken` per ruotare i token di accesso client per riconnetterti al tunnel.

Se usi i token client, puoi riutilizzare il CAT per riconnetterti al tunnel. Per riutilizzare il CAT, devi fornire al token client il CAT la prima volta che ti connetti al tunneling sicuro. Il tunneling sicuro memorizza il token client, quindi per i successivi tentativi di connessione con lo stesso token, è necessario fornire anche il token client. Per ulteriori informazioni sull'utilizzo dei token client, consulta l'implementazione di riferimento del proxy locale in. GitHub

Quando si utilizzano i token client, se usi un client in modalità di origine, è possibile che venga visualizzato il seguente errore:

```
Invalid client token: The provided client token does not match the client token
    that was previously set.
```

L'errore si verifica perché il token client fornito non corrisponde al token client fornito con il CAT quando si accede al tunnel. Per risolvere questo errore, ruota il CAT in modalità `SOURCE` per generare un nuovo CAT per l'origine. Di seguito viene riportato un esempio:

Esempio di rotazione del CAT di origine

Di seguito viene illustrato un esempio di come eseguire l'API `RotateTunnelAccessToken` in modalità `SOURCE` per generare un nuovo CAT per l'origine:

```
aws iotsecuretunneling rotate-tunnel-access-token \
```

```
    --region <region> \
    --tunnel-id <tunnel-id> \
    --client-mode SOURCE
```

L'esecuzione di questo comando genera un nuovo token di accesso di origine e restituisce l'ARN del tunnel.

```
{
    "sourceAccessToken": "<source-access-token>",
    "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/<tunnel-id>"
}
```

Ora puoi utilizzare il nuovo token di origine per connettere il proxy locale in modalità di origine.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=<source-access-token>
./localproxy -r <region> -s <port>
```

Di seguito è riportato un output di esempio dell'esecuzione del proxy locale:

```
...

[info]    Starting proxy in source mode
...
[info]    Successfully established websocket connection with proxy server ...
[info]    Listening for new connection on port <port>
...
```

Problemi di connettività del dispositivo remoto

Quando si utilizza il tunneling AWS IoT sicuro, il dispositivo potrebbe disconnettersi in modo imprevisto anche se il tunnel è aperto. Per identificare se un dispositivo è ancora connesso al tunnel, puoi utilizzare l'API o il DescribeTunneldescribe-tunnel. AWS CLI

Un dispositivo può essere disconnesso per diversi motivi. Per risolvere il problema di connettività, è possibile ruotare il CAT sulla destinazione se il dispositivo è stato disconnesso per i seguenti possibili motivi:

- Il CAT sulla destinazione è diventato non valido.
- Il token non è stato consegnato al dispositivo tramite l'argomento MQTT riservato del tunneling sicuro:

```
$aws/things/<thing-name>/tunnels/notify
```

Gli esempi seguenti mostrano come risolvere il problema:

Esempio di rotazione del CAT di destinazione

Considera il dispositivo remoto *<RemoteThing1>*. Per aprire un tunnel per quell'oggetto puoi usare il seguente comando:

```
aws iotsecuretunneling open-tunnel \
    --region <region> \
    --destination-config thingName=<RemoteThing1>,services=SSH
```

L'esecuzione di questo comando genera i dettagli del tunnel e il CAT per l'origine e la destinazione.

```
{
    "sourceAccessToken": "<source-access-token>",
    "destinationAccessToken": "<destination-access-token>",
    "tunnelId": "<tunnel-id>",
    "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/tunnel-id"
}
```

Tuttavia, quando si utilizza l'DescribeTunnelAPI, l'output indica che il dispositivo è stato disconnesso, come illustrato di seguito:

```
aws iotsecuretunneling describe-tunnel \
    --tunnel-id <tunnel-id> \
    --region <region>
```

L'esecuzione di questo comando indica che il dispositivo non è ancora connesso.

```
{
    "tunnel": {
        ...
        "destinationConnectionState": {
            "status": "DISCONNECTED"
        },
        ...
    }
```

```
}
```

Per risolvere questo errore, esegui l'API `RotateTunnelAccessToken` con il client in modalità `DESTINATION` e le configurazioni per la destinazione. L'esecuzione di questo comando revoca il vecchio token di accesso, genera un nuovo token e invia nuovamente questo token all'argomento MQTT:

`$aws/things/<thing-name>/tunnels/notify`

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --tunnel-id <tunnel-id> \
    --client-mode DESTINATION \
    --destination-config thingName=<RemoteThing1>,services=SSH \
    --region <region>
```

L'esecuzione di questo comando genera il nuovo token di accesso come mostrato di seguito. Il token viene quindi consegnato al dispositivo per connettersi al tunnel, se l'agente del dispositivo è configurato correttamente.

```
{
    "destinationAccessToken": "destination-access-token",
    "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

Provisioning dei dispositivi

AWS offre diversi modi per effettuare il provisioning di un dispositivo e installarvi certificati client univoci. Questa sezione descrive ogni modo e mostra come scegliere quello migliore per la tua soluzione IoT. Queste opzioni sono descritte in dettaglio nel whitepaper intitolato Produzione e provisioning dei dispositivi con certificati X.509 in AWS IoT Core.

Seleziona l'opzione più adatta alla tua situazione

- È possibile installare certificati sui dispositivi IoT prima che vengano consegnati

  Se è possibile installare in modo sicuro certificati client univoci sui dispositivi IoT prima che vengano consegnati per l'uso da parte dell'utente finale, è consigliabile utilizzare il just-in-timeprovisioning (JITP) o la just-in-timeregistrazione (JITR).

  Utilizzando JITP e JITR, l'autorità di certificazione (CA) utilizzata per firmare il certificato del dispositivo viene registrata e riconosciuta al momento della prima connessione AWS IoT del dispositivo. AWS IoT Il dispositivo viene fornito alla prima connessione utilizzando i dettagli del AWS IoT relativo modello di provisioning.

  Per ulteriori informazioni su single thing, JITP, JITR e provisioning in blocco di dispositivi con certificati univoci, consulta the section called “Provisioning dei dispositivi che dispongono di certificati dispositivo”.
- Gli utenti finali o gli installatori possono utilizzare un'app per installare certificati sui propri dispositivi IoT

  Se non è possibile installare in modo sicuro certificati client univoci sul dispositivo IoT prima che vengano recapitati all'utente finale, ma l'utente finale o un programma di installazione possono utilizzare un'app per registrare i dispositivi e installare i certificati univoci del dispositivo, è preferibile utilizzare il processo provisioning per utente attendibile.

  L'utilizzo di un utente attendibile, ad esempio un utente finale o un programma di installazione con un account noto, può semplificare il processo di produzione del dispositivo. Invece di un certificato client univoco, i dispositivi dispongono di un certificato temporaneo che consente al dispositivo di connettersi AWS IoT per soli 5 minuti. Durante la finestra di 5 minuti, l'utente attendibile ottiene un certificato client univoco con una durata maggiore e lo installa sul dispositivo. La durata limitata del certificato di reclamo riduce al minimo il rischio di un certificato compromesso.

Per ulteriori informazioni, consulta the section called “Provisioning tramite utente attendibile”.

- Gli utenti finali NON POSSONO utilizzare un'app per installare certificati sui propri dispositivi IoT

  Se nessuna delle opzioni precedenti funzionerà nella tua soluzione IoT, il processo provisioning per attestazione è un'opzione. Con questo processo, i dispositivi IoT dispongono di un certificato di attestazione condiviso da altri dispositivi del parco istanze. La prima volta che un dispositivo si connette con un certificato di AWS IoT attestazione, lo registra utilizzando il relativo modello di provisioning e rilascia al dispositivo il certificato client univoco a cui accedere successivamente. AWS IoT

  Questa opzione consente il provisioning automatico di un dispositivo al momento della connessione AWS IoT, ma potrebbe comportare un rischio maggiore in caso di compromissione del certificato di reclamo. Se un certificato di attestazione viene compromesso, è possibile disattivarlo. La disattivazione del certificato di attestazione impedisce a tutti i dispositivi con tale certificato di attestazione di essere registrati in futuro. Tuttavia, la disattivazione del certificato di attestazione non blocca i dispositivi di cui è già stato eseguito il provisioning.

  Per ulteriori informazioni, consulta the section called “Provisioning tramite attestazione”.

Fornitura dei dispositivi in AWS IoT

Quando si esegue il provisioning di un dispositivo AWS IoT, è necessario creare risorse in modo che i dispositivi AWS IoT possano comunicare in modo sicuro. È possibile creare altre risorse per aiutarti a gestire il parco istanze dispositivi. Durante il processo di provisioning è possibile creare le seguenti risorse:

- Un oggetto IoT.

  Gli oggetti IoT sono voci nel registro dei AWS IoT dispositivi. Ogni oggetto ha un nome univoco e un insieme di attributi ed è associato a un dispositivo fisico. Le cose possono essere definite utilizzando un tipo di oggetto o raggruppate in gruppi di oggetti. Per ulteriori informazioni, consulta Gestione dei dispositivi con AWS IoT.

  Sebbene non sia necessario, la creazione di un oggetto rende possibile gestire il parco istanze dei dispositivi in modo più efficace ricercando i dispositivi per tipo di oggetto, gruppo di oggetto e attributi di oggetto. Per ulteriori informazioni, consulta Indicizzazione del parco istanze.

> **Note**
>
> Per indicizzare i dati sullo stato di connettività dell'oggetto, esegui il provisioning dell'oggetto e configuralo in modo che il nome dell'oggetto corrisponda all'ID client utilizzato nella richiesta Connect.

- Un certificato X.509.

  I dispositivi utilizzano certificati X.509 per eseguire l'autenticazione reciproca con. AWS IoTÈ possibile registrare un certificato esistente o richiedere la AWS IoT generazione e la registrazione di un nuovo certificato. Un certificato viene associato a un dispositivo collegandolo all'oggetto che rappresenta il dispositivo. È inoltre necessario copiare il certificato e la chiave privata associata sul dispositivo. I dispositivi presentano il certificato quando si connettono a AWS IoT. Per ulteriori informazioni, consulta Autenticazione.
- Una policy IoT.

  Le policy IoT definiscono le operazioni che un dispositivo può eseguire in AWS IoT. Le policy IoT sono collegate ai certificati del dispositivo. Quando un dispositivo presenta il certificato a AWS IoT, gli vengono concesse le autorizzazioni specificate nella politica. Per ulteriori informazioni, consulta Autorizzazione. Ogni dispositivo richiede un certificato per comunicare con AWS IoT.

AWS IoT supporta il provisioning automatico del parco veicoli utilizzando modelli di provisioning. I modelli di provisioning descrivono le risorse AWS IoT necessarie per il provisioning del dispositivo. I modelli contengono variabili che consentono di utilizzare un modello per eseguire il provisioning di più dispositivi. Quando si esegue il provisioning di un dispositivo, si specificano i valori per le variabili specifiche del dispositivo utilizzando un dizionario o mappa. Per eseguire il provisioning di un altro dispositivo, specificare nuovi valori nel dizionario.

È possibile utilizzare il provisioning automatico indipendentemente dal fatto che i dispositivi abbiano certificati univoci e la chiave privata associata.

Approvvigionamento della flotta APIs

Esistono diverse categorie di prodotti APIs utilizzati nell'approvvigionamento della flotta:

- Queste funzioni del piano di controllo creano e gestiscono i modelli di provisioning del parco istanze e configurano policy utente attendibili.

  - CreateProvisioningTemplate
  - CreateProvisioningTemplateVersion
  - DeleteProvisioningTemplate
  - DeleteProvisioningTemplateVersion
  - DescribeProvisioningTemplate
  - DescribeProvisioningTemplateVersion
  - ListProvisioningTemplates
  - ListProvisioningTemplateVersions
  - UpdateProvisioningTemplate
- Gli utenti attendibili possono utilizzare questa funzione del piano di controllo per generare un'attestazione di onboarding temporanea. Questa attestazione temporanea viene passata al dispositivo durante la configurazione Wi-Fi o utilizzando un metodo analogo.
  - CreateProvisioningClaim
- API MQTT utilizzata durante il processo di provisioning dai dispositivi con un certificato di attestazione di provisioning incorporato in un dispositivo o passato a tale dispositivo da un utente attendibile.
  - the section called “CreateCertificateFromCsr”
  - the section called “CreateKeysAndCertificate”
  - the section called “RegisterThing”

Provisioning di dispositivi che non dispongono di certificati dispositivo mediante il provisioning del parco istanze dispositivi

Utilizzando il provisioning AWS IoT della flotta, AWS IoT puoi generare e fornire in modo sicuro certificati e chiavi private ai tuoi dispositivi quando si connettono AWS IoT per la prima volta. AWS IoT fornisce certificati client firmati dall'autorità di certificazione Amazon Root (CA).

Esistono due modi per utilizzare il provisioning del parco istanze dispositivi:

- Provisioning tramite attestazione
- Provisioning tramite utente attendibile

Provisioning tramite attestazione

I dispositivi possono essere fabbricati con un certificato di attestazione di provisioning e una chiave privata (che sono credenziali per scopi speciali) incorporati in essi. Se questi certificati sono registrati presso AWS IoT, il servizio può scambiarli con certificati di dispositivo unici che il dispositivo può utilizzare per le normali operazioni. Il processo include le seguenti fasi:

Prima di distribuire il dispositivo

1. Chiamare `CreateProvisioningTemplate` per creare un modello di provisioning. Questa API restituisce un ARN del modello. Per ulteriori informazioni, consulta API MQTT di provisioning del dispositivo.

   Puoi anche creare un modello di provisioning del parco veicoli nella AWS IoT console.

   a. Dal pannello di navigazione, scegli il menu a discesa Connetti molti dispositivi. Quindi, scegli Connect many devices.

   b. Scegli Crea modello di provisioning.

   c. Scegliete lo scenario di provisioning più adatto ai vostri processi di installazione. Quindi, seleziona Successivo.

   d. Completa il modello di workflow.

2. Creazione di certificati e chiavi private associate da utilizzare come certificati delle attestazioni di provisioning.

3. Registra questi certificati AWS IoT e associa una policy IoT che ne limiti l'uso. La policy di esempio IoT seguente limita l'uso del certificato associato a questa policy ai dispositivi di provisioning.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/$aws/certificates/create/
*",
    "arn:aws:iot:us-east-1:123456789012:topic/$aws/provisioning-
templates/templateName/provision/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/certificates/
create/*",
    "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/provisioning-
templates/templateName/provision/*"
            ]
        }
    ]
}
```

4. Concedi al AWS IoT servizio l'autorizzazione a creare o aggiornare risorse IoT come oggetti e certificati nel tuo account durante il provisioning dei dispositivi. A tale scopo, collega la policy `AWSIoTThingsRegistration` gestita a un ruolo IAM (chiamato ruolo di provisioning) che si fida del responsabile del servizio. AWS IoT
5. Il dispositivo viene prodotto con il certificato di attestazione di provisioning integrato in modo sicuro.

Il dispositivo è ora pronto per essere distribuito dove verrà installato per l'uso.

> **Important**
>
> Le chiavi private dell'attestazione di provisioning devono essere protette in ogni momento, anche sul dispositivo. Ti consigliamo di utilizzare AWS IoT CloudWatch metriche e log per monitorare eventuali indicazioni di uso improprio. Se si rileva un uso improprio, disattivare

il certificato di attestazione del provisioning in modo che non possa essere utilizzato per il provisioning del dispositivo.

Inizializzazione del dispositivo per l'utilizzo

1. Il dispositivo utilizza il AWS IoT Client per dispositivi SDKs SDKs, dispositivi mobili e AWS IoT dispositivi mobili per connettersi e autenticarsi AWS IoT utilizzando il certificato di provisioning claim installato sul dispositivo.

   > **Note**
   >
   > Per sicurezza, il valore `certificateOwnershipToken` restituito da `CreateCertificateFromCsr` e da `CreateKeysAndCertificate` scade dopo un'ora. `RegisterThing` deve essere chiamato prima della scadenza di `certificateOwnershipToken`. Se il certificato creato da `CreateCertificateFromCsr` o `CreateKeysAndCertificate` non è stato attivato e non è stato collegato a una policy o a un oggetto entro la scadenza del token, il certificato viene eliminato. Se il token scade, il dispositivo può chiamare nuovamente `CreateCertificateFromCsr` o `CreateKeysAndCertificate` per generare un nuovo certificato.

2. Il dispositivo ottiene un certificato permanente e una chiave privata utilizzando una di queste opzioni. Il dispositivo utilizzerà il certificato e la chiave per tutte le future autenticazioni con AWS IoT.

   a. Chiama `CreateKeysAndCertificate`per creare un nuovo certificato e una chiave privata utilizzando l'autorità di AWS certificazione.

      Or

   b. Chiama `CreateCertificateFromCsr` per generare un certificato da una richiesta di firma del certificato che mantenga la propria chiave privata protetta.

3. Dal dispositivo, richiamare `RegisterThing` per registrare il dispositivo con AWS IoT e creare risorse cloud.

   Il servizio di provisioning del parco istanze utilizza un modello di provisioning per definire e creare risorse cloud, come oggetti IoT. Il modello può specificare gli attributi e i gruppi a cui

appartiene l'oggetto. I gruppi di oggetti devono esistere prima di poter aggiungere il nuovo oggetto.

4. Dopo aver salvato il certificato permanente sul dispositivo, il dispositivo deve disconnettersi dalla sessione avviata con il certificato di attestazione di provisioning e riconnettersi utilizzando il certificato permanente.

Il dispositivo è ora pronto per comunicare normalmente con AWS IoT.

Provisioning tramite utente attendibile

In molti casi, un dispositivo si connette AWS IoT per la prima volta quando un utente affidabile, ad esempio un utente finale o un tecnico di installazione, utilizza un'app mobile per configurare il dispositivo nella posizione in cui è installato.

> **Important**
>
> Per eseguire questa procedura, è necessario gestire l'accesso e l'autorizzazione dell'utente attendibile. Un modo per eseguire questa operazione consiste nel fornire e gestire un account per l'utente attendibile che lo autentichi e conceda l'accesso alle funzionalità e alle API AWS IoT necessarie per eseguire questa procedura.

Prima di distribuire il dispositivo

1. Chiama `CreateProvisioningTemplate`per creare un modello di provisioning e restituirlo e. *`templateArn templateName`*
2. Creare un ruolo IAM utilizzato da un utente attendibile per avviare il processo di provisioning. Il modello di provisioning consente solo all'utente di eseguire il provisioning di un dispositivo. Esempio:

```
{
    "Effect": "Allow",
    "Action": [
        "iot:CreateProvisioningClaim"
    ],
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:provisioningtemplate/templateName"
    ]
```

```
}
```

3. Concedi al AWS IoT servizio l'autorizzazione a creare o aggiornare risorse IoT, come oggetti e certificati nel tuo account durante il provisioning dei dispositivi. A tale scopo, collega la policy `AWSIoTThingsRegistration` gestita a un ruolo IAM (chiamato ruolo di provisioning) che si fida del responsabile del servizio. AWS IoT
4. Fornisci i mezzi per identificare i tuoi utenti fidati, ad esempio fornendo loro un account in grado di autenticarli e autorizzare le loro interazioni con le operazioni AWS API necessarie per registrare i loro dispositivi.

Inizializzazione del dispositivo per l'utilizzo

1. Un utente attendibile accede all'app per dispositivi mobili o al servizio Web di provisioning.
2. L'applicazione per dispositivi mobili o l'applicazione Web utilizza il ruolo IAM e chiama `CreateProvisioningClaim` per ottenere un certificato di attestazione di provisioning temporaneo da AWS IoT.

> **Note**
>
> Per motivi di sicurezza, il certificato di attestazione di provisioning temporaneo restituito da `CreateProvisioningClaim` è valido solo per cinque minuti. Le fasi seguenti devono restituire un certificato valido prima della scadenza del certificato di attestazione di provisioning temporaneo. I certificati di attestazione di provisioning temporaneo non vengono visualizzati nell'elenco dei certificati dell'account.

3. L'app per dispositivi mobili o l'applicazione Web fornisce al dispositivo il certificato di attestazione di provisioning temporaneo insieme a tutte le informazioni di configurazione necessarie, ad esempio le credenziali Wi-Fi.
4. Il dispositivo utilizza il certificato di attestazione di provisioning temporaneo per connettersi a AWS IoT , utilizzando AWS IoT Client per dispositivi SDKs SDKs, dispositivi mobili e AWS IoT dispositivi mobili.
5. Il dispositivo ottiene un certificato permanente e una chiave privata utilizzando una di queste opzioni entro cinque minuti dalla connessione al certificato di richiesta AWS IoT di approvvigionamento temporaneo. Il dispositivo utilizzerà il certificato e la chiave restituiti da queste opzioni per tutte le future autenticazioni con AWS IoT.

a. Chiama `CreateKeysAndCertificate`per creare un nuovo certificato e una nuova chiave privata utilizzando l'autorità di AWS certificazione.

   Or

b. Chiama `CreateCertificateFromCsr` per generare un certificato da una richiesta di firma del certificato che mantenga la propria chiave privata protetta.

> **Note**
>
> Ricorda `CreateKeysAndCertificate`o `CreateCertificateFromCsr`devi restituire un certificato valido entro cinque minuti dalla connessione al AWS IoT certificato di richiesta di approvvigionamento temporaneo.

6. Il dispositivo chiama `RegisterThing`per registrare il dispositivo AWS IoT e creare risorse cloud.

   Il servizio di provisioning del parco istanze utilizza un modello di provisioning per definire e creare risorse cloud, come oggetti IoT. Il modello può specificare gli attributi e i gruppi a cui appartiene l'oggetto. I gruppi di oggetti devono esistere prima di poter aggiungere il nuovo oggetto.

7. Dopo aver salvato il certificato permanente sul dispositivo, il dispositivo deve disconnettersi dalla sessione avviata con il certificato di attestazione di provisioning temporaneo e riconnettersi utilizzando il certificato permanente.

Il dispositivo è ora pronto per comunicare normalmente con AWS IoT.

Utilizzo degli hook di pre-provisioning con la CLI AWS

La procedura seguente crea un modello di provisioning con hook di pre-provisioning. La funzione Lambda utilizzata qui è un esempio che può essere modificato.

Per creare e applicare un hook di pre-provisioning a un modello di provisioning

1. Creare una funzione Lambda con input e output definiti. Le funzioni Lambda sono altamente personalizzabili. `allowProvisioning` e `parameterOverrides` sono necessari per la creazione di hook pre-provisioning. Per ulteriori informazioni sulla creazione di funzioni Lambda, consulta Utilizzo AWS Lambda con l'interfaccia a riga di AWS comando.

Di seguito è riportato un esempio di output di una funzione Lambda:

```
{
  "allowProvisioning": True,
  "parameterOverrides": {
    "incomingKey0": "incomingValue0",
    "incomingKey1": "incomingValue1"
  }
}
```

2. AWS IoT utilizza politiche basate sulle risorse per chiamare Lambda, quindi è necessario autorizzare la chiamata AWS IoT alla funzione Lambda.

> **Important**
>
> Assicurati di includere il valore `source-arn` o `source-account` nelle chiavi del contesto delle condizioni globali delle policy associate all'operazione Lambda per impedire la manipolazione delle autorizzazioni. Per ulteriori informazioni, consulta Prevenzione del confused deputy tra servizi.

Di seguito è riportato un esempio che utilizza add-permission per concedere l'autorizzazione IoT all'utente Lambda.

```
aws lambda add-permission \
    --function-name myLambdaFunction \
    --statement-id iot-permission \
    --action lambda:InvokeFunction \
    --principal iot.amazonaws.com
```

3. Aggiungi un hook di pre-provisioning a un modello utilizzando il comando or. create-provisioning-templateupdate-provisioning-template

L'esempio CLI seguente utilizza create-provisioning-templateper creare un modello di provisioning con hook di pre-provisioning:

```
aws iot create-provisioning-template \
    --template-name myTemplate \
    --provisioning-role-arn arn:aws:iam:us-east-1:1234564789012:role/myRole \
    --template-body file://template.json \
```

```
    --pre-provisioning-hook file://hooks.json
```

L'output di questo comando è simile al seguente:

```
{
    "templateArn": "arn:aws:iot:us-east-1:1234564789012:provisioningtemplate/
myTemplate",
    "defaultVersionId": 1,
    "templateName": myTemplate
}
```

È in oltre possibile caricare un parametro da un file invece di digitarlo per intero come valore di parametro della riga di comando per risparmiare tempo. Per ulteriori informazioni, consulta Caricamento dei parametri da un file AWS CLI. Di seguito è riportato illustrato il parametro `template` in formato JSON espanso:

```
{
    "Parameters" : {
        "DeviceLocation": {
            "Type": "String"
        }
    },
    "Mappings": {
        "LocationTable": {
            "Seattle": {
                "LocationUrl": "https://example.aws"
            }
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "AttributePayload" : {
                    "version" : "v1",
                    "serialNumber" : "serialNumber"
                },
                "ThingName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingTypeName" : {"Fn::Join":["",["ThingTypePrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingGroups" : ["widgets", "WA"],
```

```
                "BillingGroup": "BillingGroup"
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
                "Status" : "Active",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Effect": "Allow",
                        "Action":["iot:Publish"],
                        "Resource": ["arn:aws:iot:us-east-1:504350838278:topic/foo/
bar"]
                    }]
                }
            }
        }
    },
    "DeviceConfiguration": {
        "FallbackUrl": "https://www.example.com/test-site",
        "LocationUrl": {
            "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"},
 "LocationUrl"]}
    }
}
```

Di seguito è riportato illustrato il parametro `pre-provisioning-hook` in formato JSON espanso:

```
{
```

```
    "targetArn" : "arn:aws:lambda:us-
east-1:765219403047:function:pre_provisioning_test",
    "payloadVersion" : "2020-04-01"
}
```

Provisioning dei dispositivi che dispongono di certificati dispositivo

AWS IoT fornisce tre modi per effettuare il provisioning dei dispositivi quando dispongono già di un certificato del dispositivo (e di una chiave privata associata):

- Il provisioning Single-thing con un modello di provisioning. Questa è una buona opzione se è sufficiente effettuare il provisioning di dispositivi uno alla volta.
- Just-in-time provisioning (JITP) con un modello che effettua il provisioning di un dispositivo alla prima connessione. AWS IoT Questa è una buona opzione se è necessario registrare un numero elevato di dispositivi, ma non si dispone di informazioni da assemblare in un elenco di provisioning in blocco.
- Registrazione in blocco Questa opzione consente di specificare un elenco di valori di modelli di provisioning single-thing memorizzati in un file in un bucket S3. Questo approccio è ideale se si dispone di un numero elevato di dispositivi noti le cui caratteristiche desiderate possono essere assemblate in un elenco.

Argomenti

- Provisioning di un singolo oggetto
- Just-in-time approvvigionamento
- Registrazione in blocco

Provisioning di un singolo oggetto

Per effettuare il provisioning di un oggetto, usa l'RegisterThingAPI o il `register-thing` comando CLI. Il comando `register-thing` dell'interfaccia a riga di comando accetta gli argomenti seguenti:

--template-body

Modello di provisioning.

--parameters

Elenco di coppie nome-valore per i parametri usati nel modello di provisioning, in formato JSON, ad esempio `{"ThingName" : "MyProvisionedThing", "CSR" : "csr-text"}`.

Per informazioni, consulta Modelli di provisioning.

RegisterThingo `register-thing` restituisce ARNs le risorse e il testo del certificato che ha creato:

```
{
    "certificatePem": "certificate-text",
    "resourceArns": {
    "PolicyLogicalName": "arn:aws:iot:us-
west-2:123456789012:policy/2A6577675B7CD1823E271C7AAD8184F44630FFD7",
    "certificate": "arn:aws:iot:us-west-2:123456789012:cert/
cd82bb924d4c6ccbb14986dcb4f40f30d892cc6b3ce7ad5008ed6542eea2b049",
    "thing": "arn:aws:iot:us-west-2:123456789012:thing/MyProvisionedThing"
    }
}
```

Se un parametro viene omesso dal dizionario, viene usato il valore predefinito. Se non è specificato alcun valore predefinito, il parametro non viene sostituito con un valore.

Just-in-time approvvigionamento

È possibile utilizzare il just-in-time provisioning (JITP) per effettuare il provisioning dei dispositivi al primo tentativo di connessione. AWS IoT Per eseguire il provisioning del dispositivo, devi abilitare la registrazione automatica e associare un modello di provisioning al certificato CA usato per firmare il certificato del dispositivo. I successi e gli errori di provisioning vengono registrati come in Parametri di provisioning dei dispositivi Amazon. CloudWatch

Argomenti

- Panoramica di JITP
- Registrare la CA utilizzando il modello di provisioning
- Registrare una CA utilizzando il nome del modello di provisioning

Panoramica di JITP

Quando un dispositivo tenta di connettersi AWS IoT utilizzando un certificato firmato da un certificato CA registrato, AWS IoT carica il modello dal certificato CA e lo utilizza per chiamare. RegisterThing Il flusso di lavoro JITP prima registra un certificato con un valore di stato `PENDING_ACTIVATION`. Quando il flusso di provisioning del dispositivo è completo, lo stato del certificato viene modificato in `ACTIVE`.

AWS IoT definisce i seguenti parametri a cui è possibile dichiarare e fare riferimento nei modelli di provisioning:

- `AWS::IoT::Certificate::Country`
- `AWS::IoT::Certificate::Organization`
- `AWS::IoT::Certificate::OrganizationalUnit`
- `AWS::IoT::Certificate::DistinguishedNameQualifier`
- `AWS::IoT::Certificate::StateName`
- `AWS::IoT::Certificate::CommonName`
- `AWS::IoT::Certificate::SerialNumber`
- `AWS::IoT::Certificate::Id`

I valori per questi parametri del modello di provisioning sono limitati a ciò che JITP può estrarre dal campo oggetto del certificato del dispositivo di cui viene effettuato il provisioning. Il certificato deve contenere valori per tutti i parametri nel corpo del modello. Il parametro `AWS::IoT::Certificate::Id` si riferisce a un ID generato internamente, non a un ID contenuto nel certificato. È possibile ottenere il valore di questo ID utilizzando la `principal()` funzione all'interno di una AWS IoT regola.

> **Note**
>
> È possibile effettuare il AWS IoT Core just-in-time provisioning dei dispositivi utilizzando la funzionalità di provisioning (JITP) senza dover inviare l'intera catena di fiducia alla prima connessione di un dispositivo a. AWS IoT Core La presentazione del certificato emesso da una CA è facoltativa, ma il dispositivo deve inviare l'estensione Server Name Indication (SNI) quando si collega a AWS IoT Core.

Modello di corpo di esempio

Il seguente file JSON è un esempio di modello di corpo completo di un modello JITP.

```
{
   "Parameters":{
      "AWS::IoT::Certificate::CommonName":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::SerialNumber":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Country":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Id":{
         "Type":"String"
      }
   },
   "Resources":{
      "thing":{
         "Type":"AWS::IoT::Thing",
         "Properties":{
            "ThingName":{
               "Ref":"AWS::IoT::Certificate::CommonName"
            },
            "AttributePayload":{
               "version":"v1",
               "serialNumber":{
                  "Ref":"AWS::IoT::Certificate::SerialNumber"
               }
            },
            "ThingTypeName":"lightBulb-versionA",
            "ThingGroups":[
               "v1-lightbulbs",
               {
                  "Ref":"AWS::IoT::Certificate::Country"
               }
            ]
         },
         "OverrideSettings":{
            "AttributePayload":"MERGE",
            "ThingTypeName":"REPLACE",
            "ThingGroups":"DO_NOTHING"
```

```
         }
      },
      "certificate":{
         "Type":"AWS::IoT::Certificate",
         "Properties":{
            "CertificateId":{
               "Ref":"AWS::IoT::Certificate::Id"
            },
            "Status":"ACTIVE"
         }
      },
      "policy":{
         "Type":"AWS::IoT::Policy",
         "Properties":{
            "PolicyDocument":"{ \"Version\": \"2012-10-17\",        \"Statement
\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
         }
      }
   }
}
```

Questo modello di esempio dichiara i valori per i parametri di provisioning `AWS::IoT::Certificate::CommonName`, `AWS::IoT::Certificate::SerialNumber`, `AWS::IoT::Certificate::Country` e `AWS::IoT::Certificate::Id` che vengono estratti dal certificato e utilizzati nella sezione `Resources`. Il flusso di lavoro JITP usa quindi questo modello per eseguire le seguenti operazioni:

- Registrare un certificato e impostare il relativo stato su PENDING_ACTIVE.
- Creare una risorsa oggetto.
- Creare una risorsa policy.
- Collegare la policy al certificato.
- Collegare il certificato all'oggetto.
- Aggiornare lo stato del certificato ad ACTIVE.

Il provisioning del dispositivo fallisce se il certificato non ha tutte le proprietà menzionate nella sezione del`Parameters`. `templateBody` Ad esempio, se `AWS::IoT::Certificate::Country` è incluso nel modello, ma il certificato non ha una proprietà `Country`, il provisioning del dispositivo non riesce.

Puoi anche usarlo CloudTrail per risolvere problemi con il tuo modello JITP. Per informazioni sulle metriche registrate in Amazon CloudWatch, consulta. Parametri di provisioning dei dispositivi Per ulteriori informazioni sui modelli di provisioning, consulta Modelli di provisioning.

> **Note**
>
> Durante il processo di provisioning, il just-in-time provisioning (JITP) richiama altre operazioni API del piano di controllo. AWS IoT Queste chiamate potrebbero superare le quote di throttling AWS IoT impostate per il tuo account e causare limitazioni delle chiamate. Se necessario, contatta il supporto clienti AWS per aumentare le quote di limitazione.

Registrare la CA utilizzando il modello di provisioning

Per registrare una CA utilizzando un modello di provisioning completo, attieniti alla seguente procedura:

1. Salva il modello di provisioning e le informazioni ARN del ruolo sotto forma di file JSON come nell'esempio seguente:

```
{
     "templateBody" : "{\r\n    \"Parameters\" : {\r\n
 \"AWS::IoT::Certificate::CommonName\": {\r\n           \"Type\": \"String\"\r
\n        },\r\n        \"AWS::IoT::Certificate::SerialNumber\": {\r\n
  \"Type\": \"String\"\r\n        },\r\n        \"AWS::IoT::Certificate::Country
\": {\r\n            \"Type\": \"String\"\r\n        },\r\n
 \"AWS::IoT::Certificate::Id\": {\r\n           \"Type\": \"String\"\r
\n        }\r\n    },\r\n    \"Resources\": {\r\n        \"thing\": {\r
\n            \"Type\": \"AWS::IoT::Thing\",\r\n           \"Properties
\": {\r\n               \"ThingName\": {\r\n                   \"Ref\":
 \"AWS::IoT::Certificate::CommonName\"\r\n               },\r\n
     \"AttributePayload\": {\r\n                   \"version\": \"v1\",\r\n
                  \"serialNumber\": {\r\n                       \"Ref\":
 \"AWS::IoT::Certificate::SerialNumber\"\r\n                   }\r\n
     },\r\n               \"ThingTypeName\": \"lightBulb-versionA\",\r\n
        \"ThingGroups\": [\r\n                   \"v1-lightbulbs\",\r\n
           {\r\n                       \"Ref\": \"AWS::IoT::Certificate::Country
\"\r\n                   }\r\n               ]\r\n           },\r\n
  \"OverrideSettings\": {\r\n               \"AttributePayload\": \"MERGE\",\r\n
               \"ThingTypeName\": \"REPLACE\",\r\n               \"ThingGroups
\": \"DO_NOTHING\"\r\n           }\r\n        },\r\n        \"certificate\": {\r
```

```
\n            \"Type\": \"AWS::IoT::Certificate\",\r\n            \"Properties
\": {\r\n                \"CertificateId\": {\r\n                    \"Ref\":
 \"AWS::IoT::Certificate::Id\"\r\n                },\r\n                \"Status\":
 \"ACTIVE\"\r\n            },\r\n            \"OverrideSettings\": {\r\n
     \"Status\": \"DO_NOTHING\"\r\n            }\r\n        },\r\n        \"policy
\": {\r\n            \"Type\": \"AWS::IoT::Policy\",\r\n            \"Properties
\": {\r\n                \"PolicyDocument\": \"{ \\\"Version\\\": \\\"2012-10-17
     \\\", \\\"Statement\\\": [{ \\\"Effect\\\": \\\"Allow\\\", \\\"Action\\\":[\\
\"iot:Publish\\\"], \\\"Resource\\\": [\\\"arn:aws:iot:us-east-1:123456789012:topic
\/foo\/bar\\\"] }] }\"\r\n            }\r\n        }\r\n    }\r\n}",
     "roleArn" : "arn:aws:iam::123456789012:role/JITPRole"
}
```

Il valore del campo `templateBody` deve essere un oggetto JSON specificato come stringa con caratteri di escape e può utilizzare solo i valori dell'elenco precedente. È possibile utilizzare una varietà di strumenti per creare l'output JSON richiesto, ad esempio `json.dumps` (Python) o `JSON.stringify` (Node). Il valore del campo `roleARN` deve essere l'ARN di un ruolo a cui è associata `AWSIoTThingsRegistration`. Inoltre, il modello può utilizzare un `PolicyName` esistente invece di `PolicyDocument` inline nell'esempio.

2. Registra un certificato CA con l'operazione Register CACertificate API o il comando `register-ca-certificate`CLI. Specificherai la cartella del modello di provisioning e le informazioni ARN sui ruoli salvate nel passaggio precedente:

   Di seguito viene illustrato un esempio di come registrare un certificato CA in modalità `DEFAULT` utilizzando la AWS CLI:

   ```
   aws iot register-ca-certificate --ca-certificate file://your-ca-cert --
   verification-cert file://your-verification-cert
                  --set-as-active --allow-auto-registration --registration-config
    file://your-template
   ```

   Di seguito viene illustrato un esempio di come registrare un certificato CA in modalità `SNI_ONLY` utilizzando la AWS CLI:

   ```
   aws iot register-ca-certificate --ca-certificate file://your-ca-cert --certificate-
   mode SNI_ONLY
                   --set-as-active --allow-auto-registration --registration-config
    file://your-template
   ```

Per ulteriori informazioni, consulta Registrazione di certificati CA.

3. (Facoltativo) Aggiorna le impostazioni per un certificato CA utilizzando l'operazione Update CACertificate API o il comando `update-ca-certificate`CLI.

   Di seguito viene illustrato un esempio di come aggiornare un certificato CA utilizzando la AWS CLI:

```
aws iot update-ca-certificate --certificate-id caCertificateId
                --new-auto-registration-status ENABLE --registration-config
 file://your-template
```

Registrare una CA utilizzando il nome del modello di provisioning

Per registrare una CA utilizzando il nome di un modello di provisioning, attieniti alla seguente procedura:

1. Salva il corpo del modello di provisioning sotto forma di file JSON. Puoi trovare un esempio di corpo del modello in esempio di corpo del modello.
2. Per creare un modello di provisioning, usa l'CreateProvisioningTemplateAPI o il comando `create-provisioning-template`CLI:

```
aws iot create-provisioning-template --template-name your-template-name \
        --template-body file://your-template-body.json --type JITP \
        --provisioning-role-arn arn:aws:iam::123456789012:role/test
```

> **Note**
>
> Per il just-in-time provisioning (JITP), è necessario specificare il tipo di modello da utilizzare al `JITP` momento della creazione del modello di provisioning. Per ulteriori informazioni sul tipo di modello, consulta CreateProvisioningTemplatel'API Reference.AWS

3. Per registrare CA con il nome del modello, utilizza l'CACertificateAPI Register o il `register-ca-certificate`comando CLI:

```
aws iot register-ca-certificate --ca-certificate file://your-ca-cert --
verification-cert file://your-verification-cert \
```

```
        --set-as-active --allow-auto-registration --registration-config
templateName=your-template-name
```

Registrazione in blocco

Puoi usare il comando `start-thing-registration-task` per registrare gli oggetti in blocco. Questo comando accetta un modello di provisioning, un nome di bucket S3, un nome di chiave e l'ARN di un ruolo che permette di accedere al file nel bucket S3. Il file nel bucket S3 contiene i valori usati per sostituire i parametri nel modello. Il file deve essere un file JSON delimitato da righe. Ogni riga contiene tutti i valori dei parametri per la registrazione di un singolo dispositivo. Ad esempio:

```
{"ThingName": "foo", "SerialNumber": "123", "CSR": "csr1"}
{"ThingName": "bar", "SerialNumber": "456", "CSR": "csr2"}
```

Le seguenti API relative alle operazioni di registrazione in blocco potrebbero essere utili:

- ListThingRegistrationTasks: elenca le attività correnti di bulk thing provisioning.
- DescribeThingRegistrationTask: fornisce informazioni su una specifica attività di registrazione di oggetti in blocco.
- StopThingRegistrationTask: interrompe un'attività di registrazione di oggetti in blocco.
- ListThingRegistrationTaskReports: Utilizzato per controllare i risultati e gli errori di un'attività di registrazione di oggetti in blocco.

> **Note**
>
> - Puoi eseguire una sola attività di registrazione in blocco per volta (per account).
> - Le operazioni di registrazione in blocco richiamano altre operazioni API AWS IoT del piano di controllo. Queste chiamate potrebbero superare le Quote di throttling AWS IoT del tuo account e causare errori di limitazione delle chiamate. Contatta l'assistenza AWS clienti per aumentare le quote di AWS IoT limitazione, se necessario.

Modelli di provisioning

Un modello di provisioning è un documento JSON che utilizza parametri per descrivere le risorse che il dispositivo deve utilizzare per interagire. AWS IoT Un modello di provisioning contiene due sezioni: `Parameters` e `Resources`. Esistono due tipi di modelli di provisioning in. AWS IoT Uno viene utilizzato per il just-in-time provisioning (JITP) e la registrazione in blocco, mentre il secondo viene utilizzato per l'approvvigionamento del parco veicoli.

Argomenti

- Sezione Parametri
- Sezione Risorse
- Esempio di modello per la registrazione in blocco
- Esempio di modello per il just-in-time provisioning (JITP)
- Provisioning del parco istanze

Sezione Parametri

La sezione `Parameters` dichiara i parametri usati all'interno della sezione `Resources`. Ogni parametro dichiara un nome, un tipo e un valore predefinito facoltativo. Il valore predefinito viene usato quando il dizionario passato con il modello non contiene un valore per il parametro. La sezione `Parameters` del documento di un modello è simile alla seguente:

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber" : {
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
        "CSR" : {
            "Type" : "String"
        }
    }
```

```
}
```

Il frammento di codice di questo corpo del modello dichiara quattro parametri: `ThingName`, `SerialNumber`, `Location` e `CSR`. Tutti questi parametri sono di tipo `String`. Il parametro `Location` dichiara un valore predefinito `"WA"`.

Sezione Risorse

La `Resources` sezione del corpo del modello dichiara le risorse necessarie per la comunicazione del dispositivo AWS IoT: un oggetto, un certificato e una o più policy IoT. Ogni risorsa specifica un nome logico, un tipo e un set di proprietà.

Un nome logico permette di fare riferimento a una risorsa in un'altra parte del modello.

Il tipo specifica il tipo di risorsa che intendi dichiarare. I tipi validi sono:

- `AWS::IoT::Thing`
- `AWS::IoT::Certificate`
- `AWS::IoT::Policy`

Le proprietà specificate dipendono dal tipo di risorsa dichiarato.

Risorse oggetto

Le risorse oggetto vengono dichiarate usando le proprietà seguenti:

- `ThingName`: stringa.
- `AttributePayload`: facoltativo. Un elenco di coppie nome-valore.
- `ThingTypeName`: facoltativo. Stringa per un tipo di oggetto associato per l'oggetto.
- `ThingGroups`: facoltativo. Elenco di gruppi cui appartiene l'oggetto.
- `BillingGroup`: facoltativo. Stringa per il nome di un gruppo di fatturazione associato.
- `PackageVersions`: facoltativo. Stringa per un pacchetto associato e nomi delle versioni.

Risorse certificato

È possibile specificare i certificati in uno dei modi seguenti:

- Richiesta di firma del certificato.
- ID certificato di un certificato del dispositivo esistente. (Solo il certificato IDs può essere utilizzato con un modello di provisioning del parco veicoli.)
- Certificato del dispositivo creato con un certificato CA registrato con AWS IoT. Se esistono più certificati CA registrati con lo stesso campo dell'oggetto, devi passare anche il certificato CA usato per firmare il certificato del dispositivo.

> **Note**
>
> Quando dichiari un certificato in un modello, usa solo uno di questi metodi. Ad esempio, se usi una richiesta di firma del certificato, non potrai specificare anche un ID certificato o un certificato del dispositivo. Per ulteriori informazioni, consulta Certificati client X.509.

Per ulteriori informazioni, consulta Panoramica sui certificati X.509.

Le risorse certificato vengono dichiarate usando le proprietà seguenti:

- `CertificateSigningRequest`: stringa.
- `CertificateId`: stringa.
- `CertificatePem`: stringa.
- `CACertificatePem`: stringa.
- `Status`: facoltativo. Stringa che può essere `ACTIVE` o `INACTIVE`. Il valore predefinito è ACTIVE.
- `ThingPrincipalType`: facoltativo. Stringa che specifica il tipo di relazione tra l'oggetto e il principale (il certificato).
  - `EXCLUSIVE_THING`: stabilisce una relazione esclusiva. Il principale può essere associato solo a questo oggetto specifico e non ad altri.
  - `NON_EXCLUSIVE_THING`: associa il principio specificato agli oggetti. È possibile allegare più oggetti al principale. Se non specificato, questo è il valore predefinito.

> **Note**
>
> È inoltre possibile effettuare il provisioning dei dispositivi senza certificati dei dispositivi. Per ulteriori informazioni, consulta Provisioning di dispositivi che non dispongono di certificati dispositivo mediante il provisioning del parco istanze dispositivi.

Esempi:

- Certificato specificato con una richiesta di firma del certificato (CSR):

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CertificateSigningRequest": {"Ref" : "CSR"},
            "Status" : "ACTIVE"
        }
    }
}
```

- Certificato specificato con un ID certificato esistente:

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CertificateId": {"Ref" : "CertificateId"}
        }
    }
}
```

- Certificate specificato con un .pem del certificato esistente e un .pem del certificato CA:

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CACertificatePem": {"Ref" : "CACertificatePem"},
            "CertificatePem": {"Ref" : "CertificatePem"}
        }
    }
}
```

- Associa esclusivamente una cosa a un principale:

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
```

```
            "ThingPrincipalType" : "EXCLUSIVE_THING"
        }
    }
}
```

Risorse relative alle policy

Le risorse policy vengono dichiarate con una delle seguenti proprietà:

- `PolicyName`: facoltativo. Stringa. Il valore predefinito è un hash del documento della policy. La `PolicyName` può solo fare riferimento alle policy AWS IoT ma non policy IAM. Se utilizzi una policy AWS IoT esistente, immetti il nome della policy per la proprietà `PolicyName`. Non includere la proprietà `PolicyDocument`.
- `PolicyDocument`: facoltativo. Un oggetto JSON specificato come stringa con carattere di escape. Se la proprietà `PolicyDocument` non è specificata, la policy deve essere già stata creata.

> **Note**
>
> Se è presente una sezione `Policy`, è necessario specificare la proprietà `PolicyName` o `PolicyDocument`, ma non entrambe.

Sostituzione delle impostazioni

Se un modello specifica una risorsa già esistente, la sezione `OverrideSettings` permette di specificare l'operazione da eseguire:

`DO_NOTHING`

Lascia la risorsa inalterata.

`REPLACE`

Sostituisce la risorsa con quella specificata nel modello.

`FAIL`

La richiesta non riesce con `ResourceConflictsException`.

`MERGE`

Valido solo per le proprietà `ThingGroups` e `AttributePayload` di una risorsa `thing`. Unisce gli attributi o le appartenenze ai gruppi esistenti dell'oggetto a quelli specificati nel modello.

Quando si dichiara una risorsa oggetto, è possibile specificare `OverrideSettings` per le seguenti proprietà:

- `ATTRIBUTE_PAYLOAD`
- `THING_TYPE_NAME`
- `THING_GROUPS`

Quando si dichiara una risorsa certificato, è possibile specificare `OverrideSettings` per la proprietà `Status`.

`OverrideSettings` non sono disponibili per le risorse policy.

Esempi di risorsa

Il frammento di codice del modello seguente dichiara un oggetto, un certificato e una policy:

```
{
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "ThingName" : {"Ref" : "ThingName"},
                "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
                "ThingTypeName" :  "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
```

```
                "CertificateSigningRequest": {"Ref" : "CSR"},
                "Status" : "ACTIVE"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : "{ \"Version\": \"2012-10-17\",
 \"Statement\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
            }
        }
    }
}
```

L'oggetto viene dichiarato con:

- Nome logico `"thing"`.
- Tipo `AWS::IoT::Thing`.
- Un set di proprietà dell'oggetto.

  Le proprietà dell'oggetto includono il nome dell'oggetto, un set di attributi, un nome di tipo di oggetto facoltativo e un elenco facoltativo di gruppi di oggetti cui appartiene l'oggetto.

Ai parametri viene fatto riferimento tramite `{"Ref":"`*`parameter-name`*`"}`. Quando il modello viene valutato, i parametri vengono sostituiti con il valore del parametro specificato nel dizionario passato con il modello.

Il certificato viene dichiarato con:

- Nome logico `"certificate"`.
- Tipo `AWS::IoT::Certificate`.
- Un set di proprietà.

  Le proprietà includono la richiesta di firma per il certificato e l'impostazione dello stato su `ACTIVE`. Il testo della richiesta di firma del certificato viene passato come parametro nel dizionario a sua volta passato con il modello.

La policy viene dichiarata con:

- Nome logico `"policy"`.
- Tipo `AWS::IoT::Policy`.
- Nome di una policy esistente o di un documento di policy.

Esempio di modello per la registrazione in blocco

Il seguente file JSON è un esempio di un modello di provisioning completo che specifica il certificato con una CSR:

(Il valore del campo `PolicyDocument` deve essere un oggetto JSON specificato come stringa con carattere escape).

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber" : {
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
        "CSR" : {
            "Type" : "String"
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "ThingName" : {"Ref" : "ThingName"},
                "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
                "ThingTypeName" :  "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
```

```
            "Properties" : {
                "CertificateSigningRequest": {"Ref" : "CSR"},
                "Status" : "ACTIVE",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : "{ \"Version\": \"2012-10-17\",
 \"Statement\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
            }
        }
    }
}
```

Esempio di modello per il just-in-time provisioning (JITP)

Il seguente file JSON è un esempio di un modello di provisioning completo che specifica un certificato esistente con un ID certificato:

```
{
   "Parameters":{
      "AWS::IoT::Certificate::CommonName":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::SerialNumber":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Country":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Id":{
         "Type":"String"
      }
   },
   "Resources":{
      "thing":{
         "Type":"AWS::IoT::Thing",
         "Properties":{
            "ThingName":{
               "Ref":"AWS::IoT::Certificate::CommonName"
```

```
            },
            "AttributePayload":{
               "version":"v1",
               "serialNumber":{
                  "Ref":"AWS::IoT::Certificate::SerialNumber"
               }
            },
            "ThingTypeName":"lightBulb-versionA",
            "ThingGroups":[
               "v1-lightbulbs",
               {
                  "Ref":"AWS::IoT::Certificate::Country"
               }
            ]
         },
         "OverrideSettings":{
            "AttributePayload":"MERGE",
            "ThingTypeName":"REPLACE",
            "ThingGroups":"DO_NOTHING"
         }
      },
      "certificate":{
         "Type":"AWS::IoT::Certificate",
         "Properties":{
            "CertificateId":{
               "Ref":"AWS::IoT::Certificate::Id"
            },
            "Status":"ACTIVE",
            "ThingPrincipalType" : "EXCLUSIVE_THING"
         }
      },
      "policy":{
         "Type":"AWS::IoT::Policy",
         "Properties":{
            "PolicyDocument":"{ \"Version\": \"2012-10-17\",        \"Statement
\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
         }
      }
   }
}
```

> **Important**
>
> Devi utilizzare `CertificateId` in un modello utilizzato per il provisioning JIT.

Per ulteriori informazioni sul tipo di modello di provisioning, consulta il riferimento `CreateProvisioningTemplate`all'API. AWS

Per ulteriori informazioni su come utilizzare questo modello per il just-in-time provisioning, vedere: Just-in-time provisioning.

Provisioning del parco istanze

I modelli di provisioning della flotta vengono utilizzati da AWS IoT per configurare la configurazione del cloud e dei dispositivi. Questi modelli utilizzano gli stessi parametri e risorse dei modelli di registrazione JITP e in blocco. Per ulteriori informazioni, consulta Modelli di provisioning. I modelli di provisioning del parco istanze dispositivi possono contenere una sezione `Mapping` e una sezione `DeviceConfiguration`. È possibile utilizzare funzioni intrinseche all'interno di un modello di provisioning del parco istanze dei dispositivi per generare una configurazione specifica del dispositivo. I modelli di approvvigionamento della flotta sono risorse denominate e sono identificati da ARNs (ad esempio,`arn:aws:iot:us-west-2:1234568788:provisioningtemplate/`*`templateName`*).

Mappature

La sezione `Mappings` opzionale abbina una chiave a un corrispondente set di valori denominati. Ad esempio, se si desidera impostare valori basati su una AWS regione, è possibile creare una mappatura che utilizzi il Regione AWS nome come chiave e contenga i valori che si desidera specificare per ogni regione specifica. Utilizzi quindi la funzione intrinseca `Fn::FindInMap` per recuperare i valori in una mappa.

Non puoi includere parametri, pseudoparametri o funzioni intrinseche di chiamata nella sezione `Mappings`.

Configurazione del dispositivo

La sezione di configurazione del dispositivo contiene dati arbitrari che si desidera inviare ai dispositivi durante il provisioning. Esempio:

```
{
```

```
    "DeviceConfiguration": {
        "Foo":"Bar"
    }
}
```

Se invii messaggi ai tuoi dispositivi utilizzando il formato di payload JavaScript Object Notation (JSON), AWS IoT Core formatta questi dati come JSON. Se si utilizza il formato di payload Concise Binary Object Representation (CBOR), AWS IoT Core formatta questi dati come CBOR. La sezione `DeviceConfiguration` non supporta gli oggetti JSON nidificati.

Funzioni intrinseche

Le funzioni intrinseche vengono utilizzate in qualsiasi sezione del modello di provisioning ad eccezione della sezione `Mappings`.

`Fn::Join`

Aggiunge un set di valori in un singolo valore, separato dal delimitatore specificato. Se un delimitatore è una stringa vuota, i valori sono concatenati senza alcun delimitatore.

> **Important**
>
> `Fn::Join` non è supportato per the section called “Risorse relative alle policy”.

`Fn::Select`

Restituisce un singolo oggetto da un elenco di oggetti per indice.

> **Important**
>
> `Fn::Select` non verifica la presenza di valori `null` o se l'indice è fuori dai limiti dell'array. Entrambe le condizioni generano un errore di provisioning, quindi assicurati di aver scelto un valore di indice valido e che l'elenco contenga valori non nulli.

`Fn::FindInMap`

Restituisce i valori corrispondenti alle chiavi in una mappatura a due livelli dichiarata nella sezione `Mappings`.

`Fn::Split`

Divide una stringa in un elenco di valori stringa in modo da poter selezionare un elemento dall'elenco di stringhe. Si specifica un delimitatore che stabilisce dove è suddivisa la stringa (ad esempio, una virgola). Dopo aver suddiviso una stringa, utilizzare `Fn::Select` per selezionare un elemento.

Ad esempio, se nel modello di stack IDs viene importata una stringa di sottorete delimitata da virgole, potete dividere la stringa per ogni virgola. Dall'elenco delle sottoreti IDs, utilizzare per specificare un ID di sottorete `Fn::Select` per una risorsa.

`Fn::Sub`

Sostituisce le variabili in una stringa di input con valori che puoi specificare. Questa funzione è utilizzabile per costruire comandi o output che includono valori non disponibili finché non crei o aggiorni uno stack.

Esempio di modello per il provisioning del parco istanze dei dispositivi

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber": {
            "Type": "String"
        },
        "DeviceLocation": {
            "Type": "String"
        }
    },
    "Mappings": {
        "LocationTable": {
            "Seattle": {
                "LocationUrl": "https://example.aws"
            }
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
```

```
                "AttributePayload" : {
                    "version" : "v1",
                    "serialNumber" : "serialNumber"
                },
                "ThingName" : {"Ref" : "ThingName"},
                "ThingTypeName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingGroups" : ["v1-lightbulbs", "WA"],
                "BillingGroup": "LightBulbBillingGroup"
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
                "Status" : "Active",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Effect": "Allow",
                        "Action":["iot:Publish"],
                        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/foo/
bar"]
                    }]
                }
            }
        }
    },
    "DeviceConfiguration": {
        "FallbackUrl": "https://www.example.com/test-site",
        "LocationUrl": {
            "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"},
 "LocationUrl"]}
```

```
        }
}
```

> **Note**
>
> È possibile aggiornare un modello di provisioning esistente per aggiungere un hook di pre-provisioning.

Hook di pre-provisioning

AWS consiglia di utilizzare le funzioni hook di pre-provisioning durante la creazione di modelli di provisioning per consentire un maggiore controllo su quali e quanti dispositivi sono installati dall'account. Gli hook di pre-provisioning sono funzioni Lambda che convalidano i parametri passati dal dispositivo prima di consentire il provisioning dello stesso. Questa funzione Lambda deve esistere nell'account prima di effettuare il provisioning di un dispositivo perché viene chiamata ogni volta che un dispositivo invia una richiesta tramite the section called “RegisterThing”.

> **Important**
>
> Assicurati di includere il valore `source-arn` o `source-account` nelle chiavi del contesto delle condizioni globali delle policy associate all'operazione Lambda per impedire la manipolazione delle autorizzazioni. Per ulteriori informazioni, consulta Prevenzione del confused deputy tra servizi.

Per il provisioning dei dispositivi, la funzione Lambda deve accettare l'oggetto di input e restituire l'oggetto di output descritto in questa sezione. Il provisioning procede solo se la funzione Lambda restituisce un oggetto con `"allowProvisioning": True`.

Input dell'hook di pre-provisioning

AWS IoT invia questo oggetto alla funzione Lambda quando un dispositivo si registra con. AWS IoT

```
{
    "claimCertificateId" : "string",
    "certificateId" : "string",
    "certificatePem" : "string",
```

```
    "templateArn" : "arn:aws:iot:us-east-1:1234567890:provisioningtemplate/MyTemplate",
    "clientId" : "221a6d10-9c7f-42f1-9153-e52e6fc869c1",
    "parameters" : {
        "string" : "string",
        ...
    }
}
```

L'oggetto `parameters` passato alla funzione Lambda contiene le proprietà nell'argomento `parameters` passato nel payload della richiesta the section called “RegisterThing”.

Valore restituito dall'hook di pre-provisioning

La funzione Lambda deve restituire una risposta che indica se ha autorizzato la richiesta di provisioning e i valori di qualsiasi proprietà da sovrascrivere.

Di seguito è riportato un esempio di risposta riuscita dalla funzione di pre-provisioning.

```
{
    "allowProvisioning": true,
    "parameterOverrides" : {
        "Key": "newCustomValue",
        ...
    }
}
```

I valori `"parameterOverrides"` verranno aggiunti al parametro `"parameters"` nella richiesta di payload the section called “RegisterThing”.

> **Note**
>
> - Se la funzione Lambda fallisce, la richiesta di provisioning fallisce `ACCESS_DENIED` e viene registrato un errore in Logs. CloudWatch
> - Se la funzione Lambda non restituisce `"allowProvisioning": "true"` nella risposta, la richiesta di provisioning fallisce restituendo `ACCESS_DENIED`.
> - La funzione Lambda deve terminare l'esecuzione e restituire il valore entro 5 secondi, altrimenti la richiesta di provisioning non riesce.

Esempio di hook di pre-provisioning Lambda

Python

Un esempio di hook di pre-provisioning Lambda in Python.

```
import json

def pre_provisioning_hook(event, context):
    print(event)

    return {
        'allowProvisioning': True,
        'parameterOverrides': {
            'DeviceLocation': 'Seattle'
        }
    }
```

Java

Un esempio di hook di pre-provisioning Lambda in Java.

Classe gestore:

```
package example;

import java.util.Map;
import java.util.HashMap;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class PreProvisioningHook implements
 RequestHandler<PreProvisioningHookRequest, PreProvisioningHookResponse> {

    public PreProvisioningHookResponse handleRequest(PreProvisioningHookRequest
 object, Context context) {
        Map<String, String> parameterOverrides = new HashMap<String, String>();
        parameterOverrides.put("DeviceLocation", "Seattle");

        PreProvisioningHookResponse response = PreProvisioningHookResponse.builder()
                .allowProvisioning(true)
                .parameterOverrides(parameterOverrides)
                .build();
```

```
        return response;
    }

}
```

Classe richiesta:

```
package example;

import java.util.Map;
import lombok.Builder;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;

@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class PreProvisioningHookRequest {
    private String claimCertificateId;
    private String certificateId;
    private String certificatePem;
    private String templateArn;
    private String clientId;
    private Map<String, String> parameters;
}
```

Classe risposta:

```
package example;

import java.util.Map;
import lombok.Builder;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;


@Data
@Builder
@AllArgsConstructor
```

```
@NoArgsConstructor
public class PreProvisioningHookResponse {
    private boolean allowProvisioning;
    private Map<String, String> parameterOverrides;
}
```

JavaScript

Un esempio di hook di pre-provisioning Lambda in. JavaScript

```
exports.handler = function(event, context, callback) {
    console.log(JSON.stringify(event, null, 2));
    var reply = {
        allowProvisioning: true,
        parameterOverrides: {
            DeviceLocation: 'Seattle'
        }
     };
     callback(null, reply);
}
```

Firma dei certificati autogestita tramite provider di certificati AWS IoT Core

È possibile creare un fornitore di AWS IoT Core certificati per firmare le richieste di firma dei certificati (CSRs) in AWS IoT Fleet Provisioning. Un fornitore di certificati fa riferimento a una funzione Lambda e all'API `CreateCertificateFromCsr` MQTT per il provisioning della flotta. La funzione Lambda accetta un CSR e restituisce un certificato client firmato.

Quando non disponete di un fornitore di certificati Account AWS, l'API CreateCertificateFromCsr MQTT viene richiamata in Fleet Provisioning per generare il certificato a partire da una CSR. Dopo aver creato un fornitore di certificati, il comportamento dell'API CreateCertificateFromCsr MQTT cambierà e tutte le chiamate a questa API MQTT richiameranno il fornitore del certificato per emettere il certificato.

Con il fornitore di AWS IoT Core certificati, è possibile implementare soluzioni che utilizzano autorità di certificazione private (CAs) AWS Private CA, ad esempio altre autorità di fiducia CAs pubblica o la propria infrastruttura a chiave pubblica (PKI) per firmare la CSR. Inoltre, è possibile utilizzare il

fornitore di certificati per personalizzare i campi del certificato client, ad esempio periodi di validità, algoritmi di firma, emittenti ed estensioni.

> **Important**
>
> È possibile creare un solo fornitore di certificati per. Account AWS La modifica del comportamento di firma si applica all'intera flotta che chiama l'API CreateCertificateFromCsr MQTT fino a quando non si elimina il fornitore di certificati dal proprio Account AWS.

In questo argomento:

- Come funziona la firma dei certificati autogestita nel provisioning del parco veicoli
- Input della funzione Lambda del fornitore di certificati
- Valore restituito dalla funzione Lambda del fornitore di certificati
- Funzione Lambda di esempio
- Firma dei certificati autogestita per il rifornimento del parco veicoli
- AWS CLI comandi per il fornitore di certificati

Come funziona la firma dei certificati autogestita nel provisioning del parco veicoli

Concetti chiave

I seguenti concetti forniscono dettagli che possono aiutarti a capire come funziona la firma dei certificati autogestita nel AWS IoT provisioning del parco veicoli. Per ulteriori informazioni, consulta Provisioning dei dispositivi che non dispongono di certificati di dispositivo utilizzando il provisioning del parco veicoli.

AWS IoT approvvigionamento del parco veicoli

> Con AWS IoT Fleet Provisioning (abbreviazione di fleet provisioning), AWS IoT Core genera e consegna in modo sicuro i certificati dei dispositivi ai tuoi dispositivi quando si connettono AWS IoT Core per la prima volta. Puoi utilizzare il provisioning del parco veicoli per connettere dispositivi che non dispongono di certificati di dispositivo a. AWS IoT Core

Richiesta di firma del certificato (CSR)

Nel processo di approvvigionamento della flotta, un dispositivo invia una richiesta AWS IoT Core tramite il sistema MQTT di approvvigionamento della flotta. APIs Questa richiesta include una richiesta di firma del certificato (CSR), che verrà firmata per creare un certificato client.

AWS firma gestita dei certificati nell'ambito del provisioning della flotta

AWS managed è l'impostazione predefinita per la firma dei certificati nel provisioning del parco veicoli. Con la firma AWS gestita dei certificati, AWS IoT Core firmerà CSRs utilizzando la propria CAs firma.

Firma dei certificati gestita automaticamente durante il provisioning del parco veicoli

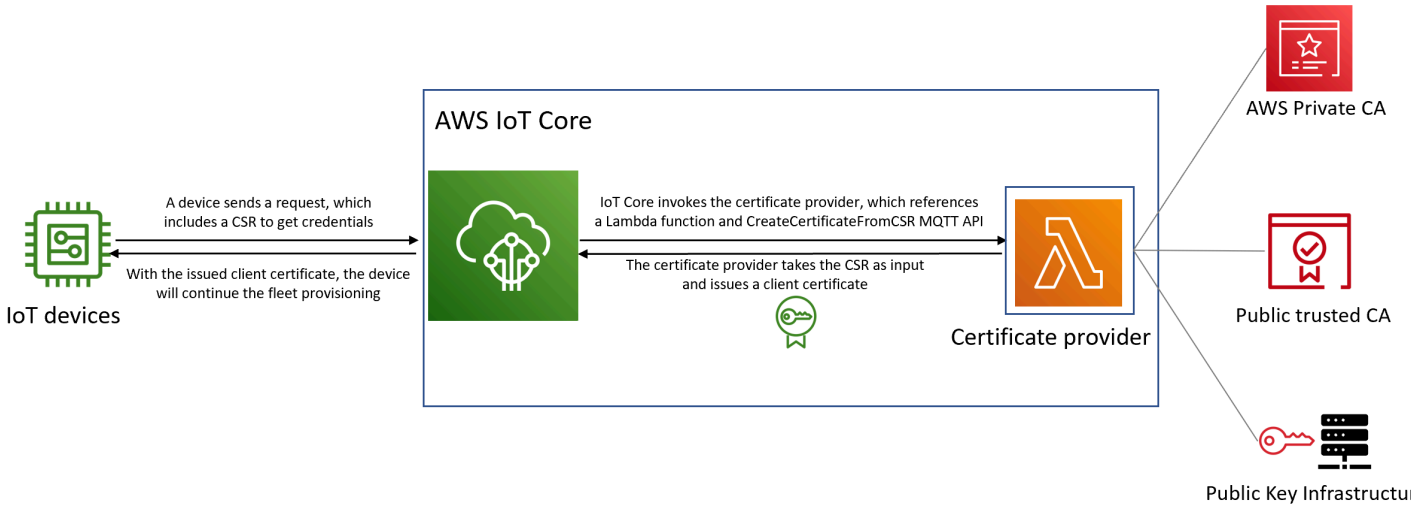
L'autogestione è un'altra opzione per la firma dei certificati nell'ambito del provisioning del parco veicoli. Con la firma dei certificati autogestita, puoi creare un fornitore di AWS IoT Core certificati da firmare. CSRs Puoi utilizzare la firma dei certificati autogestita per firmare CSRs con una CA generata da una CA AWS privata, da un'altra CA pubblicamente affidabile o dalla tua infrastruttura a chiave pubblica (PKI).

AWS IoT Core fornitore di certificati

AWS IoT Core certificate provider (abbreviazione di certificate provider) è una risorsa gestita dal cliente che viene utilizzata per la firma dei certificati autogestita nel provisioning del parco veicoli.

Diagramma

Il diagramma seguente è un'illustrazione semplificata di come funziona la firma degli autocertificati nell'approvvigionamento del parco veicoli. AWS IoT



- Quando un nuovo dispositivo IoT viene prodotto o introdotto nella flotta, ha bisogno di certificati client con AWS IoT Core cui autenticarsi.
- Come parte del processo di approvvigionamento della flotta, il dispositivo effettua una richiesta di certificati client tramite il Fleet Provisioning MQTT. AWS IoT Core APIs Questa richiesta include una richiesta di firma del certificato (CSR).
- AWS IoT Core richiama il fornitore del certificato e trasmette la CSR come input al provider.
- Il fornitore del certificato accetta la CSR come input ed emette un certificato client.

  Per la firma AWS gestita dei certificati, AWS IoT Core firma la CSR utilizzando la propria CA ed emette un certificato client.
- Con il certificato client emesso, il dispositivo continuerà il provisioning del parco veicoli e stabilirà una connessione sicura con. AWS IoT Core

Input della funzione Lambda del fornitore di certificati

AWS IoT Core invia il seguente oggetto alla funzione Lambda quando un dispositivo si registra con essa. Il valore di `certificateSigningRequest` è la CSR in formato Privacy-Enhanced Mail (PEM) fornita nella richiesta. `CreateCertificateFromCsr` `principalId`È l'ID del principale a cui ci si connette quando si effettua la AWS IoT Core richiesta. `CreateCertificateFromCsr` `clientId`è l'ID client impostato per la connessione MQTT.

```
{
 "certificateSigningRequest": "string",
 "principalId": "string",
 "clientId": "string"
}
```

Valore restituito dalla funzione Lambda del fornitore di certificati

La funzione Lambda deve restituire una risposta che contenga il `certificatePem` valore. Di seguito è riportato un esempio di risposta riuscita. AWS IoT Core utilizzerà il valore restituito (`certificatePem`) per creare il certificato.

```
{
 "certificatePem": "string"
}
```

Se la registrazione va a buon fine, `CreateCertificateFromCsr` restituirà lo stesso `certificatePem` nella `CreateCertificateFromCsr` risposta. Per ulteriori informazioni, vedere l'esempio del payload di risposta di CreateCertificateFromCsr.

Funzione Lambda di esempio

Prima di creare un fornitore di certificati, devi creare una funzione Lambda per firmare una CSR. Di seguito è riportato un esempio di funzione Lambda in Python. Questa funzione chiama AWS Private CA per firmare l'input CSR, utilizzando una CA privata e l'`SHA256WITHRSA`algoritmo di firma. Il certificato client restituito sarà valido per un anno. Per ulteriori informazioni AWS Private CA e su come creare una CA privata, consulta Cos'è una CA AWS privata? e Creazione di una CA privata.

```
import os
import time
import uuid
import boto3

def lambda_handler(event, context):
    ca_arn = os.environ['CA_ARN']
    csr = (event['certificateSigningRequest']).encode('utf-8')

    acmpca = boto3.client('acm-pca')
    cert_arn = acmpca.issue_certificate(
        CertificateAuthorityArn=ca_arn,
        Csr=csr,
        Validity={"Type": "DAYS", "Value": 365},
        SigningAlgorithm='SHA256WITHRSA',
        IdempotencyToken=str(uuid.uuid4())
    )['CertificateArn']

    # Wait for certificate to be issued
    time.sleep(1)
    cert_pem = acmpca.get_certificate(
        CertificateAuthorityArn=ca_arn,
        CertificateArn=cert_arn
    )['Certificate']

    return {
        'certificatePem': cert_pem
    }
```

Important

- I certificati restituiti dalla funzione Lambda devono avere lo stesso nome dell'oggetto e la stessa chiave pubblica della Certificate Signing Request (CSR).
- L'esecuzione della funzione Lambda deve terminare entro 5 secondi.
- La funzione Lambda deve trovarsi nella stessa regione della Account AWS risorsa del fornitore di certificati.
- Al responsabile del AWS IoT servizio deve essere concessa l'autorizzazione di invoca la funzione Lambda. Per evitare confusi problemi con l'assistente, ti consigliamo di impostare `sourceArn` e `sourceAccount` per i permessi di invoca. Per ulteriori informazioni consulta la pagina relativa alla prevenzione del problema "confused deputy" tra servizi.

Il seguente esempio di policy basata sulle risorse per Lambda concede l' AWS IoT autorizzazione a richiamare la funzione Lambda:

```
{
 "Version":"2012-10-17",
 "Id": "InvokePermission",
 "Statement": [
  {
   "Sid": "LambdaAllowIotProvider",
   "Effect": "Allow",
   "Principal": {
    "Service": "iot.amazonaws.com"
   },
   "Action": "lambda:InvokeFunction",
   "Resource": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
   "Condition": {
    "StringEquals": {
     "AWS:SourceAccount": "123456789012"
    },
    "ArnLike": {
    "AWS:SourceArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider/my-
certificate-provider"
    }
   }
  }
 ]
```

```
}
```

Firma dei certificati autogestita per il rifornimento del parco veicoli

Puoi scegliere la firma dei certificati autogestita per il rifornimento della flotta utilizzando o. AWS CLI Console di gestione AWS

AWS CLI

Per scegliere la firma dei certificati autogestita, è necessario creare un fornitore di AWS IoT Core certificati per accedere CSRs al provisioning del parco veicoli. AWS IoT Core richiama il fornitore del certificato, che accetta un CSR come input e restituisce un certificato client. Per creare un fornitore di certificati, utilizza l'operazione `CreateCertificateProvider` API o il comando `create-certificate-provider` CLI.

> **Note**
>
> Dopo aver creato un fornitore di certificati, il comportamento dell'`CreateCertificateFromCsr`API for fleet provisioning cambierà in modo che tutte le chiamate a `CreateCertificateFromCsr` richiameranno il fornitore di certificati per creare i certificati. La modifica di questo comportamento dopo la creazione di un fornitore di certificati può richiedere alcuni minuti.

```
aws iot create-certificate-provider \
               --certificateProviderName my-certificate-provider \
               --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-function-1 \
               --accountDefaultForOperations CreateCertificateFromCsr
```

Di seguito viene illustrato un esempio di output per questo comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-certificate-provider"
}
```

Per ulteriori informazioni, vedere `CreateCertificateProvider` dall'AWS IoTAPI Reference.

Console di gestione AWS

Per scegliere di utilizzare la firma dei certificati autogestita Console di gestione AWS, procedi nel seguente modo:

1. Accedere alla console AWS IoT.
2. Nella barra di navigazione a sinistra, in Sicurezza, scegli Firma del certificato.
3. Nella pagina di firma del certificato, in Dettagli di firma del certificato, scegli Modifica metodo di firma del certificato.
4. Nella pagina Modifica metodo di firma del certificato, in Metodo di firma del certificato, scegli Gestione automatica.
5. Nella sezione Impostazioni autogestite, inserisci un nome per il fornitore di certificati, quindi crea o scegli una funzione Lambda.
6. Scegli Aggiorna la firma del certificato.

AWS CLI comandi per il fornitore di certificati

Crea un fornitore di certificati

Per creare un fornitore di certificati, utilizza l'operazione `CreateCertificateProvider` API o il comando `create-certificate-provider` CLI.

> **Note**
>
> Dopo aver creato un fornitore di certificati, il comportamento dell'`CreateCertificateFromCsr`API for fleet provisioning cambierà in modo che tutte le chiamate a `CreateCertificateFromCsr` richiameranno il fornitore di certificati per creare i certificati. La modifica di questo comportamento dopo la creazione di un fornitore di certificati può richiedere alcuni minuti.

```
aws iot create-certificate-provider \
                --certificateProviderName my-certificate-provider \
                --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-
function-1 \
                --accountDefaultForOperations CreateCertificateFromCsr
```

Di seguito viene illustrato un esempio di output per questo comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-certificate-provider"
}
```

Per ulteriori informazioni, vedere `CreateCertificateProvider` dall'AWS IoTAPI Reference.

Aggiorna il fornitore di certificati

Per aggiornare un fornitore di certificati, utilizza l'operazione `UpdateCertificateProvider` API o il comando `update-certificate-provider` CLI.

```
aws iot update-certificate-provider \
               --certificateProviderName my-certificate-provider \
               --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-function-2 \
               --accountDefaultForOperations CreateCertificateFromCsr
```

Di seguito viene illustrato un esempio di output per questo comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-certificate-provider"
}
```

Per ulteriori informazioni, vedere `UpdateCertificateProvider` dall'AWS IoTAPI Reference.

Descrivi il fornitore di

Per descrivere un fornitore di certificati, utilizza l'operazione `DescribeCertificateProvider` API o il comando `describe-certificate-provider` CLI.

```
aws iot describe-certificate-provider --certificateProviderName my-certificate-provider
```

Di seguito viene illustrato un esempio di output per questo comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "lambdaFunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
 "accountDefaultForOperations": [
  "CreateCertificateFromCsr"
 ],
 "creationDate": "2022-11-03T00:15",
 "lastModifiedDate": "2022-11-18T00:15"
}
```

Per ulteriori informazioni, vedere `DescribeCertificateProvider` dall'AWS IoTAPI Reference.

Eliminare il fornitore di certificati

Per eliminare un fornitore di certificati, utilizza l'operazione `DeleteCertificateProvider` API o il comando `delete-certificate-provider` CLI. Se elimini la risorsa relativa al fornitore di certificati, il comportamento di `CreateCertificateFromCsr` riprenderà e AWS IoT verranno creati certificati firmati AWS IoT da una CSR.

```
aws iot delete-certificate-provider --certificateProviderName my-certificate-provider
```

Il comando non produce output.

Per ulteriori informazioni, consulta l'AWS IoTAPI *`DeleteCertificateProvider`* Reference.

Elenca i fornitori di certificati

Per elencare i fornitori di certificati all'interno del tuo Account AWS, utilizza l'operazione `ListCertificateProviders` API o il comando `list-certificate-providers` CLI.

```
aws iot list-certificate-providers
```

Di seguito viene illustrato un esempio di output per questo comando:

```
{
 "certificateProviders": [
  {
   "certificateProviderName": "my-certificate-provider",
   "certificateProviderArn": "arn:aws:iot:us-
east-1:123456789012:certificateprovider:my-certificate-provider"
```

```
  }
 ]
}
```

Per ulteriori informazioni, vedere [ListCertificateProvider](#)dall'AWS IoTAPI Reference.

Creazione di policy e ruoli IAM per un utente che installa un dispositivo

> **Note**
>
> Queste procedure possono essere utilizzate solo se indicate dalla AWS IoT console. Per accedere a questa pagina dalla console, apri crea un nuovo modello di provisioning.

Perché questa operazione non può essere eseguita nella AWS IoT console?

Per un'esperienza più sicura, le operazioni IAM vengono eseguite nella console IAM. Le procedure di questa sezione illustrano i passaggi per creare i ruoli e le policy IAM necessarie per l'utilizzo del modello di provisioning.

Creazione di una policy IAM per l'utente che installerà un dispositivo

Questa procedura descrive in che modo creare una policy IAM che autorizza un utente a installare un dispositivo utilizzando un modello di provisioning.

Durante l'esecuzione di questa procedura, passerai dalla console IAM alla AWS IoT console. Si consiglia di tenere entrambe le console aperte contemporaneamente mentre si completa questa procedura.

Per creare una policy IAM per l'utente che installerà un dispositivo

1. Aprire la pagina Policies hub in the IAM console (Hub delle policy nella console IAM).
2. Scegli Crea policy.
3. Nella pagina Crea policy, scegli la scheda JSON.
4. Passa alla pagina della AWS IoT console in cui hai scelto Configura la politica e il ruolo dell'utente.

5. Nella Sample provisioning policy (Policy di provisioning di esempio), scegliere Copy (Copia).
6. Tornare alla console IAM.
7. Nell'editor JSON, incolla la policy che hai copiato dalla AWS IoT console. Questa politica è specifica per il modello che stai creando nella AWS IoT console.
8. Scegliere Next: Tags (Avanti: Tag) per continuare.
9. Sulla pagina Add tags (Optional) (Aggiungi tag (Facoltativo)), scegliere Add tags (Aggiungi tag) per ogni tag che si desidera aggiungere a questa policy. È possibile ignorare questo passaggio se non si dispone di tag da aggiungere.
10. Per continuare, selezionare Next: Review (Avanti: esamina).
11. Nella pagina Review Policy (Rivedi policy), effettua le operazioni seguenti:

    a. Alla voce Name* (Nome*), inserire un nome per la policy che aiuterà a ricordare lo scopo della policy.

       Prendere nota del nome assegnato a questa policy perché verrà utilizzato nella procedura successiva.

    b. È possibile scegliere di inserire una descrizione opzionale per la policy che si sta creando.

    c. Consultare il resto di questa policy e i relativi tag.
12. Scegliere Create Policy (Crea policy) per completare la creazione della policy.

Dopo aver creato la nuova policy, proseguire su the section called “Creazione di un ruolo IAM per l'utente che installerà un dispositivo” per creare la voce del ruolo dell'utente a cui associare questa policy.

Creazione di un ruolo IAM per l'utente che installerà un dispositivo

Questi passaggi descrivono il modo in cui creare un ruolo IAM che autentichi l'utente che installerà un dispositivo utilizzando un modello di provisioning.

Per creare una policy IAM per l'utente che installerà un dispositivo

1. Aprire la pagina Role hub in the IAM console (Hub dei ruoli nella console IAM).
2. Scegli Crea ruolo.
3. Alla voce Select trusted entity (Seleziona entità attendibile), scegliere il tipo di entità attendibile a cui si desidera concedere l'accesso al modello che si stai creando.

4. Scegliere o immettere l'identificativo dell'entità attendibile a cui si desidera concedere l'accesso, quindi scegliere Next (Successivo).
5. Nella pagina Add permissions (Aggiungi autorizzazioni), alla voce Permission policies (Policy di autorizzazione), immettere nella casella di ricerca il nome della policy creata nella procedura precedente.
6. Per l'elenco delle policy, scegliere la policy creata nella procedura precedente, quindi scegliere Next (Successivo).
7. Nella sezione Name, review, and create (Assegna un nome, rivedi e crea), eseguire le operazioni seguenti:

    a. In Role name (Nome ruolo), immettere un nome del ruolo che consenta di identificarne lo scopo.
    b. Per Description (Descrizione), è possibile scegliere di immettere una descrizione opzionale del ruolo. Questo passaggio non è obbligatorio per proseguire.
    c. Rivedere i valori di Passaggio 1 e Passaggio 2.
    d. Alla voce Add tags (Optional) (Aggiungi tag (Facoltativo)), è possibile scegliere di aggiungere tag a questo ruolo. Questo passaggio non è obbligatorio per proseguire.
    e. Verificare che le informazioni in questa pagina siano complete e corrette, quindi scegliere Create role (Crea ruolo).

Dopo aver creato il nuovo ruolo, torna alla AWS IoT console per continuare a creare il modello.

Aggiornamento di una policy esistente per autorizzare un nuovo modello

Nei passaggi seguenti viene descritto come aggiungere un nuovo modello a una policy IAM che autorizza un utente a installare un dispositivo utilizzando un modello di provisioning.

Per aggiungere un nuovo modello a una policy IAM esistente

1. Aprire la pagina Policies hub in the IAM console (Hub delle policy nella console IAM).
2. Nella casella di ricerca inserire il nome della policy da aggiornare.
3. Nell'elenco sotto la casella di ricerca, individuare la policy che si desidera aggiornare e scegliere il suo nome.
4. Alla voce Policy summary (Riepilogo policy), scegliere la scheda JSON se tale pannello non è già visibile.

5. Se si desidera modificare la policy, scegliere Edit policy (Modifica policy).
6. Nell'editor, scegliere la scheda JSON se tale pannello non è già visibile.
7. Nel documento sulla policy, individuare l'istruzione della policy che contiene l'operazione `iot:CreateProvisioningClaim`.

   Se il documento della policy non contiene un'istruzione della policy con l'operazione `iot:CreateProvisioningClaim`, copiare il seguente frammento di istruzione e incollarlo come voce aggiuntiva nella sezione `Statement` nel documento della policy.

   > **Note**
   >
   > Questo frammento deve essere inserito prima del carattere `]` di chiusura della sezione `Statement`. Potrebbe essere necessario aggiungere una virgola prima o dopo questo frammento per correggere eventuali errori di sintassi.

   ```
   {
       "Effect": "Allow",
       "Action": [
           "iot:CreateProvisioningClaim"
       ],
       "Resource": [
           "--PUT YOUR NEW TEMPLATE ARN HERE--"
       ]
   }
   ```
8. Passa alla pagina della AWS IoT console in cui hai scelto Modifica le autorizzazioni del ruolo utente.
9. Individuare il Resource ARN (ARN della risorsa) del modello e scegliere Copy (Copia).
10. Tornare alla console IAM.
11. Incolla l'Amazon Resource Name (ARN) copiato nella parte superiore dell'elenco di ARNs modelli nell'array in `Statement` modo che sia la prima voce.

    Se questo è l'unico ARN nella sezione, rimuovere la virgola alla fine del valore appena incollato.
12. Esaminare l'istruzione della policy aggiornata e correggere eventuali errori indicati dall'editor.
13. Per salvare il documento di policy aggiornato, scegliere Review policy (Revisione policy).
14. Esaminare la policy e quindi scegliere Save changes (Salva modifiche).

15. Torna alla console. AWS IoT

API MQTT di provisioning del dispositivo

Il servizio Fleet Provisioning supporta le seguenti operazioni dell'API MQTT:

- the section called “CreateCertificateFromCsr”
- the section called “CreateKeysAndCertificate”
- the section called “RegisterThing”

Questa API supporta buffer di risposta in formato Concise Binary Object Representation (CBOR) e JavaScript Object Notation (JSON), a seconda dell'argomento. *payload-format* Per maggiore chiarezza, gli esempi di risposta e richiesta in questa sezione sono mostrati in formato JSON.

payload-format	Tipo di dati del formato della risposta
cbor	Concise Binary Object Representation (CBOR)
json	JavaScript Notazione di oggetti (JSON)

> **Important**
>
> Prima di pubblicare un argomento del messaggio di richiesta, sottoscrivere gli argomenti di risposta per ricevere la risposta. I messaggi utilizzati da questa API utilizzano il protocollo di pubblicazione/sottoscrizione di MQTT per fornire un'interazione di richiesta e risposta.
> Se non ti iscrivi agli argomenti di risposta prima di pubblicare una richiesta, potresti non ricevere i risultati di quella richiesta.
> IoT Core Fleet Provisioning restituisce i risultati dell'API MQTT di provisioning del dispositivo tramite la stessa connessione MQTT utilizzata per pubblicare la richiesta API.

CreateCertificateFromCsr

Crea un certificato da una richiesta di firma del certificato (CSR). AWS IoT fornisce certificati client firmati dall'autorità di certificazione Amazon Root (CA). Il nuovo certificato ha uno stato `PENDING_ACTIVATION`. Quando si chiama `RegisterThing` per eseguire il provisioning di un

oggetto con questo certificato, lo stato del certificato cambia in `ACTIVE` o `INACTIVE` come descritto nel modello.

Per ulteriori informazioni sulla creazione di un certificato client utilizzando il certificato dell'autorità di certificazione e una richiesta di firma del certificato, consulta Creare un certificato client utilizzando il certificato CA.

> **Note**
>
> Per sicurezza, il valore `certificateOwnershipToken` restituito da `CreateCertificateFromCsr` scade dopo un'ora. `RegisterThing` deve essere chiamato prima della scadenza di `certificateOwnershipToken`. Se il certificato creato da `CreateCertificateFromCsr` non è stato attivato e allegato a una politica o a un oggetto entro la scadenza del token, il certificato viene eliminato. Se il token scade, il dispositivo può chiamare `CreateCertificateFromCsr` per generare un nuovo certificato.

CreateCertificateFromCsr richiesta

Pubblicare un messaggio con l'argomento `$aws/certificates/create-from-csr/`*`payload-format`*.

`payload-format`

Formato di payload del messaggio come `cbor` o `json`.

CreateCertificateFromCsr richiedere un payload

```
{
    "certificateSigningRequest": "string"
}
```

`certificateSigningRequest`

La CSR, in formato PEM.

CreateCertificateFromCsr risposta

Sottoscrizione ad `$aws/certificates/create-from-csr/`*`payload-format`*`/accepted`.

`payload-format`

Formato di payload del messaggio come `cbor` o `json`.

CreateCertificateFromCsr carico utile di risposta

```
{
    "certificateOwnershipToken": "string",
    "certificateId": "string",
    "certificatePem": "string"
}
```

`certificateOwnershipToken`

Il token per dimostrare la proprietà del certificato durante il provisioning.

`certificateId`

ID del certificato. Le operazioni di gestione dei certificati accettano solo un parametro certificateId.

`certificatePem`

Dati del certificato, in formato PEM.

CreateCertificateFromCsr errore

Per ricevere risposte di errore, iscriviti a `$aws/certificates/create-from-csr/`*`payload-format`*`/rejected`.

`payload-format`

Formato di payload del messaggio come `cbor` o `json`.

CreateCertificateFromCsr payload di errore

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

Il codice di stato.

`errorCode`

Il codice di errore.

`errorMessage`

Messaggio di errore.

CreateKeysAndCertificate

Crea nuove chiavi e un certificato. AWS IoT fornisce certificati client firmati dall'autorità di certificazione Amazon Root (CA). Il nuovo certificato ha uno stato `PENDING_ACTIVATION`. Quando si chiama `RegisterThing` per eseguire il provisioning di un oggetto con questo certificato, lo stato del certificato cambia in `ACTIVE` o `INACTIVE` come descritto nel modello.

> **Note**
>
> Per sicurezza, il valore `certificateOwnershipToken` restituito da `CreateKeysAndCertificate` scade dopo un'ora. `RegisterThing` deve essere chiamato prima della scadenza di `certificateOwnershipToken`. Se il certificato creato da `CreateKeysAndCertificate` non è stato attivato e allegato a una politica o a un oggetto entro la scadenza del token, il certificato viene eliminato. Se il token scade, il dispositivo può chiamare `CreateKeysAndCertificate` per generare un nuovo certificato.

CreateKeysAndCertificate richiesta

Pubblicare un messaggio su `$aws/certificates/create/`*`payload-format`* con un payload di messaggio vuoto.

`payload-format`

Formato di payload del messaggio come `cbor` o `json`.

CreateKeysAndCertificate risposta

Sottoscrizione ad `$aws/certificates/create/`*`payload-format`*`/accepted`.

`payload-format`

Formato di payload del messaggio come `cbor` o `json`.

CreateKeysAndCertificate risposta

```
{
    "certificateId": "string",
    "certificatePem": "string",
    "privateKey": "string",
    "certificateOwnershipToken": "string"
}
```

`certificateId`

ID certificato.

`certificatePem`

Dati del certificato, in formato PEM.

`privateKey`

Chiave privata.

`certificateOwnershipToken`

Il token per dimostrare la proprietà del certificato durante il provisioning.

CreateKeysAndCertificate errore

Per ricevere risposte di errore, iscriviti a `$aws/certificates/create/`*`payload-format`*`/rejected`.

`payload-format`

Formato di payload del messaggio come `cbor` o `json`.

CreateKeysAndCertificate payload di errore

```
{
```

```
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

Il codice di stato.

`errorCode`

Il codice di errore.

`errorMessage`

Messaggio di errore.

RegisterThing

Esegue il provisioning di un oggetto utilizzando un modello predefinito.

RegisterThing richiesta

Pubblicare un messaggio su `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*.

`payload-format`

Formato di payload del messaggio come `cbor` o `json`.

`templateName`

Il nome del modello di provisioning.

RegisterThing richiedere un payload

```
{
    "certificateOwnershipToken": "string",
    "parameters": {
        "string": "string",
        ...
    }
```

```
}
```

`certificateOwnershipToken`

Il token per dimostrare la proprietà del certificato. AWS IoT genera il token quando si crea un certificato su MQTT.

`parameters`

Opzionale. Coppie chiave-valore dal dispositivo utilizzato dagli hook di pre-provisioning per valutare la richiesta di registrazione.

RegisterThing risposta

Sottoscrizione ad `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*`/accepted`.

`payload-format`

Formato di payload del messaggio come `cbor` o `json`.

`templateName`

Il nome del modello di provisioning.

RegisterThing carico utile di risposta

```
{
    "deviceConfiguration": {
        "string": "string",
        ...
    },
    "thingName": "string"
}
```

`deviceConfiguration`

La configurazione del dispositivo definita nel modello.

`thingName`

Il nome dell'oggetto IoT creato durante il provisioning.

RegisterThing risposta all'errore

Per ricevere risposte di errore, iscriviti a `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*`/rejected`.

`payload-format`

Formato di payload del messaggio come `cbor` o `json`.

`templateName`

Il nome del modello di provisioning.

RegisterThing payload di risposta agli errori

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

Il codice di stato.

`errorCode`

Il codice di errore.

`errorMessage`

Messaggio di errore.

Indicizzazione del parco istanze

Puoi utilizzare l'indicizzazione del parco istanze per indicizzare, cercare e aggregare i dati dei dispositivi dalle seguenti origini: registro di AWS IoT, AWS IoT Device Shadow, connettività di AWS IoT, Software Package Catalog gestione dispositivi AWS IoT e violazioni di AWS IoT Device Defender. È possibile eseguire query su un gruppo di dispositivi e aggregare statistiche sui record dei dispositivi basati su diverse combinazioni di attributi del dispositivo, tra cui stato, connettività e violazioni del dispositivo. Con l'indicizzazione del parco istanze puoi organizzare, indagare e risolvere i problemi del tuo parco istanze di dispositivi.

L'indicizzazione del parco istanze offre le caratteristiche elencate di seguito.

Gestione degli aggiornamenti degli indici

È possibile impostare un indice del parco istanze per indicizzare gli aggiornamenti dei gruppi di oggetti, dei registri di oggetti, le shadow dei dispositivi, della connettività dei dispositivi e delle violazioni del dispositivo. Quando attivi l'indicizzazione del parco istanze, AWS IoT crea un indice per gli oggetti o i gruppi di oggetti. `AWS_Things` è l'indice creato per tutti i tuoi oggetti. `AWS_ThingGroups` è l'indice che contiene tutti i tuoi gruppi di oggetti. Quando l'indicizzazione del parco istanze è attiva, puoi eseguire query sull'indice. Ad esempio, puoi trovare tutti i dispositivi portatili con una durata della batteria superiore al 70%. AWS IoT aggiorna continuamente l'indice con i dati più recenti. Per ulteriori informazioni, consulta Servizio Managing fleet indexing (Gestione dell'indicizzazione del parco istanze).

Interrogazione dello stato della connettività per un dispositivo specifico

Questa API fornisce un accesso a bassa latenza e ad alta velocità alle informazioni di connettività specifiche del dispositivo più recenti. Per ulteriori informazioni, consulta Stato della connettività del dispositivo.

Ricerca tra origini dei dati

Puoi creare una stringa di query basata su un linguaggio di query e utilizzarla per cercare tra le origini dati. È inoltre necessario configurare le fonti di dati nell'impostazione di indicizzazione del parco veicoli in modo che la configurazione di indicizzazione contenga le fonti di dati da cui effettuare la

ricerca. La stringa di query descrive gli oggetti che desideri trovare. Puoi creare query utilizzando campi AWS gestiti, campi personalizzati e qualsiasi attributo delle tue fonti di dati indicizzate. Per ulteriori informazioni sulle origini dati che supportano l'indicizzazione del parco istanze, consulta Managing thing indexing (Gestione dell'indicizzazione degli oggetti).

Interrogazione di dati aggregati

Puoi cercare i tuoi dispositivi per trovare dati aggregati e restituire statistiche, percentile, cardinalità o un elenco di oggetti con query di ricerca relative a campi specifici. Puoi eseguire aggregazioni su campi AWS gestiti o su qualsiasi attributo configurato come campi personalizzati all'interno delle impostazioni di indicizzazione del parco veicoli. Per ulteriori informazioni sull'esecuzione di query, consulta Interrogazione di dati aggregati.

Monitoraggio dei dati di aggregazione e creazione di allarmi utilizzando parametri del parco istanze

Puoi utilizzare le metriche della flotta per inviare CloudWatch automaticamente dati aggregati, analizzare le tendenze e creare allarmi per monitorare lo stato aggregato della tua flotta in base a soglie predefinite. Per ulteriori informazioni sui parametri del parco istanze, consulta Fleet metrics (Parametri del parco istanze).

Gestione dell'indicizzazione del parco istanze

L'indicizzazione del parco istanze consente di gestire automaticamente due tipi di indici: l'indicizzazione di oggetti e l'indicizzazione di gruppi di oggetti.

Indicizzazione degli oggetti

L'indice creato per tutti i tuoi oggetti è chiamato `AWS_Things`. L'indicizzazione degli oggetti supporta le seguenti origini dati: dati di registro di AWS IoT, dati di AWS IoT Device Shadow , dati di connettività di AWS IoT e dati di violazioni di AWS IoT Device Defender. Aggiungendo queste origini dei dati alla configurazione di indicizzazione del parco istanze, puoi cercare oggetti, eseguire query su dati aggregati e creare gruppi di oggetti dinamici e parametri del parco istanze in base alle tue query di ricerca.

Registro:AWS IoT fornisce un registro che ti aiuta a gestire le cose. È possibile aggiungere i dati di registro alla configurazione di indicizzazione del parco istanze per cercare i dispositivi in base ai

nomi degli oggetti, alle descrizioni e ad altri attributi del registro. Per ulteriori informazioni sul registro, consulta Come gestire gli oggetti con il registro.

Shadow: il servizio Device Shadow AWS IoT fornisce copie shadow che facilitano l'archiviazione dei dati sullo stato del dispositivo. L'indicizzazione degli oggetti supporta sia le classiche copie shadow senza nome che le copie shadow con nome. Per indicizzare le copie shadow denominate, attiva le impostazioni delle tue copie shadow denominate e specificane i nomi nella configurazione di indicizzazione oggetti. Per impostazione predefinita, è possibile aggiungere fino a 10 nomi shadow per persona Account AWS. Per informazioni su come aumentare il limite del numero di nomi delle copie shadow, consulta AWS IoT Device Management Quotas nella Guida di riferimento di AWS .

Per aggiungere copie shadow specifiche per l'indicizzazione:

- Se utilizzi la console AWS IoT, attiva l'indicizzazione di oggetti, scegli Add named shadows (Aggiungi copie shadow denominate), quindi aggiungi i nomi delle copie shadow tramite Named shadow selection (Selezione della copia shadow denominata).
- Se usate il AWS Command Line Interface (AWS CLI), impostate su `namedShadowIndexingMode` be e `ON` specificate i nomi delle ombre in `IndexingFilter`. Per gli esempi di comandi della CLI, consulta Manage thing indexing (Gestione dell'indicizzazione di oggetti).

> **Important**
>
> Il 20 luglio 2022 è disponibile la versione General Availability (GA) dell'integrazione dell'indicizzazione della flotta di AWS IoT Device Management con shadow AWS IoT Core denominati e AWS IoT Device Defender rilevamento delle violazioni. Con questo rilascio pubblico, potrai indicizzare copie shadow denominate specificando i nomi delle copie shadow. Se hai aggiunto le tue copie shadow denominate per l'indicizzazione durante il periodo di anteprima pubblica di questa funzione, ovvero dal 30 novembre 2021 al 19 luglio 2022, ti invitiamo a riconfigurare le impostazioni di indicizzazione del parco istanze e scegliere nomi di copie shadow specifici per ridurre i costi di indicizzazione e ottimizzare le prestazioni.

Per ulteriori informazioni sulle shadow, consulta Servizio Device Shadow AWS IoT.

Connettività: i dati di connettività del dispositivo consentono di identificare lo stato di connessione dei dispositivi. Questi dati di connettività sono guidati da eventi del ciclo di vita. Quando un client si connette o si disconnette, AWS IoT pubblica eventi del ciclo di vita con messaggi su argomenti

MQTT. Un messaggio di connessione o disconnessione può essere un elenco di elementi JSON che forniscono dettagli sullo stato della connessione. Per ulteriori informazioni sulla connettività del dispositivo, consulta Eventi del ciclo di vita.

Violazioni di Device Defender: i dati AWS IoT Device Defender sulle violazioni aiutano a identificare i comportamenti anomali dei dispositivi rispetto ai comportamenti normali definiti in un profilo di sicurezza. Un profilo di sicurezza contiene un set di comportamenti previsti del dispositivo. Ogni comportamento utilizza un parametro che specifica il comportamento normale dei dispositivi. Per ulteriori informazioni sulle violazioni di Device Defender, consulta detect.AWS IoT Device Defender

Per ulteriori informazioni, consulta Gestione dell'indicizzazione degli oggetti.

Indicizzazione di gruppi di oggetti

`AWS_ThingGroups` è l'indice che contiene tutti i tuoi gruppi di oggetti. Questo indice consente di cercare gruppi in base al nome del gruppo, alla descrizione, agli attributi e a tutti i nomi del gruppo padre.

Per ulteriori informazioni, consulta Gestione dell'indicizzazione di gruppi di oggetti.

Campi gestiti

I campi gestiti contengono dati associati a oggetti, gruppi di oggetti, ombre dei dispositivi, connettività dei dispositivi e violazioni di Device Defender. AWS IoT definisce il tipo di dati nei campi gestiti. Si specificano i valori di ogni campo gestito quando si crea un AWS IoT oggetto. Ad esempio i nomi degli oggetti, i gruppi di oggetti e le descrizioni degli oggetti sono tutti campi gestiti. L'indicizzazione del parco istanze indica i campi gestiti in base alla modalità di indicizzazione che specifici. I campi gestiti non possono essere modificati o visualizzati in `customFields`. Per ulteriori informazioni, consulta Custom fields (Personalizza campi).

Di seguito sono elencati i campi gestiti per l'indicizzazione delle cose:

- Campi gestiti per il Registro di sistema

```
"managedFields" : [
  {name:thingId, type:String},
  {name:thingName, type:String},
  {name:registry.version, type:Number},
  {name:registry.thingTypeName, type:String},
  {name:registry.thingGroupNames, type:String},
]
```

- Campi gestiti per shadow classiche senza nome

```
"managedFields" : [
  {name:shadow.version, type:Number},
  {name:shadow.hasDelta, type:Boolean}
]
```

- Campi gestiti per shadow con nome

```
"managedFields" : [
  {name:shadow.name.shadowName.version, type:Number},
  {name:shadow.name.shadowName.hasDelta, type:Boolean}
]
```

- Campi gestiti per la connettività degli oggetti

```
"managedFields" : [
  {name:connectivity.timestamp, type:Number},
  {name:connectivity.version, type:Number},
  {name:connectivity.connected, type:Boolean},
  {name:connectivity.disconnectReason, type:String}
]
```

- Campi gestiti per Device Defender

```
"managedFields" : [
  {name:deviceDefender.violationCount, type:Number},
  {name:deviceDefender.securityprofile.behaviorname.metricName, type:String},
  {name:deviceDefender.securityprofile.behaviorname.lastViolationTime, type:Number},
  {name:deviceDefender.securityprofile.behaviorname.lastViolationValue, type:String},
  {name:deviceDefender.securityprofile.behaviorname.inViolation, type:Boolean}
]
```

- Campi gestiti per i gruppi di oggetti

```
"managedFields" : [
  {name:description, type:String},
  {name:parentGroupNames, type:String},
  {name:thingGroupId, type:String},
  {name:thingGroupName, type:String},
  {name:version, type:Number},
]
```

Nella tabella seguente sono elencati i campi gestiti che non sono ricercabili.

Origine dati	Il campo gestito non è ricercabile
Registro	`registry.version`
Copie shadow senza nome	`shadow.version`
Copie shadow con nome	`shadow.name.*.version`
Device Defender	`deviceDefender.version`
Gruppi di oggetti	`version`

Campi personalizzati

È possibile aggregare gli attributi degli oggetti, i dati di Device Shadow e i dati delle violazioni di Device Defender creando campi personalizzati per indicizzarli. L'attributo `customFields` è un elenco di coppie di nomi di campi e tipi di dati. È possibile eseguire query di aggregazione in base al tipo di dati. La modalità di indicizzazione che scegli influenza i campi che è possibile specificare in `customFields`. Ad esempio, se si specifica la modalità di indicizzazione `REGISTRY`, non è possibile specificare un campo personalizzato da una copia shadow dell'oggetto. È possibile utilizzare il comando update-indexing-configurationCLI per creare o aggiornare i campi personalizzati (vedere un comando di esempio in Aggiornamento degli esempi di configurazione dell'indicizzazione).

- Nomi di campo personalizzati

I nomi dei campi personalizzati per gli attributi di oggetti e gruppi di oggetti iniziano con `attributes.`, seguito dal nome dell'attributo. Se l'indicizzazione delle copie shadow senza nome è attiva, gli oggetti possono avere nomi di campo personalizzati che iniziano con `shadow.desired` o `shadow.reported`, seguito dal nome del valore dei dati delle copie shadow senza nome. Se l'indicizzazione delle copie shadow con nome è attiva, gli oggetti possono avere nomi di campo personalizzati che iniziano con `shadow.name.*.desired.` o `shadow.name.*.reported.`, seguito dal valore dei dati delle copie shadow con nome. Se l'indicizzazione delle violazioni di Device Defender è attiva, gli oggetti possono avere nomi di campo personalizzati che iniziano con `deviceDefender.`, seguito dal valore dei dati delle violazioni di Device Defender.

Il nome dell'attributo o del valore dei dati che segue il prefisso può contenere solo caratteri alfanumerici, - (trattino) e _ (carattere di sottolineatura). Non può avere spazi.

Se è presente un'incongruenza di tipo tra un campo personalizzato nella configurazione e il valore indicizzato, il servizio di indicizzazione del parco istanze ignora il valore incoerente per le query di aggregazione. CloudWatch I log sono utili per la risoluzione dei problemi relativi alle query di aggregazione. Per ulteriori informazioni, consulta Risoluzione dei problemi delle query di aggregazione per il servizio di indicizzazione del parco istanze.

- Tipi di campo personalizzati

I tipi di campo personalizzati hanno i seguenti valori supportati: `Number`, `String`, e `Boolean`.

Gestione dell'indicizzazione degli oggetti

L'indice creato per tutti i tuoi oggetti è `AWS_Things`. È possibile controllare cosa indicizzare dalle seguenti origini dati: dati di registro di AWS IoT, dati di AWS IoT Device Shadow , dati di connettività di AWS IoT, e dati di violazioni di AWS IoT Device Defender.

In questo argomento:

- Abilitazione dell'indicizzazione degli oggetti
- Descrizione di un indice dell'oggetto
- Esecuzione di query su un indice di oggetti
- Restrizioni e limitazioni
- Autorizzazione

Abilitazione dell'indicizzazione degli oggetti

Si utilizza il comando update-indexing-configurationCLI o l'operazione UpdateIndexingConfigurationAPI per creare l'`AWS_Things`indice e controllarne la configurazione. Il parametro `--thing-indexing-configuration` (`thingIndexingConfiguration`) consente di controllare il tipo di dati indicizzati (ad esempio, registro, copia shadow, dati di connettività del dispositivo e dati di violazioni di Device Defender).

Il parametro `--thing-indexing-configuration` accetta una stringa con la seguente struttura:

```
{
  "thingIndexingMode": "OFF"|"REGISTRY"|"REGISTRY_AND_SHADOW",
```

```
  "thingConnectivityIndexingMode": "OFF"|"STATUS",
  "deviceDefenderIndexingMode": "OFF"|"VIOLATIONS",
  "namedShadowIndexingMode": "OFF"|"ON",
  "managedFields": [
    {
      "name": "string",
      "type": "Number"|"String"|"Boolean"
    },
    ...
  ],
  "customFields": [
    {
      "name": "string",
      "type": "Number"|"String"|"Boolean"
    },
    ...
  ],
  "filter": {
     "namedShadowNames": [ "string" ],
     "geoLocations": [
        {
            "name": "String",
            "order": "LonLat|LatLon"
        }
    ]
  }
}
```

Modalità di indicizzazione delle cose

È possibile specificare diverse modalità di indicizzazione degli oggetti nella configurazione di indicizzazione, a seconda delle fonti di dati da cui si desidera indicizzare e cercare i dispositivi:

- `thingIndexingMode`: controlla se il registro o l'ombra sono indicizzati. Quando `thingIndexingMode` è impostato su be`OFF`, l'indicizzazione degli oggetti è disabilitata.

- `thingConnectivityIndexingMode`: specifica se i dati di connettività dell'oggetto sono indicizzati.

- `deviceDefenderIndexingMode`: specifica se i dati delle violazioni di Device Defender sono indicizzati.

- `namedShadowIndexingMode`: specifica se i dati shadow denominati sono indicizzati. Per selezionare le copie shadow denominate da aggiungere alla configurazione di indicizzazione del parco istanze, imposta `namedShadowIndexingMode` in modo che sia `ON` e specifica i nomi delle copie shadow denominate in `filter`.

La tabella seguente mostra i valori validi per ogni modalità di indicizzazione e l'origine dati indicizzata per ogni valore.

Attributo	Valori validi	Registro	Shadow	Connettività	Violazioni DD	Copia shadow con nome
`thingIndexingMode`	OFF					
	REGISTRY	✓				
	REGISTRY_AND_SHADOW	✓	✓			
`thingConnectivityIndexingMode`	Non specificato.					
	OFF					
	STATUS			✓		
`deviceDefenderIndexingMode`	Non specificato.					
	OFF					
	VIOLAZIONI				✓	
`namedShadowIndexingMode`	Non specificato.					
	OFF					

Attributo	Valori validi	Registro	Shadow	Connettività	Violazioni DD	Copia shadow con nome
	ATTIVATO					✓

Campi gestiti e campi personalizzati

Campi gestiti

I campi gestiti contengono dati associati a oggetti, gruppi di oggetti, ombre dei dispositivi, connettività dei dispositivi e violazioni di Device Defender. AWS IoT definisce il tipo di dati nei campi gestiti. Specifica i valori di ogni campo gestito quando crei un oggetto AWS IoT . Ad esempio i nomi degli oggetti, i gruppi di oggetti e le descrizioni degli oggetti sono tutti campi gestiti. L'indicizzazione del parco istanze indica i campi gestiti in base alla modalità di indicizzazione che specifici. I campi gestiti non possono essere modificati o visualizzati in `customFields`.

Campi personalizzati

È possibile aggregare gli attributi, i dati di Device Shadow e i dati delle violazioni di Device Defender creando campi personalizzati per indicizzarli. L'attributo `customFields` è un elenco di coppie di nomi di campi e tipi di dati. È possibile eseguire query di aggregazione in base al tipo di dati. La modalità di indicizzazione che scegli influenza i campi che è possibile specificare in `customFields`. Ad esempio, se si specifica la modalità di indicizzazione `REGISTRY`, non è possibile specificare un campo personalizzato da una copia shadow dell'oggetto. È possibile utilizzare il comando update-indexing-configurationCLI per creare o aggiornare i campi personalizzati (vedere un comando di esempio in Aggiornamento degli esempi di configurazione dell'indicizzazione). Per ulteriori informazioni, consulta Custom fields (Personalizza campi).

Filtro di indicizzazione

Il filtro di indicizzazione fornisce selezioni aggiuntive per ombre denominate e dati di geolocalizzazione.

`namedShadowNames`

Per aggiungere ombre denominate alla configurazione di indicizzazione della flotta, impostate be e specificate i nomi delle ombre denominate `namedShadowIndexingMode` `ON` nel filtro. `namedShadowNames`

Esempio

```
"filter": {
     "namedShadowNames": [ "namedShadow1", "namedShadow2" ]
}
```

`geoLocations`

Per aggiungere dati di geolocalizzazione alla configurazione di indicizzazione della flotta:

- Se i dati di geolocalizzazione sono archiviati in un'ombra classica (senza nome), impostala su REGISTRY_AND_SHADOW e specifica `thingIndexingMode` i dati di geolocalizzazione nel filtro. `geoLocations`

  Il filtro di esempio seguente specifica un oggetto GeoLocation in un'ombra classica (senza nome):

  ```
  "filter": {
       "geoLocations": [
          {
              "name": "shadow.reported.location",
              "order": "LonLat"
          }
       ]
    }
  ```

- Se i dati di geolocalizzazione sono archiviati in un'ombra denominata, `namedShadowIndexingMode` impostatela su ON, aggiungete il nome dell'ombra nel filtro e specificate i dati di geolocalizzazione nel `namedShadowNames` filtro. `geoLocations`

  Il filtro di esempio riportato di seguito specifica un oggetto GeoLocation in un'ombra denominata (): `nameShadow1`

  ```
  "filter": {
       "namedShadowNames": [ "namedShadow1" ],
       "geoLocations": [
          {
              "name": "shadow.name.namedShadow1.reported.location",
              "order": "LonLat"
          }
       ]
  ```

```
    }
```

Per ulteriori informazioni, consulta API _ IndexingFilterReference AWS IoT.

Aggiornamento degli esempi di configurazione di indicizzazione

Per aggiornare la configurazione di indicizzazione, usa il comando CLI AWS IoT update-indexing-configuration. Gli esempi seguenti mostrano come utilizzare update-indexing-configuration.

Sintassi breve:

```
aws iot update-indexing-configuration --thing-indexing-configuration \
'thingIndexingMode=REGISTRY_AND_SHADOW, deviceDefenderIndexingMode=VIOLATIONS,
namedShadowIndexingMode=ON,filter={namedShadowNames=[thing1shadow]},
 thingConnectivityIndexingMode=STATUS,
customFields=[{name=attributes.version,type=Number},
{name=shadow.name.thing1shadow.desired.DefaultDesired, type=String},
 {name=shadow.desired.power, type=Boolean},
{name=deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number,
 type=Number}]'
```

Sintassi JSON:

```
aws iot update-indexing-configuration --cli-input-json \ '{
          "thingIndexingConfiguration": { "thingIndexingMode": "REGISTRY_AND_SHADOW",
          "thingConnectivityIndexingMode": "STATUS",
          "deviceDefenderIndexingMode": "VIOLATIONS",
          "namedShadowIndexingMode": "ON",
          "filter": { "namedShadowNames": ["thing1shadow"]},
          "customFields": [ { "name": "shadow.desired.power", "type": "Boolean" },
          {"name": "attributes.version", "type": "Number"},
          {"name": "shadow.name.thing1shadow.desired.DefaultDesired", "type":
 "String"},
          {"name":
 "deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number",
 "type": Number} ] } }'
```

Il comando non produce output.

Per controllare lo stato dell'indice degli oggetti, esegui il comando CLI `describe-index`:

```
aws iot describe-index --index-name "AWS_Things"
```

L'output del comando `describe-index` è simile al seguente:

```
{
    "indexName": "AWS_Things",
    "indexStatus": "ACTIVE",
    "schema": "MULTI_INDEXING_MODE"
}
```

> **Note**
>
> L'indicizzazione del parco istanze può richiedere un momento per aggiornare l'indice del parco istanze. Consigliamo di attendere fino al momento in cui `indexStatus` mostra ACTIVE prima di utilizzarlo. È possibile avere valori diversi nel campo dello schema a seconda delle origini dati configurate. Per ulteriori informazioni, consulta Describing a thing index (Descrizione di un indice dell'oggetto).

Per ottenere i dettagli di configurazione dell'indicizzazione delle cose, esegui il comando CLI `get-indexing-configuration`:

```
aws iot get-indexing-configuration
```

L'output del comando `get-indexing-configuration` è simile al seguente:

```
{
    "thingIndexingConfiguration": {
        "thingIndexingMode": "REGISTRY_AND_SHADOW",
        "thingConnectivityIndexingMode": "STATUS",
        "deviceDefenderIndexingMode": "VIOLATIONS",
        "namedShadowIndexingMode": "ON",
        "managedFields": [
            {
                "name": "connectivity.disconnectReason",
                "type": "String"
            },
            {
                "name": "registry.version",
                "type": "Number"
```

```
            },
            {
                "name": "thingName",
                "type": "String"
            },
            {
                "name": "deviceDefender.violationCount",
                "type": "Number"
            },
            {
                "name": "shadow.hasDelta",
                "type": "Boolean"
            },
            {
                "name": "shadow.name.*.version",
                "type": "Number"
            },
            {
                "name": "shadow.version",
                "type": "Number"
            },
            {
                "name": "connectivity.version",
                "type": "Number"
            },
            {
                "name": "connectivity.timestamp",
                "type": "Number"
            },
            {
                "name": "shadow.name.*.hasDelta",
                "type": "Boolean"
            },
            {
                "name": "registry.thingTypeName",
                "type": "String"
            },
            {
                "name": "thingId",
                "type": "String"
            },
            {
                "name": "connectivity.connected",
                "type": "Boolean"
```

```
            },
            {
                "name": "registry.thingGroupNames",
                "type": "String"
            }
        ],
        "customFields": [
            {
                "name": "shadow.name.thing1shadow.desired.DefaultDesired",
                "type": "String"
            },

            {
                "name":
 "deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number",
                "type": "Number"
            },
            {
                "name": "shadow.desired.power",
                "type": "Boolean"
            },
            {
                "name": "attributes.version",
                "type": "Number"
            }
        ],
        "filter": {
              "namedShadowNames": [
                  "thing1shadow"
              ]
          }
      },
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "OFF"
    }
}
```

Per aggiornare i campi personalizzati, esegui il comando `update-indexing-configuration`. Di seguito è riportato l'esempio:

```
aws iot update-indexing-configuration --thing-indexing-configuration

 'thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.version,type=Number},
```

```
{name=attributes.color,type=String},{name=shadow.desired.power,type=Boolean},
{name=shadow.desired.intensity,type=Number}]'
```

Questo comando ha aggiunto `shadow.desired.intensity` alla configurazione di indicizzazione.

> **Note**
>
> L'aggiornamento della configurazione di indicizzazione dei campi personalizzati sovrascrive tutti i campi personalizzati esistenti. Assicurarsi di specificare tutti i campi personalizzati durante la chiamata a update-indexing-configuration.

Dopo che l'indice è stato ricostruito è possibile utilizzare query di aggregazione sui campi appena aggiunti, ricercare dati del registro, dati shadow e dati di stato connettività dell'oggetto.

Quando si modifica la modalità di indicizzazione, assicurarsi che tutti i campi personalizzati siano validi utilizzando la nuova modalità di indicizzazione. Ad esempio, se si inizia con la modalità `REGISTRY_AND_SHADOW` con un campo personalizzato chiamato `shadow.desired.temperature`, è necessario eliminare il campo personalizzato `shadow.desired.temperature` prima di cambiare la modalità di indicizzazione in `REGISTRY`. Se la configurazione di indicizzazione contiene campi personalizzati che non sono indicizzati dalla modalità di indicizzazione, l'aggiornamento non riesce.

Descrizione di un indice dell'oggetto

Il comando seguente mostra come usare il comando dell'interfaccia a riga di comando describe-index per recuperare lo stato corrente dell'indice degli oggetti.

```
aws iot describe-index --index-name "AWS_Things"
```

L'output del comando è simile al seguente:

```
{
    "indexName": "AWS_Things",
    "indexStatus": "BUILDING",
    "schema": "REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS"
}
```

La prima volta che esegui l'indicizzazione della flotta, crea il tuo indice. AWS IoT Non puoi eseguire query sull'indice se `indexStatus` è nello stato `BUILDING`. `schema` per l'indice degli oggetti indica quale tipo di dati (`REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS`) sarà indicizzato.

La modifica della configurazione dell'indice comporta la ricostruzione dell'indice. Durante questo processo, `indexStatus` è `REBUILDING`. È possibile eseguire query sui dati nell'indice degli oggetti, mentre è in fase di ricostruzione. Ad esempio, se modifichi la configurazione dell'indice da `REGISTRY` a `REGISTRY_AND_SHADOW` mentre l'indice è in fase di ricompilazione, puoi eseguire query sui dati del registro, inclusi gli aggiornamenti più recenti. Tuttavia, non puoi eseguire query sui dati shadow fino al completamento della ricompilazione. L'intervallo di tempo necessario per creare o ricostruire l'indice dipende dalla quantità di dati.

È possibile visualizzare valori diversi nel campo dello schema a seconda delle origini dati che hai configurato. La tabella seguente mostra i diversi valori dello schema e le descrizioni corrispondenti:

Schema	Description
OFF	Nessuna origine dati è configurata o indicizza ta.
REGISTRY	I dati di registro sono indicizzati.
REGISTRY_AND_SHADOW	I dati di registro e i dati shadow (classici) con nome sono indicizzati.
REGISTRY_AND_CONNETTIVITY	I dati di registro e i dati di connettività sono indicizzati.
REGISTRY_AND_SHADOW_AND_CON NECTIVITY_STATUS	I dati del registro, i dati shadow senza nome (classici) e i dati di connettività sono indicizzati.
MULTI_INDEXING_MODE	I dati delle copie shadow con nome o delle violazioni di Device Defender vengono indicizza ti, oltre ai dati del registro, delle copie shadow (classiche) senza nome o della connettività.

Esecuzione di query su un indice di oggetti

Utilizzare il comando dell'interfaccia a riga di comando search-index per eseguire query dei dati nell'indice.

```
aws iot search-index --index-name "AWS_Things" --query-string
          "thingName:mything*"
```


```
{
    "things":[{
         "thingName":"mything1",
         "thingGroupNames":[
            "mygroup1"
         ],
         "thingId":"a4b9f759-b0f2-4857-8a4b-967745ed9f4e",
         "attributes":{
            "attribute1":"abc"
         },
         "connectivity": {
            "connected":false,
            "timestamp":1556649874716,
            "disconnectReason": "CONNECTION_LOST"
         }
    },
    {
        "thingName":"mything2",
        "thingTypeName":"MyThingType",
        "thingGroupNames":[
            "mygroup1",
            "mygroup2"
        ],
        "thingId":"01014ef9-e97e-44c6-985a-d0b06924f2af",
        "attributes":{
            "model":"1.2",
            "country":"usa"
        },
        "shadow":{
            "desired":{
                "location":"new york",
                "myvalues":[3, 4, 5]
            },
            "reported":{
                "location":"new york",
```

```
                "myvalues":[1, 2, 3],
                "stats":{
                    "battery":78
                }
            },
            "metadata":{
                 "desired":{
                       "location":{
                            "timestamp":123456789
                        },
                       "myvalues":{
                            "timestamp":123456789
                       }
                 },
                 "reported":{
                       "location":{
                            "timestamp":34535454
                        },
                       "myvalues":{
                            "timestamp":34535454
                       },
                       "stats":{
                            "battery":{
                                 "timestamp":34535454
                            }
                       }
                }
            },
            "version":10,
            "timestamp":34535454
        },
        "connectivity": {
            "connected":true,
            "timestamp":1556649855046
        }
    }],
    "nextToken":"AQFCuvk7zZ3D9pOYMbFCeHbdZ+h=G"
}
```

Nella risposta JSON, "`connectivity`", abilitata dall'impostazione `thingConnectivityIndexingMode=STATUS`, fornisce un valore booleano, un timestamp e un disconnectReason che indica se il dispositivo è collegato ad AWS IoT Core. Il dispositivo

"mything1" scollegato (false) a POSIX time 1556649874716 a causa di CONNECTION_LOST. Per ulteriori informazioni sui motivi di disconnessione, consulta Eventi del ciclo di vita.

```
"connectivity": {
    "connected":false,
    "timestamp":1556649874716,
    "disconnectReason": "CONNECTION_LOST"
}
```

Il dispositivo "mything2" connesso (true) a POSIX time 1556649855046:

```
"connectivity": {
    "connected":true,
    "timestamp":1556649855046
}
```

I timestamp vengono forniti in millisecondi dall'epoca, perciò 1556649855046 rappresenta 6:44:15.046 PM di martedì 30 aprile 2019 (GMT).

> **Important**
>
> Se un dispositivo è stato disconnesso per circa un'ora, il valore "timestamp" e il valore "disconnectReason" dello stato di connettività potrebbe mancare.

Restrizioni e limitazioni

Queste sono le limitazioni e le restrizioni per AWS_Things.

Campi della copia shadow con valori di tipo complesso

Un campo della copia shadow viene indicizzato solo se il relativo valore è un tipo semplice, come un oggetto JSON che non contiene una matrice o una matrice composta interamente da tipi semplici. Per tipo semplice si intende una stringa, un numero o un valore letterale true o false. Ad esempio, per il seguente stato shadow, il valore del campo "palette" non viene indicizzato perché è una matrice che contiene elementi di tipo complesso. Il valore del campo "colors" viene indicizzato perché ogni valore della matrice è una stringa.

```
{
```

```
    "state": {
        "reported": {
            "switched": "ON",
            "colors": [ "RED", "GREEN", "BLUE" ],
            "palette": [
                {
                    "name": "RED",
                    "intensity": 124
                },
                {
                    "name": "GREEN",
                    "intensity": 68
                },
                {
                    "name": "BLUE",
                    "intensity": 201
                }
            ]
        }
    }
}
```

Nomi di campi della copia shadow nidificati

I nomi dei campi della copia shadow nidificati vengono archiviati come stringa delimitata da punto (.). Ad esempio, dato un documento shadow:

```
{
  "state": {
    "desired": {
      "one": {
        "two": {
          "three": "v2"
        }
      }
    }
  }
}
```

Il nome del campo `three` viene archiviato come `desired.one.two.three`. Se sei in possesso di un documento shadow, è archiviato in questo modo:

```
{
```

```
  "state": {
    "desired": {
      "one.two.three": "v2"
    }
  }
}
```

Entrambi corrispondono a una query per `shadow.desired.one.two.three:v2`. Come best practice, non utilizzare punti nei nomi dei campi shadow.

Metadati delle copie shadow

Un campo nella sezione metadati delle copie shadow viene indicizzato, ma solo se lo è anche il campo corrispondente nella sezione `"state"` della copia shadow. (Nell'esempio precedente, neanche il campo `"palette"` nella sezione dei metadati delle copie shadow verrà indicizzato.)

Dispositivi non registrati

L'indicizzazione del parco istanze consente di indicizzare lo stato di connettività di un dispositivo la cui connessione `clientId` è identica al `thingName` di un oggetto registrato nel Registro.

Copie shadow non registrate

Se UpdateThingShadowcrei un'ombra utilizzando il nome di un oggetto che non è stato registrato nel tuo AWS IoT account, i campi in questa ombra non vengono indicizzati. Questo vale sia per la classica shadow senza nome che per shadow con nome.

Valori numerici

Se eventuali dati del registro o dati shadow vengono riconosciuti dal servizio come valori numerici, verranno indicizzati come tali. Puoi formulare delle query che includono intervalli e operatori di confronto in merito ai valori numerici, ad esempio `"attribute.foo<5"` o `"shadow.reported.foo:[75 TO 80]"`. Per essere riconosciuto come numerico, il valore dei dati deve essere un numero JSON di tipo letterale valido. Il valore può essere un numero intero compreso nell'intervallo -2^53...2^53-1 o in virgola mobile a precisione doppia con notazione esponenziale opzionale, oppure una parte di una serie contenente solo questi valori.

Valori nulli

I valori nulli non sono indicizzati.

Valori massimi

Il numero massimo di campi personalizzati per le query di aggregazione è 5.

Il numero massimo di percentili richiesti per le query di aggregazione è 100.

Autorizzazione

Puoi specificare l'indice degli oggetti come Amazon Resource Name (ARN) in un'azione AWS IoT politica, come segue.

Azione	Risorsa
`iot:SearchIndex`	Un indice ARN (ad esempio, `arn:aws:iot:` *`your-aws-region your-aws-account`* `:index/AWS_Things`).
`iot:DescribeIndex`	Un indice ARN (ad esempio, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_Things`).

> **Note**
>
> Se si dispone di autorizzazioni per eseguire query all'indice del parco istanze, è possibile accedere ai dati degli oggetti sull'intero parco istanze.

Gestione dell'indicizzazione di gruppi di oggetti

`AWS_ThingGroups` è l'indice che contiene tutti i tuoi gruppi di oggetti. Questo indice consente di cercare gruppi in base al nome del gruppo, alla descrizione, agli attributi e a tutti i nomi del gruppo padre.

Abilitazione dell'indicizzazione di gruppi di oggetti

Puoi utilizzare l'`thing-group-indexing-configuration`impostazione nell'UpdateIndexingConfigurationAPI per creare l'`AWS_ThingGroups`indice e controllarne la configurazione. È possibile utilizzare l'GetIndexingConfigurationAPI per recuperare la configurazione di indicizzazione corrente.

Per aggiornare le configurazioni dell'indicizzazione dei gruppi di oggetti, esegui il comando CLI update-indexing-configuration:

```
aws iot update-indexing-configuration --thing-group-indexing-configuration
 thingGroupIndexingMode=ON
```

Puoi inoltre aggiornare le configurazioni per l'indicizzazione sia degli oggetti che dei gruppi di oggetti con un unico comando, come descritto di seguito:

```
aws iot update-indexing-configuration --thing-indexing-configuration
 thingIndexingMode=REGISTRY --thing-group-indexing-configuration
 thingGroupIndexingMode=ON
```

Di seguito sono riportati i valori validi per `thingGroupIndexingMode`.

OFF

Nessun indice. indexing/delete

ATTIVATO

Creazione o configurazione dell'indice `AWS_ThingGroups`.

Utilizzare il comando CLI get-indexing-configuration per recuperare l'attuale oggetto e le configurazioni di indicizzazione del gruppo di oggetti:

```
aws iot get-indexing-configuration
```

La risposta del comando è simile alla seguente:

```
{
   "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "ON"
    }
}
```

Descrizione degli indici di gruppi

Per recuperare lo stato corrente dell'indice `AWS_ThingGroups`, utilizzare il comando CLI describe-index:

```
aws iot describe-index --index-name "AWS_ThingGroups"
```

La risposta del comando è simile alla seguente:

```
{
   "indexStatus": "ACTIVE",
   "indexName": "AWS_ThingGroups",
   "schema": "THING_GROUPS"
}
```

AWS IoT crea l'indice la prima volta che esegui l'indicizzazione. Non è possibile eseguire query dell'indice se `indexStatus` è `BUILDING`.

Esecuzione di query su un indice di gruppi di oggetti

Utilizzare il comando CLI search-index per eseguire query dei dati nell'indice:

```
aws iot search-index --index-name "AWS_ThingGroups" --query-string
 "thingGroupName:mythinggroup*"
```

Autorizzazione

È possibile specificare l'indice dei gruppi di oggetti come ARN di risorse in un'azione AWS IoT politica, come segue.

Azione	Risorsa
`iot:SearchIndex`	Un indice ARN (ad esempio, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).
`iot:DescribeIndex`	Un indice ARN (ad esempio, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).

Interrogazioni sullo stato della connettività dei dispositivi

AWS IoT Fleet Indexing supporta l'interrogazione della connettività dei singoli dispositivi, consentendoti di recuperare in modo efficiente lo stato della connettività e i relativi metadati per dispositivi specifici. Questa funzionalità integra le funzionalità di indicizzazione e interrogazione esistenti a livello di flotta.

Come funziona

Il supporto per le query di connettività dei dispositivi può essere utilizzato per il recupero ottimizzato dello stato di connettività di un singolo dispositivo. Questa API fornisce un accesso a bassa latenza e ad alta velocità effettiva alle informazioni di connettività specifiche del dispositivo più recenti. Una volta abilitata l'indicizzazione della connettività, avrai accesso a questa API di query che verrà addebitata come query standard. Per ulteriori informazioni, consulta AWS IoT i prezzi di Device Management

Funzionalità

Con il supporto per le richieste di connettività dei dispositivi, puoi:

1. Interroga lo stato di connettività corrente (connesso o disconnesso) per un determinato dispositivo utilizzando il su`thingName`.
2. Recupera metadati di connettività aggiuntivi, tra cui:
   a. Motivo di disconnessione
   b. Timestamp per l'evento di connessione o disconnessione più recente.

> **Note**
>
> L'indicizzazione del parco istanze consente di indicizzare lo stato di connettività di un dispositivo la cui connessione `clientId` è identica al `thingName` di un oggetto registrato nel Registro.

Vantaggi

1. Bassa latenza: riflette lo stato di connettività del dispositivo più recente e offre una bassa latenza per riflettere le modifiche dello stato di connessione rispetto a IoT Core. IoT Core determina che un dispositivo è disconnesso non appena riceve una richiesta di disconnessione dal dispositivo o nel caso in cui un dispositivo si disconnetta senza inviare una richiesta di disconnessione. Il core IoT attenderà 1,5 volte il tempo di keep-alive configurato prima che il client venga determinato come disconnesso. L'API Connectivity status rifletterà queste modifiche in genere entro un secondo dopo che IoT Core determina la modifica dello stato di connessione di un dispositivo.
2. Throughput elevato: supporta 350 transazioni al secondo (TPS) per impostazione predefinita e può essere impostato su un valore superiore su richiesta.

3. Conservazione dei dati: memorizza i dati degli eventi a tempo indeterminato quando la ConnectivityIndexing modalità Fleet Indexing (FI) è abilitata e l'oggetto non viene eliminato. Se disabiliti l'indicizzazione della connettività, i record non verranno conservati.

> **Note**
>
> Se l'indicizzazione dello stato di connettività era abilitata prima del lancio di questa API, Fleet Indexing inizia a tenere traccia delle modifiche allo stato della connettività dopo il lancio dell'API e riflette lo stato aggiornato in base a tali modifiche.

Prerequisiti

Per utilizzare il supporto per le query di connettività del dispositivo:

1. Configura un AWS account
2. Effettua l'onboarding e registra i dispositivi AWS IoT Core nella tua regione preferita
3. Abilita l'indicizzazione del parco veicoli con l'indicizzazione della connettività

> **Note**
>
> Non è richiesta alcuna configurazione aggiuntiva se l'indicizzazione della connettività è già abilitata

Per istruzioni dettagliate sulla configurazione, consulta la Guida per gli sviluppatori AWS IoT

Esempi

```
aws iot get-thing-connectivity-data --thing-name myThingName
```

```
{
   "connected": true,
   "disconnectReason": "NONE",
   "thingName": "myThingName",
```

```
   "timestamp": "2024-12-19T10:00:00.000000-08:00"
}
```

- `thingName`: il nome del dispositivo indicato nella richiesta. Corrisponde anche al ClientID utilizzato per la connessione. AWS IoT Core
- `disconnectReason`: Motivo della disconnessione. Sarà NONE per un dispositivo connesso.
- `connected`: Il valore booleano true che indica che questo dispositivo è attualmente connesso.
- `timestamp`: Il timestamp che rappresenta la disconnessione più recente del dispositivo in millisecondi.

```
aws iot get-thing-connectivity-data --thing-name myThingName
```

```
{
   "connected": false,
   "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
   "thingName": "myThingName",
   "timestamp": "2024-12-19T10:30:00.000000-08:00"
}
```

- `thingName`: il nome del dispositivo indicato nella richiesta. Corrisponde anche al ClientID utilizzato per la connessione. AWS IoT Core
- `disconnectReason`: Il motivo della disconnessione è CLIENT_INITIATED_DISCONNECT che indica il client indicato che si sarebbe disconnesso. AWS IoT Core
- `connected`: Il valore booleano false che indica che questo dispositivo è attualmente disconnesso.
- `timestamp`: Il timestamp che rappresenta la disconnessione più recente del dispositivo in millisecondi.

```
aws iot get-thing-connectivity-data --thing-name neverConnectedThing
```

```
{
   "connected": false,
   "disconnectReason": "UNKNOWN",
   "thingName": "neverConnectedThing"
```

```
}
```

- `thingName`: il nome del dispositivo indicato nella richiesta. Corrisponde anche al ClientID utilizzato per la connessione. AWS IoT Core
- `disconnectReason`: Motivo della disconnessione. Sarà «SCONOSCIUTO» per un dispositivo che non è mai stato connesso o per il quale Fleet Indexing non ha memorizzato l'ultimo motivo di disconnessione.
- `connected`: Il valore booleano false che indica che questo dispositivo è attualmente disconnesso.
- `timestamp`: Il timestamp non viene restituito per un dispositivo che non è mai stato connesso o per il quale Fleet Indexing non ha l'ultimo timestamp memorizzato.

Interrogazione di dati aggregati

AWS IoT fornisce quattro APIs (`GetStatistics`, `GetCardinalityGetPercentiles`, `eGetBucketsAggregation`) che consentono di cercare dati aggregati nel parco dispositivi.

> **Note**
>
> Per problemi relativi ai valori mancanti o imprevisti per l'aggregazione APIs, leggi la guida alla risoluzione dei problemi di indicizzazione del parco veicoli.

GetStatistics

L'GetStatisticsAPI e il comando get-statistics CLI restituiscono il conteggio, la media, la somma, il minimo, il massimo, la somma dei quadrati, la varianza e la deviazione standard per il campo aggregato specificato.

Il comando CLI get-statistics accetta i parametri seguenti:

`index-name`

Nome dell'indice su cui eseguire una ricerca. Il valore predefinito è `AWS_Things`.

`query-string`

La query usata per eseguire la ricerca nell'indice. Puoi specificare di ottenere il conteggio `"*"` di tutte le cose indicizzate nel tuo. Account AWS

`aggregationField`

(Facoltativo) Il campo da aggregare. Questo campo deve essere un campo gestito o personalizzato definito quando si chiama update-indexing-configuration. Se non specifichi un campo di aggregazione, viene utilizzato `registry.version` come campo di aggregazione.

`query-version`

La versione della query da usare. Il valore predefinito è `2017-09-30`.

Il tipo di campo di aggregazione può influenzare le statistiche restituite.

GetStatistics con valori di stringa

Se si aggrega in un campo stringa, la chiamata `GetStatistics` restituisce un conteggio di dispositivi con attributi che corrispondono alla query. Ad esempio:

```
aws iot get-statistics --aggregation-field 'attributes.stringAttribute'
            --query-string '*'
```

Questo comando restituisce il numero di dispositivi che contengono un attributo denominato `stringAttribute`:

```
{
  "statistics": {
    "count": 3
  }
}
```

GetStatistics con valori booleani

Quando chiami `GetStatistics` con un campo di aggregazione booleano:

- AVERAGE è la percentuale di dispositivi che corrispondono alla query.
- MINIMUM è 0 o 1 secondo le seguenti regole:
  - Se tutti i valori per il campo di aggregazione sono `false`, MINIMUM è 0.
  - Se tutti i valori per il campo di aggregazione sono `true`, MINIMUM è 1.
  - Se i valori per il campo di aggregazione sono una miscela di `false` e `true`, MINIMUM è 0.

- MAXIMUM è 0 o 1 secondo le seguenti regole:
  - Se tutti i valori per il campo di aggregazione sono `false`, MAXIMUM è 0.
  - Se tutti i valori per il campo di aggregazione sono `true`, MAXIMUM è 1.
  - Se i valori per il campo di aggregazione sono una miscela di `false` e `true`, MAXIMUM è 1.
- SUM è la somma dell'equivalente intero dei valori booleani.
- COUNT è il numero di oggetti che corrispondono ai criteri della stringa di query e contengono un valore di campo di aggregazione valido.

GetStatistics con valori numerici

Quando si chiama `GetStatistics` e si specifica un campo di aggregazione di tipo `Number`, `GetStatistics` restituisce i seguenti valori:

count

Numero di oggetti che corrispondono ai criteri della stringa di query e contengono un valore di campo di aggregazione valido.

average

La media dei valori numerici che corrispondono alla query.

sum

La somma dei valori numerici che corrispondono alla query.

minimum

Il più piccolo dei valori numerici che corrisponde alla query.

maximum

Il più grande dei valori numerici che corrisponde alla query.

sumOfSquares

La somma dei quadrati dei valori numerici che corrispondono alla query.

variance

La varianza dei valori numerici che corrispondono alla query. La varianza di un insieme di valori è la media dei quadrati delle differenze di ciascun valore rispetto al valore medio del set.

stdDeviation

La deviazione standard dei valori numerici che corrispondono alla query. La deviazione standard di un insieme di valori è una misura di quanto sono distribuiti i valori.

Nell'esempio seguente viene illustrato come chiamare get-statistics con un campo personalizzato numerico.

```
aws iot get-statistics --aggregation-field 'attributes.numericAttribute2'
            --query-string '*'
```

```
{
  "statistics": {
    "count": 3,
    "average": 33.333333333333336,
    "sum": 100.0,
    "minimum": -125.0,
    "maximum": 150.0,
    "sumOfSquares": 43750.0,
    "variance": 13472.22222222222,
    "stdDeviation": 116.06990230986766
  }
}
```

Per i campi di aggregazione numerica, se i valori dei campi superano il valore doppio massimo, i valori delle statistiche sono vuoti.

GetCardinality

L'GetCardinalityAPI e il comando get-cardinality CLI restituiscono il conteggio approssimativo di valori univoci che corrispondono alla query. Ad esempio, potresti cercare il numero di dispositivi con livelli di batteria inferiori al 50%:

```
aws iot get-cardinality --index-name AWS_Things --query-string "batterylevel
          > 50" --aggregation-field "shadow.reported.batterylevel"
```

Questo comando restituisce il numero di elementi con livelli di batteria oltre il 50%:

```
{
```

```
    "cardinality": 100
}
```

`cardinality` viene sempre restituito da get-cardinality anche se non ci sono campi corrispondenti. Ad esempio:

```
aws iot get-cardinality --query-string "thingName:Non-existent*"
          --aggregation-field "attributes.customField_STR"
```

```
{
    "cardinality": 0
}
```

Il comando CLI get-cardinality accetta i parametri seguenti:

`index-name`

Nome dell'indice su cui eseguire una ricerca. Il valore predefinito è `AWS_Things`.

`query-string`

La query usata per eseguire la ricerca nell'indice. Puoi specificare di `"*"` ottenere il conteggio di tutte le cose indicizzate nel tuo. Account AWS

`aggregationField`

Il campo da aggregare.

`query-version`

La versione della query da usare. Il valore predefinito è `2017-09-30`.

GetPercentiles

L'GetPercentilesAPI e il comando get-percentiles CLI raggruppano i valori aggregati che corrispondono alla query in raggruppamenti percentili. I raggruppamenti percentili predefiniti sono: 1, 5, 25, 50, 75, 95, 99, anche se è possibile specificare il proprio quando si chiama `GetPercentiles`. Questa funzione restituisce un valore per ogni gruppo percentile specificato (o i raggruppamenti percentili predefiniti). Il gruppo percentile "1" contiene il valore del campo aggregato che si verifica in circa l'1% dei valori corrispondenti alla query. Il gruppo percentile "5" contiene il valore del campo

aggregato che si verifica in circa il cinque percento dei valori corrispondenti alla query e così via. Il risultato è un'approssimazione, più valori corrispondono alla query, più precisi sono i valori percentili.

Nell'esempio seguente viene illustrato come chiamare il comando CLI get-percentiles.

```
aws iot get-percentiles --query-string "thingName:*" --aggregation-field
          "attributes.customField_NUM" --percents 10 20 30 40 50 60 70 80 90 99
```

```
{
    "percentiles": [
        {
            "value": 3.0,
            "percent": 80.0
        },
        {
            "value": 2.5999999999999996,
            "percent": 70.0
        },
        {
            "value": 3.0,
            "percent": 90.0
        },
        {
            "value": 2.0,
            "percent": 50.0
        },
        {
            "value": 2.0,
            "percent": 60.0
        },
        {
            "value": 1.0,
            "percent": 10.0
        },
        {
            "value": 2.0,
            "percent": 40.0
        },
        {
            "value": 1.0,
            "percent": 20.0
        },
        {
```

```
            "value": 1.4,
            "percent": 30.0
        },
        {
            "value": 3.0,
            "percent": 99.0
        }
    ]
}
```

Il comando seguente mostra l'output restituito da get-percentiles quando non ci sono documenti corrispondenti.

```
aws iot get-percentiles --query-string "thingName:Non-existent*"
          --aggregation-field "attributes.customField_NUM"
```

```
{
    "percentiles": []
}
```

Il comando CLI get-percentile accetta i parametri seguenti:

`index-name`

Nome dell'indice su cui eseguire una ricerca. Il valore predefinito è `AWS_Things`.

`query-string`

La query usata per eseguire la ricerca nell'indice. Puoi specificare di `"*"` ottenere il conteggio di tutte le cose indicizzate nel tuo. Account AWS

`aggregationField`

Il campo da aggregare, che deve essere di tipo `Number`.

`query-version`

La versione della query da usare. Il valore predefinito è `2017-09-30`.

`percents`

(Facoltativo) È possibile utilizzare questo parametro per specificare raggruppamenti percentili personalizzati.

GetBucketsAggregation

L'GetBucketsAggregationAPI e il comando get-buckets-aggregation CLI restituiscono un elenco di bucket e il numero totale di elementi che soddisfano i criteri della stringa di query.

Nell'esempio seguente viene illustrato come chiamare il comando CLI get-buckets-aggregation.

```
aws iot get-buckets-aggregation --query-string '*' --index-name AWS_Things --
aggregation-field 'shadow.reported.batterylevelpercent' --buckets-aggregation-type
 'termsAggregation={maxBuckets=5}'
```

Questo comando restituisce quanto segue:

```
{
    "totalCount": 20,
    "buckets": [
        {
            "keyValue": "100",
            "count": 12
        },
        {
            "keyValue": "90",
            "count": 5
        },
        {
            "keyValue": "75",
            "count": 3
        }
    ]
}
```

Il comando CLI get-buckets-aggregation accetta i parametri seguenti:

`index-name`

Nome dell'indice su cui eseguire una ricerca. Il valore predefinito è `AWS_Things`.

`query-string`

La query usata per eseguire la ricerca nell'indice. Puoi specificare di `"*"` ottenere il conteggio di tutte le cose indicizzate nel tuo. Account AWS

`aggregation-field`

Il campo da aggregare.

`buckets-aggregation-type`

Il controllo di base della forma di risposta e il tipo di aggregazione del bucket da eseguire.

Autorizzazione

È possibile specificare l'indice dei gruppi di oggetti come ARN di risorse in un'azione AWS IoT politica, come segue.

Azione	Risorsa
`iot:GetStatistics`	Un indice ARN (ad esempio `arn:aws:iot:` *`your-aws-region`* `:index/AWS_Things` o `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).

Sintassi delle query

Nell'indicizzazione del parco istanze, si utilizza una sintassi di query per specificare le query.

Funzionalità supportate

La sintassi di query supporta le seguenti caratteristiche:

- Termini e frasi
- Ricerca nei campi
- Ricerca di prefissi
- Ricerca di intervalli
- Operatori booleani `AND`, `OR`, `NOT` e `-`. Il trattino viene utilizzato per escludere qualcosa dai risultati di ricerca (ad esempio, `thingName:(tv* AND -plasma)`).
- Raggruppamento
- Raggruppamento di campi
- Caratteri di escape speciali (come con \)

- L'uso principale dei caratteri jolly è limitato a 1 termine di query per query. Ad esempio, non è possibile cercare «`thingName:*mye`» `thingGroupNames:*echo` nella stessa query. Le interrogazioni che includono un carattere jolly iniziale hanno una lunghezza massima di 100 caratteri.

> **Note**
>
> Gli account con un limite API di query Fleet Indexing superiore a 30 richieste al secondo sono soggetti a limitazioni quando utilizzano i termini di query principali con wildcard. Questo requests-per-second limite di 30 si applica a livello di account per tutte le query di Fleet Indexing. APIs

Caratteristiche non supportate

La sintassi di query non supporta le seguenti caratteristiche:

- Espressioni regolari
- Aumento priorità
- Classificazione
- Ricerche fuzzy
- Ricerca per prossimità
- Ordinamento
- Aggregazione
- Caratteri speciali: `` ` ``, `@`, `#`, `%`, `\`, `/`, `'`, `;` e `,`. Tieni presente che `,` è supportato solo nelle geoquery.

Note

Alcune note relative al linguaggio di query:

- L'operatore predefinito è AND. Una query per `"thingName:abc thingType:xyz"` equivale a `"thingName:abc AND thingType:xyz"`.
- Se non viene specificato un campo, AWS IoT cerca il termine in tutti i campi del Registro di sistema, Device Shadow e Device Defender.
- Tutti i nomi di campo fanno distinzione tra maiuscole e minuscole.

- La ricerca non fa distinzione tra maiuscole e minuscole. Le parole sono separate da spazi come definito dal metodo di Java `Character.isWhitespace(int)`.
- L'indicizzazione di dati Device Shadow (shadow senza nome e shadow con nome) include le sezioni reported, desired, delta e metadata.
- Non è possibile eseguire ricerche nelle versioni Device Shadow e del registro dei dispositivi, ma queste sono presenti nella risposta.
- Il numero massimo di termini in una query è 12.
- Il carattere speciale `,` è supportato solo nelle geoquery.

Esempio di query per oggetti

Specificare query in una stringa di query utilizzando una sintassi di query. Le query vengono passate all'API `SearchIndex`. La tabella seguente elenca alcune stringhe di query di esempio.

Stringa di query	Risultato
`abc`	Esegue una query per il valore "abc" in qualsiasi campo di registro, copia shadow (copia shadow classica senza nome e copia shadow con nome) o violazioni Device Defender.
`thingName:myThingName`	Interrogazioni per un oggetto con nome "». myThingName
`thingName:my*`	Query per oggetti con nomi che iniziano per "mio".
`thingName:*my`	Interrogazioni relative a elementi con nomi che terminano con «my». L'uso principale dei caratteri jolly è limitato a 1 termine di query per query. Le interrogazioni che includono un carattere jolly iniziale hanno una lunghezza massima di 100 caratteri.
`thingName:*my*`	Interrogazioni per elementi che contengono la sottostri nga «my». L'uso principale dei caratteri jolly è limitato a 1 termine di query per query. Le interrogazioni che includono un carattere jolly iniziale hanno una lunghezza massima di 100 caratteri.

Stringa di query	Risultato
`thingName:ab?`	Query per oggetti con nomi che contengono "ab" più un altro carattere (ad esempio "aba", "abb", "abc" e così via).
`thingTypeName:aa`	Query per oggetti associati al tipo "aa".
`thingGroupNames:a`	Interrogazioni relative a elementi il cui nome del gruppo di oggetti principale o del gruppo di fatturazione è «a».
`thingGroupNames:a*`	Interrogazioni relative a elementi il cui nome del gruppo di oggetti principale o del gruppo di fatturazione corrisponde allo schema «a*».
`attributes.myAttribute:75`	Query per oggetti con un attributo denominato "myAttrib ute" il cui valore è 75.
`attributes.myAttribute: [75 TO 80]`	Query per oggetti con un attributo denominato "myAttrib ute" il cui valore è compreso in un intervallo numerico (75-80, inclusi).
`attributes.myAttribute: {75 TO 80]`	Query per oggetti con un attributo denominato "myAttrib ute" il cui valore è compreso in un intervallo numerico (>75 e <=80).
`attributes.serialNumber: ["abcd" TO "abcf"]`	Query per oggetti con un attributo denominato "serialNu mber" il cui valore è compreso in un intervallo di stringhe alfanumeriche. Questa query restituisce gli oggetti con un attributo "serialNumber" con i valori "abcd", "abce" o "abcf".
`attributes.myAttri bute:i*t`	Query per oggetti con un attributo denominato "myAttrib ute" il cui valore è "i", seguito da un numero qualsiasi di caratteri, seguito da 't'.

Stringa di query	Risultato
`attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10`	Query per oggetti che combinano termini usando espressioni booleane. Questa query restituisce gli oggetti che hanno un attributo denominato "attr1" con un valore "abc", un attributo denominato "attr2" il cui valore è minore di 5 e un attributo denominato "attr3" che non è maggiore di 10.
`shadow.hasDelta:true`	Query per oggetti con una copia shadow senza nome che ha un elemento delta.
`NOT attributes.model:legacy`	Query per oggetti in cui l'attributo denominato "model" non è "legacy".
`shadow.reported.stats.battery:{70 TO 100} (v2 OR v3) NOT attributes.model:legacy`	Query per oggetti con le caratteristiche seguenti: • L'attributo `stats.battery` della copia shadow dell'oggetto ha un valore compreso tra 70 e 100. • Il testo "v2" o "v3" è contenuto in un nome di oggetto, un nome di tipo o in valori di attributo. • L'attributo `model` dell'oggetto non è impostato su "legacy".
`shadow.reported.myvalues:2`	Query per oggetti in cui la serie `myvalues` nella sezione reported della copia shadow contiene un valore pari a 2.
`shadow.reported.location:* NOT shadow.desired.stats.battery:*`	Query per oggetti con le caratteristiche seguenti: • L'attributo `location` è presente nella sezione `reported` della copia shadow. • L'attributo `stats.battery` non è presente nella sezione `desired` della copia shadow.
`shadow.name.<shadowName>.hasDelta:true`	Query per oggetti che hanno una copia shadow con nome specificato e anche un elemento delta.

Stringa di query	Risultato
`shadow.name.<shadowName>.desired.filament:*`	Query per oggetti che hanno una copia shadow con nome specificato e anche una proprietà del filamento desiderata.
`shadow.name.<shadowName>.reported.location:*`	Query per oggetti che hanno una copia shadow con nome specificato e dove l'attributo `location` è presente nella sezione segnalata di shadow con nome.
`connectivity.connected:true`	Interroga tutti i dispositivi connessi.
`connectivity.connected:false`	Interroga tutti i dispositivi disconnessi.
`connectivity.connected:true AND connectivity.timestamp : [1557651600000 TO 1557867600000]`	Interroga tutti i dispositivi connessi con un timestamp di connessione >= 1557651600000 e <= 1557867600000. I timestamp vengono forniti in millisecondi dall'epoch.
`connectivity.connected:false AND connectivity.timestamp : [1557651600000 TO 1557867600000]`	Interroga tutti i dispositivi disconnessi con un timestamp di disconnessione >= 1557651600000 e <= 1557867600000. I timestamp vengono forniti in millisecondi dall'epoch.
`connectivity.connected:true AND connectivity.timestamp > 1557651600000`	Interroga tutti i dispositivi connessi con un timestamp di connessione > 557651600000. I timestamp vengono forniti in millisecondi dall'epoch.
`connectivity.connected:*`	Ricerche di tutti i dispositivi con informazioni sulla connettività.
`connectivity.disconnectReason:*`	Query per tutti i dispositivi con connettività disconnectReason presente.

Stringa di query	Risultato
`connectivity.disconnectReason:CLIENT_INITIATED_DISCONNECT`	Query per tutti i dispositivi disconnessi a causa di CLIENT_INITIATED_DISCONNECT.
`deviceDefender.violationCount:[0 TO 100]`	Le query per oggetti con violazioni di Device Defender contano valori che rientrano nell'intervallo numerico (0-100, incluso).
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.inViolation:true`	Query per oggetti che sono in violazione del comportamento `disconnectBehavior` come definito nel profilo di sicurezza `device-SecurityProfile` . Nota che InViolation:false non è una query valida.
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.lastViolationValue.number>2`	Interrogazioni relative a elementi che violano il comportamento `disconnectBehavior` definito nel dispositivo del profilo di sicurezza, SecurityProfile con un valore dell'evento dell'ultima violazione maggiore di 2.
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.lastViolationTime>1634227200000`	Interrogazioni relative a elementi che violano il comportamento `disconnectBehavior` definito nel dispositivo del profilo di sicurezza, SecurityProfile con un ultimo evento di violazione dopo un periodo di tempo specificato.
`shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km`	Interrogazioni per oggetti che si trovano entro la distanza radiale di 15,5 km dalle coordinate di 47.6204, -122.3491 . Questa stringa di query si applica a quando i dati sulla posizione vengono archiviati in un'ombra denominata.
`shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km`	Interrogazioni per oggetti che si trovano entro la distanza radiale di 15,5 km dalle coordinate di 47.6204, -122.3491 . Questa stringa di query si applica a quando i dati sulla posizione vengono archiviati in un'ombra classica.

Esempio di query per gruppi di oggetti

Le query vengono specificate in una stringa di query usando una sintassi di query, quindi vengono passate all'API `SearchIndex`. La tabella seguente elenca alcune stringhe di query di esempio.

Stringa di query	Risultato
`abc`	Query per "abc" in qualsiasi campo.
`thingGroupName:myGroupThingName`	Interrogazioni per un gruppo di oggetti con nome "myGroupThingNome».
`thingGroupName:my*`	Query per gruppi di oggetti con nomi che iniziano per "my".
`thingGroupName:ab?`	Query per gruppi di oggetti con nomi che contengono "ab" più un altro carattere, ad esempio "aba", "abb", "abc" e così via.
`attributes.myAttribute:75`	Query per gruppi di oggetti con un attributo denominato "myAttribute" il cui valore è 75.
`attributes.myAttribute:[75 TO 80]`	Query per gruppi di oggetti con un attributo denominato "myAttribute" il cui valore è compreso in un intervallo numerico (75-80, inclusi).
`attributes.myAttribute:[75 TO 80]`	Query per gruppi di oggetti con un attributo denominato "myAttribute" il cui valore è compreso in un intervallo numerico (tra >75 e <=80).
`attributes.myAttribute:["abcd" TO "abcf"]`	Query per gruppi di oggetti con un attributo denominato "myAttribute" il cui valore è compreso in un intervallo di stringhe alfanumeriche. Questa query restituisce i gruppi di oggetti con un attributo "serialNumber" con i valori "abcd", "abce" o "abcf".
`attributes.myAttribute:i*t`	Query per gruppi di oggetti con un attributo denominato "myAttribute" il cui valore è "i", seguito da un numero qualsiasi di caratteri, seguito da "t".

Stringa di query	Risultato
`attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10`	Query per gruppi di oggetti che combinano termini usando espressioni booleane. Questa query restituis ce i gruppi di oggetti che hanno un attributo denominat o "attr1" con un valore "abc", un attributo denominato "attr2" il cui valore è minore di 5 e un attributo denominat o "attr3" che non è maggiore di 10.
`NOT attributes.myAttri bute:cde`	Query per gruppi di oggetti in cui l'attributo denominato "myAttribute" non è "cde".
`parentGroupNames:( myParentThingGroupName)`	Interrogazioni per gruppi di oggetti il cui nome del gruppo principale corrisponde a "» myParentThingGroupName.
`parentGroupNames:( myParentThingGroupName OR myRootThingGroupName)`	Interrogazioni per gruppi di oggetti il cui nome del gruppo principale corrisponde a "myParentThingGroupName" o "» myRootThingGroupName.
`parentGroupNames:( myParentThingGroupNa*)`	Interrogazioni per gruppi di cose il cui nome del gruppo principale inizia con "» myParentThingGroupNa.

Indicizzazione dei dati sulla posizione

Puoi utilizzare l'indicizzazione del AWS IoT parco veicoli per indicizzare gli ultimi dati sulla posizione inviati dai tuoi dispositivi e cercare i dispositivi utilizzando le geoquery. Questa funzionalità risolve casi d'uso per il monitoraggio e la gestione dei dispositivi come il tracciamento della posizione e la ricerca di prossimità. L'indicizzazione della posizione funziona in modo simile ad altre funzionalità di indicizzazione del parco veicoli e prevede configurazioni aggiuntive da specificare nell'indicizzazione degli oggetti.

I casi d'uso più comuni includono: ricerca e aggregazione di dispositivi situati entro i confini geografici desiderati, acquisizione di informazioni specifiche sulla posizione utilizzando termini di query relativi ai metadati e allo stato del dispositivo da fonti di dati indicizzate, fornitura di una visualizzazione granulare, ad esempio filtrando i risultati in base a un'area geografica specifica per ridurre i ritardi di rendering nelle mappe di monitoraggio del parco veicoli e tracciare l'ultima posizione del dispositivo segnalata, identificare i dispositivi che si trovano al di fuori dei limiti di confine desiderati e generare

allarmi utilizzando le metriche della flotta. Per iniziare con l'indicizzazione della posizione e le geoquery, consulta???.

Formati di dati supportati

AWS IoT l'indicizzazione della flotta supporta i seguenti formati di dati sulla posizione:

1. Rappresentazione testuale ben nota dei sistemi di riferimento di coordinate

   Una stringa che segue il formato Informazioni geografiche - Rappresentazione testuale ben nota dei sistemi di riferimento di coordinate. Un esempio può essere`"POINT(long lat)"`.
2. Una stringa che rappresenta le coordinate

   Una stringa con il formato `"latitude, longitude"` o`"longitude, latitude"`. Se si utilizza`"longitude, latitude"`, è necessario specificare anche `order` in`geoLocations`. Un esempio può essere`"41.12,-71.34"`.
3. Un oggetto composto da tasti lat (latitudine), lon (longitudine)

   Questo formato è applicabile all'ombra classica e all'ombra denominata. Chiavi supportate: `lat`latitude,`lon`,,`long`,`longitude`. Un esempio può essere`{"lat": 41.12, "lon": -71.34}`.
4. Un array che rappresenta le coordinate

   Una matrice con il formato `[lat,lon]` o`[lon,lat]`. Se si utilizza il formato`[lon,lat]`, che è lo stesso delle coordinate in GeoJSON (applicabile all'ombra classica e all'ombra denominata), è necessario specificare anche in. `order` `geoLocations`

   Un esempio può essere:

```
{
  "location": {
    "coordinates": [
      **Longitude**,
      **Latitude**
    ],
    "type": "Point",
    "properties": {
      "country": "United States",
      "city": "New York",
      "postalCode": "*****",
```

```
      "horizontalAccuracy": 20,
      "horizontalConfidenceLevel": 0.67,
      "state": "New York",
      "timestamp": "2023-01-04T20:59:13.024Z"
    }
  }
}
```

Come indicizzare i dati sulla posizione

I passaggi seguenti mostrano come aggiornare la configurazione di indicizzazione per i dati sulla posizione e utilizzare le geoquery per cercare dispositivi.

1. Scopri dove sono archiviati i dati sulla tua posizione

   L'indicizzazione della flotta attualmente supporta l'indicizzazione dei dati sulla posizione memorizzati nelle ombre classiche o nelle ombre denominate.

2. Utilizza i formati di dati sulla posizione supportati

   Assicurati che il formato dei dati sulla posizione segua uno dei formati di dati supportati.

3. Aggiorna la configurazione di indicizzazione

   Se necessario, abilita la configurazione di indicizzazione di thing (registry). È inoltre necessario abilitare l'indicizzazione su shadow classic o named shadow che contengono i dati sulla posizione. Quando si aggiorna l'indicizzazione degli oggetti, è necessario includere i dati sulla posizione nella configurazione di indicizzazione.

4. Crea ed esegui geoquery

   A seconda dei casi d'uso, crea geoquery ed eseguile per cercare dispositivi. La geoquery che componi deve seguire la sintassi Query. Alcuni esempi sono disponibili in ???.

Aggiorna la configurazione di indicizzazione degli oggetti

Per indicizzare i dati sulla posizione, è necessario aggiornare la configurazione di indicizzazione e includere i dati sulla posizione. A seconda di dove sono archiviati i dati sulla posizione, segui i passaggi per aggiornare la configurazione di indicizzazione:

Dati sulla posizione memorizzati nelle ombre classiche

Se i dati sulla posizione sono memorizzati in un'ombra classica, è `thingIndexingMode` necessario `REGISTRY_AND_SHADOW` impostarli e specificarli nei `geoLocations` campi (`nameeorder`) in `filter`.

Nel seguente esempio di configurazione di indicizzazione degli oggetti, specificate il percorso dei dati sulla posizione `shadow.reported.coordinates` così `name` e `LonLat` come. `order`

```
{
 "thingIndexingMode": "REGISTRY_AND_SHADOW",
 "filter": {
  "geoLocations": [
   {
    "name": "shadow.reported.coordinates",
    "order": "LonLat"
   }
  ]
 }
}
```

- `thingIndexingMode`

  La modalità di indicizzazione controlla se il registro o lo shadow sono indicizzati. Quando `thingIndexingMode` è impostato su `beOFF`, l'indicizzazione degli oggetti è disabilitata.

  Per indicizzare i dati sulla posizione memorizzati in un'ombra classica, è necessario `thingIndexingMode` impostare su be. `REGISTRY_AND_SHADOW` Per ulteriori informazioni, consulta ???.

- `filter`

  Il filtro di indicizzazione fornisce selezioni aggiuntive per ombre denominate e dati di geolocalizzazione. Per ulteriori informazioni, consulta ???.

- `geoLocations`

  L'elenco degli obiettivi di geolocalizzazione che scegli di indicizzare. Il numero massimo predefinito di obiettivi di geolocalizzazione per l'indicizzazione è. 1 Per aumentare il limite, vedi Quote.AWS IoT Device Management

- `name`

Il nome del campo di destinazione della geolocalizzazione. Un valore di esempio di `name` può essere il percorso dei dati sulla posizione della tua ombra:. `shadow.reported.coordinates`

- `order`

  L'ordine del campo di destinazione della geolocalizzazione. Valori validi: `LatLon` e. `LonLat` `LatLon`significa latitudine e longitudine. `LonLat`significa longitudine e latitudine. Questo campo è facoltativo. Il valore predefinito è `LatLon`.

Dati sulla posizione memorizzati in ombre denominate

Se i dati sulla posizione sono memorizzati in un'ombra denominata, `namedShadowIndexingMode` impostate su be`ON`, aggiungete il nome o i nomi delle ombre denominate al `namedShadowNames` campo in `filter`e specificate il percorso dei dati sulla posizione nel `geoLocations` campo in `filter`.

Nel seguente esempio di configurazione di indicizzazione degli oggetti, specificate il percorso dei dati della posizione `shadow.name.namedShadow1.reported.coordinates` così `name` e `LonLat` come. `order`

```
{
 "thingIndexingMode": "REGISTRY",
 "namedShadowIndexingMode": "ON",
 "filter": {
  "namedShadowNames": [
   "namedShadow1"
  ],
  "geoLocations": [
   {
    "name": "shadow.name.namedShadow1.reported.coordinates",
    "order": "LonLat"
   }
  ]
 }
}
```

- `thingIndexingMode`

  La modalità di indicizzazione controlla se il registro o lo shadow sono indicizzati. Quando `thingIndexingMode` è impostato su be`OFF`, l'indicizzazione degli oggetti è disabilitata.

Per indicizzare i dati sulla posizione memorizzati in un'ombra denominata, è necessario `thingIndexingMode` impostare su be `REGISTRY` (or`REGISTRY_AND_SHADOW`). Per ulteriori informazioni, consulta ???.

- `filter`

  Il filtro di indicizzazione fornisce selezioni aggiuntive per le ombre denominate e i dati di geolocalizzazione. Per ulteriori informazioni, consulta ???.

- `geoLocations`

  L'elenco degli obiettivi di geolocalizzazione che scegli di indicizzare. Il numero massimo predefinito di obiettivi di geolocalizzazione per l'indicizzazione è. 1 Per aumentare il limite, vedi Quote.AWS IoT Device Management

- `name`

  Il nome del campo di destinazione della geolocalizzazione. Un valore di esempio di `name` può essere il percorso dei dati sulla posizione della tua ombra:. `shadow.name.namedShadow1.reported.coordinates`

- `order`

  L'ordine del campo di destinazione della geolocalizzazione. Valori validi: `LatLon` e. `LonLat` `LatLon`significa latitudine e longitudine. `LonLat`significa longitudine e latitudine. Questo campo è facoltativo. Il valore predefinito è `LatLon`.

Geoquery di esempio

Dopo aver completato la configurazione di indicizzazione dei dati sulla posizione, esegui delle geoquery per cercare i dispositivi. Puoi anche combinare le tue geoquery con altre stringhe di query. Per ulteriori informazioni, consultare ??? e ???.

Query di esempio 1

L'esempio presuppone che i dati sulla posizione siano memorizzati in un'ombra `gps-tracker` denominata. L'output di questo comando è l'elenco dei dispositivi che si trovano entro una distanza radiale di 15,5 km dal punto centrale con coordinate (47.6204, -122.3491).

```
aws iot search-index --query-string \
"shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Query di esempio 2

Questo esempio presuppone che i dati sulla posizione siano archiviati in un'ombra classica. L'output di questo comando è l'elenco dei dispositivi che si trovano entro una distanza radiale di 15,5 km dal punto centrale con coordinate (47.6204, -122.3491).

```
aws iot search-index --query-string \
"shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Query di esempio 3

Questo esempio presuppone che i dati sulla posizione siano archiviati in un'ombra classica. L'output di questo comando è l'elenco dei dispositivi non collegati e che si trovano al di fuori della distanza radiale di 15,5 km dal punto centrale con coordinate (47.6204, -122.3491).

```
aws iot search-index --query-string \
"connectivity.connected:false AND (NOT
 shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km)"
```

Tutorial sulle nozioni di base

Questo tutorial illustra come utilizzare l'indicizzazione del parco veicoli per indicizzare i dati sulla posizione. Per semplicità, crei un oggetto per rappresentare il tuo dispositivo e memorizzi i dati sulla posizione in un'ombra denominata, aggiorni la configurazione di indicizzazione degli oggetti per l'indicizzazione della posizione ed esegui geoquery di esempio per cercare dispositivi all'interno di un confine radiale.

Questo tutorial dura circa 15 minuti.

In questo argomento:

- Prerequisiti
- Crea cose e ombre
- Aggiorna la configurazione di indicizzazione degli oggetti
- Esegui geoquery

Prerequisiti

- AWS CLIInstalla la versione più recente di.

- Acquisisci familiarità con l'indicizzazione della posizione e le geoquery, l'indicizzazione di Manage Thing e la sintassi delle query.

Crea cose e ombre

Crei un oggetto per rappresentare il tuo dispositivo e un'ombra denominata per memorizzarne i dati sulla posizione (coordinate 47.61564, -122.33584).

1. Esegui il seguente comando per creare l'oggetto che rappresenta la tua bici chiamato Bike-1. Per ulteriori informazioni su come creare un oggetto utilizzando AWS CLI, vedi create-thing from Reference. AWS CLI

   ```
   aws iot create-thing --thing-name "Bike-1" \
   --attribute-payload '{"attributes": {"model":"OEM-2302-12", "battery":"35",
    "acqDate":"06/09/23"}}'
   ```

   L'output di questo comando può essere simile al seguente:

   ```
   {
       "thingName": "Bike-1",
       "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/Bike-1",
       "thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df"
   }
   ```

2. Esegui il comando seguente per creare un'ombra con nome per memorizzare i dati sulla posizione di Bike-1 (coordinate 47.61564, -122.33584). Per ulteriori informazioni su come creare un'ombra con nome utilizzando, vedere da Reference. AWS CLIupdate-thing-shadowAWS CLI

   ```
   aws iot-data update-thing-shadow \
   --thing-name Bike-1 \
   --shadow-name Bike1-shadow \
   --cli-binary-format raw-in-base64-out \
   --payload '{"state":{"reported":{"coordinates":{"lat": 47.6153, "lon": -122.3333}}}}'
    \
   "output.txt" \
   ```

   Il comando non produce output. Per visualizzare l'ombra denominata che hai creato, puoi eseguire il comando CLI list-named-shadows-for-thing.

```
aws iot-data list-named-shadows-for-thing --thing-name Bike-1
```

L'output di questo comando può essere simile al seguente:

```
{
    "results": [
        "Bike1-shadow"
    ],
    "timestamp": 1699574309
}
```

Aggiorna la configurazione di indicizzazione degli oggetti

Per indicizzare i dati sulla posizione, è necessario aggiornare la configurazione di indicizzazione degli oggetti in modo da includere i dati sulla posizione. Poiché i dati sulla posizione sono memorizzati in un'ombra denominata, in questo tutorial, `thingIndexingMode` impostate come `REGISTRY` (come requisito minimo), `namedShadowIndexingMode` impostate `ON` come tale e aggiungete i dati sulla posizione alla configurazione. In questo esempio, è necessario aggiungere il nome dell'ombra denominata e il percorso dei dati sulla posizione dell'ombra `afilter`.

1. Eseguite il comando per aggiornare la configurazione di indicizzazione per l'indicizzazione della posizione.

```
aws iot update-indexing-configuration --cli-input-json '{
"thingIndexingConfiguration": { "thingIndexingMode": "REGISTRY",
"thingConnectivityIndexingMode": "OFF",
"deviceDefenderIndexingMode": "OFF",
"namedShadowIndexingMode": "ON",
"filter": {
    "namedShadowNames": ["Bike1-shadow"],
    "geoLocations":[{
        "name":"shadow.name.Bike1-shadow.reported.coordinates"
    }]
},
"customFields": [
{ "name":"attributes.battery",
"type":"Number"}] } }'
```

Il comando non produce output. Potrebbe essere necessario attendere qualche istante fino al completamento dell'aggiornamento. Per verificare lo stato, esegui il comando CLI describe-index. Se vedi `indexStatus` shows:`ACTIVE`, l'aggiornamento dell'indicizzazione delle cose è completo.

2. Esegui il comando per verificare la configurazione di indicizzazione. Questa fase è facoltativa.

```
aws iot get-indexing-configuration
```

L'output può essere simile al seguente:

```
{
    "thingIndexingConfiguration": {
        "thingIndexingMode": "REGISTRY",
        "thingConnectivityIndexingMode": "OFF",
        "deviceDefenderIndexingMode": "OFF",
        "namedShadowIndexingMode": "ON",
        "managedFields": [
            {
                "name": "shadow.name.*.hasDelta",
                "type": "Boolean"
            },
            {
                "name": "registry.version",
                "type": "Number"
            },
            {
                "name": "registry.thingTypeName",
                "type": "String"
            },
            {
                "name": "registry.thingGroupNames",
                "type": "String"
            },
            {
                "name": "shadow.name.*.version",
                "type": "Number"
            },
            {
                "name": "thingName",
                "type": "String"
            },
            {
```

```
                "name": "thingId",
                "type": "String"
            }
        ],
        "customFields": [
            {
                "name": "attributes.battery",
                "type": "Number"
            }
        ],
        "filter": {
            "namedShadowNames": [
                "Bike1-shadow"
            ],
            "geoLocations": [
                {
                    "name": "shadow.name.Bike1-shadow.reported.coordinates",
                    "order": "LatLon"
                }
            ]
        }
    },
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "OFF"
    }
}
```

Esegui geoquery

Ora hai aggiornato la configurazione di indicizzazione degli oggetti per includere i dati sulla posizione. Prova a creare alcune geoquery ed eseguile per vedere se riesci a ottenere i risultati di ricerca desiderati. Una geoquery deve seguire la sintassi Query. Puoi trovare alcuni utili esempi di geoquery in. ???

Nel comando di esempio seguente, si utilizza la geoquery per `shadow.name.Bike1-shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km` cercare dispositivi che si trovano entro la distanza radiale di 15,5 km dal punto centrale con coordinate (47.6204, -122.3491).

```
aws iot search-index --query-string "shadow.name.Bike1-
shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Poiché hai un dispositivo situato alle coordinate «lat»: 47.6153, «lon»: -122.3333, che rientra nella distanza di 15,5 km dal punto centrale, dovresti essere in grado di vedere questo dispositivo (Bike-1) nell'output. L'output può essere simile al seguente:


```
{
    "things": [
        {
            "thingName": "Bike-1",
            "thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df",
            "attributes": {
                "acqDate": "06/09/23",
                "battery": "35",
                "model": "OEM-2302-12"
            },
            "shadow": "{\"reported\":{\"coordinates\":{\"lat\":47.6153,\"lon
\":-122.3333}},\"metadata\":{\"reported\":{\"coordinates\":{\"lat\":{\"timestamp
\":1699572906},\"lon\":{\"timestamp\":1699572906}}}},\"hasDelta\":false,\"version\":1}"
        }
    ]
}
```


Per ulteriori informazioni, consulta ???.

Parametri del parco istanze

Le metriche della flotta sono una funzionalità dell'indicizzazione della flotta, un servizio gestito che ti consente di indicizzare, cercare e aggregare i dati dei tuoi dispositivi. AWS IoT Puoi utilizzare le metriche della flotta per monitorare lo stato aggregato dei dispositivi del tuo parco dispositivi nel CloudWatchtempo, inclusa la revisione della velocità di disconnessione dei dispositivi del parco dispositivi o delle variazioni medie del livello della batteria in un periodo specifico.

Utilizzando le metriche della flotta, puoi creare query di aggregazione i cui risultati vengono continuamente utilizzati come metriche per l'analisi delle tendenze e la creazione di allarmi. CloudWatch Per i processi di monitoraggio, è possibile specificare le query di aggregazione di diversi tipi di aggregazione (Statistiche, Cardinalità e Percentile). Puoi salvare tutte le query di aggregazione per creare parametri del parco istanze da riutilizzare in futuro.

Tutorial sulle nozioni di base

In questo tutorial viene creato un parametro del parco istanza per monitorare le temperature dei sensori e rilevare potenziali anomalie. Durante la creazione del parametro del parco istanze, si definisce una query di aggregazione che rileva il numero di sensori con temperature superiori a 80 gradi Fahrenheit. Si specifica che l'interrogazione deve essere eseguita ogni 60 secondi e i risultati dell'interrogazione vengono inviati CloudWatch, dove è possibile visualizzare il numero di sensori che presentano potenziali rischi legati alle alte temperature e impostare allarmi. Per completare questo tutorial, userai AWS CLI.

In questo tutorial, apprenderai come:

- Configurazione
- Creare parametri del parco istanze
- Visualizza le metriche in CloudWatch
- Pulizia delle risorse

Questo tutorial dura circa 15 minuti.

Prerequisiti

- Installazione della versione più recente di AWS CLI
- Familiarizza con la query di dati aggregati
- Acquisisci familiarità con l'utilizzo dei parametri di Amazon CloudWatch

Configurazione

Abilita l'indicizzazione del parco istanze per utilizzare i parametri del parco istanze. Per abilitare l'indicizzazione del parco istanze per i tuoi oggetti o gruppi di oggetti con origini dati specificate e configurazioni associate, segui le istruzioni riportate in Gestione dell'indicizzazione di oggetti e Gestione dell'indicizzazione di gruppi di oggetti.

Per configurare

1. Esegui il seguente comando per abilitare l'indicizzazione del parco istanze e specificare le origini dati da cui eseguire la ricerca.

```
aws iot update-indexing-configuration \
--thing-indexing-configuration
 "thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.temperature,type=Num
{name=attributes.rackId,type=String},
{name=attributes.stateNormal,type=Boolean}],thingConnectivityIndexingMode=STATUS" \
```

L'esempio del comando CLI precedente consente l'indicizzazione del parco istanze per supportare la ricerca dei dati del registro, i dati shadow e lo stato di connettività dell'oggetto utilizzando l'indice `AWS_Things`.

La modifica della configurazione può richiedere alcuni minuti. Verifica che l'indicizzazione del tuo parco istanze sia abilitata prima di creare i parametri del parco istanze.

Per controllare se l'indicizzazione del parco istanze è stata abilitata, esegui il seguente comando della CLI:

```
aws --region us-east-1 iot describe-index --index-name "AWS_Things"
```

Per ulteriori informazioni, consulta Abilitazione dell'indicizzazione di oggetti.

2. Esegui il seguente script bash per creare dieci oggetti e descriverli.

```
# Bash script. Type `bash` before running in other shells.

Temperatures=(70 71 72 73 74 75 47 97 98 99)
Racks=(Rack1 Rack1 Rack2 Rack2 Rack3 Rack4 Rack5 Rack6 Rack6 Rack6)
IsNormal=(true true true true true true false false false false)

for ((i=0; i < 10; i++))
do
  thing=$(aws iot create-thing --thing-name "TempSensor$i" --attribute-
payload attributes="{temperature=${Temperatures[@]:$i:1},rackId=${Racks[@]:
$i:1},stateNormal=${IsNormal[@]:$i:1}}")
  aws iot describe-thing --thing-name "TempSensor$i"
done
```

Questo script crea dieci oggetti per rappresentare dieci sensori. Ogni oggetto ha attributi di `temperature`, `rackId`, e `stateNormal` come descritto nella tabella seguente:

Attributo	Tipo di dati	Description
`temperature`	Numero	Valore della temperatura in Fahrenheit
`rackId`	Stringa	ID del server rack contenente sensori
`stateNormal`	Booleano	Se il valore della temperatura del sensore è normale o meno

L'output di questo script contiene dieci file JSON. Uno dei file JSON ha il seguente aspetto:

```
{
    "version": 1,
    "thingName": "TempSensor0",
    "defaultClientId": "TempSensor0",
    "attributes": {
        "rackId": "Rack1",
        "stateNormal": "true",
        "temperature": "70"
    },
    "thingArn": "arn:aws:iot:region:account:thing/TempSensor0",
    "thingId": "example-thing-id"
}
```

Per ulteriori informazioni, consulta Crea un oggetto.

Crea parametri del parco istanze

Per creare un parametro del parco istanze

1. Esegui il comando seguente per creare una metrica della flotta denominata. *high_temp_FM* Crei la metrica della flotta per monitorare il numero di sensori con temperature superiori a 80 gradi Fahrenheit in. CloudWatch

```
aws iot create-fleet-metric --metric-name "high_temp_FM" --query-string
 "thingName:TempSensor* AND attributes.temperature >80" --period 60 --aggregation-
field "attributes.temperature" --aggregation-type name=Statistics,values=count
```

--nome-parametro

Tipo di dati: stringa. Il parametro `--metric-name` specifica il nome parametro del parco istanze. In questo esempio, stai creando un parametro del parco istanze denominato high_temp_FM.

--query-stringa

Tipo di dati: stringa. Il parametro `--query-string` specifica la stringa di query. In questo esempio, la stringa di query significa interrogare tutte le cose con nomi che iniziano con TempSensore con temperature superiori a 80 gradi Fahrenheit. Per ulteriori informazioni, consulta Sintassi delle query.

--periodo

Tipo di dati: numero intero. Il parametro `--period` specifica il tempo necessario per recuperare i dati aggregati in secondi. In questo esempio, specifichi che il parametro del parco istanze che stai creando recupera i dati aggregati ogni 60 secondi.

--aggregazione--campo

Tipo di dati: stringa. Il parametro `--aggregation-field` specifica l'attributo da valutare. In questo esempio, l'attributo della temperatura deve essere valutato.

--tipo--aggregazione

Il parametro `--aggregation-type` specifica il riepilogo statistico da visualizzare nel parametro del parco istanze. Per i processi di monitoraggio, è possibile personalizzare le proprietà della query di aggregazione per i diversi tipi di aggregazione (Statistiche, Cardinalità e Percentile). In questo esempio, si specifica count per il tipo di aggregazione e Statistics per restituire il conteggio dei dispositivi con attributi che corrispondono alla query, in altre parole, per restituire il conteggio dei dispositivi con nomi che iniziano con TempSensore con temperature superiori a 80 gradi Fahrenheit. Per ulteriori informazioni, consulta Query di dati aggregati.

L'output di questo comando è simile al seguente:

```
{
    "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
    "metricName": "high_temp_FM"
}
```

> **Note**
>
> La visualizzazione dei punti dati può richiedere alcuni istanti. CloudWatch

Per ulteriori informazioni sulla creazione di un parametro di parco istanze, leggi Gestione dei parametri del parco istanze.

Se non riesci a creare un parametro del parco di istanze, leggi Risoluzione dei problemi dei parametri del parco istanze.

2. (Facoltativo) Esegui il comando seguente per descrivere il parametro del parco istanze denominato high_temp_FM:

```
aws iot describe-fleet-metric --metric-name "high_temp_FM"
```

L'output di questo comando è simile al seguente:

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625249775.834,
    "queryString": "*",
    "period": 60,
    "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
    "aggregationField": "registry.version",
    "version": 1,
    "aggregationType": {
        "values": [
            "count"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625249775.834,
    "metricName": "high_temp_FM"
```

```
}
```

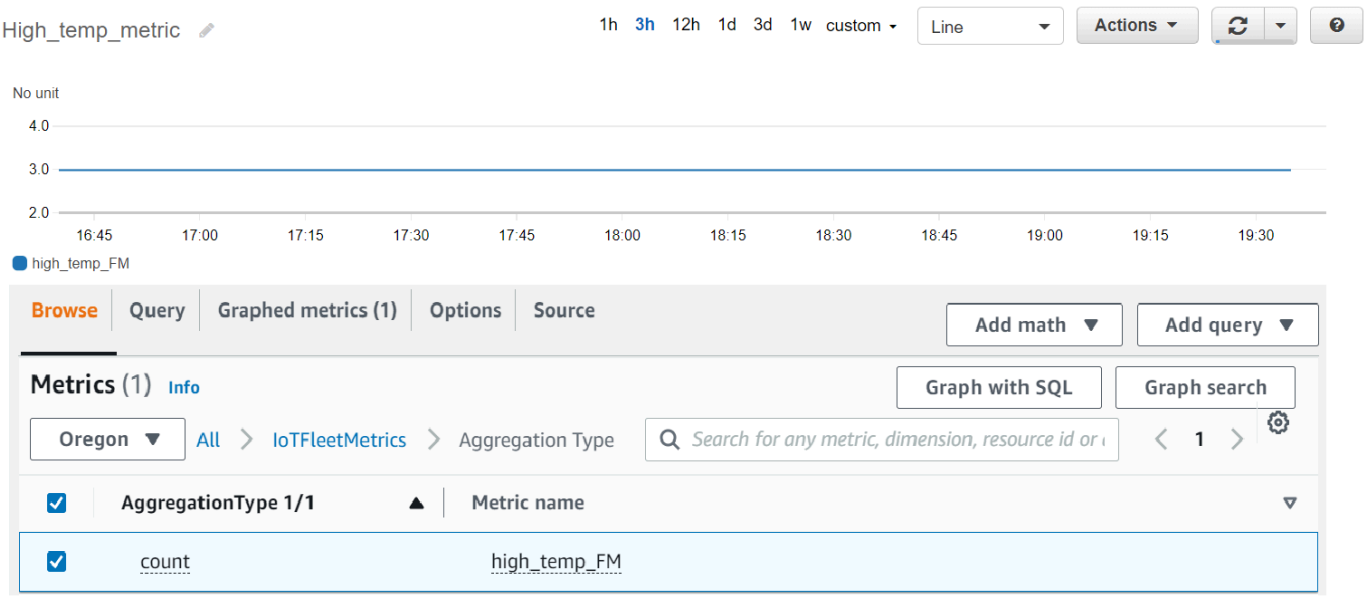
Visualizza le metriche della flotta in CloudWatch

Dopo aver creato la metrica della flotta, puoi visualizzare i dati metrici in. CloudWatch In questo tutorial, vedrai la metrica che mostra il numero di sensori con nomi che iniziano con TempSensore con temperature superiori a 80 gradi Fahrenheit.

Per visualizzare i punti dati in CloudWatch

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. CloudWatch Nel menu sul pannello di sinistra, scegli Metriche per espandere il sottomenu, quindi scegli Tutte le metriche. Si apre la pagina in cui la metà superiore mostra il grafico e la metà inferiore contiene quattro sezioni a schede.
3. La prima sezione a schede Tutte le metriche elenca tutte le metriche che puoi visualizzare in gruppi, scegli Io Metrics. TFleet Contiene tutti i parametri della tua flotta.
4. Nella sezione Tipo di aggregazione della scheda Tutti i parametri, scegli Tipo di aggregazione per visualizzare tutti parametri del parco istanze che hai creato.
5. Scegli i parametri del parco istanze per visualizzare il grafico a sinistra della sezione Tipo di aggregazione. Vedrai il valore *`count`* a sinistra del nome della tua metrica, e questo è il valore del tipo di aggregazione che hai specificato nella sezione Crea metriche sulla flotta di questo tutorial.
6. Scegli la seconda scheda denominata Rappresentazione grafica di parametri a destra della scheda Tutti i parametri per visualizzare il parametro del parco istanze che hai scelto nella fase precedente.

   Dovresti essere in grado di vedere un grafico che mostra il numero di sensori con temperature superiori a 80 gradi Fahrenheit come segue:



> **Note**
>
> L'attributo Periodo è CloudWatch predefinito su 5 minuti. È l'intervallo di tempo tra i punti dati visualizzati in. CloudWatch È possibile modificare l'impostazione Periodo in base alle tue esigenze.

7. (Facoltativo) È possibile impostare un allarme di parametro.

   1. CloudWatch Nel menu sul pannello di sinistra, scegli Allarmi per espandere il sottomenu, quindi scegli Tutti gli allarmi.
   2. Sulla pagina Allarmi, scegli Creazione di allarme nell'angolo in alto a destra. Segui le istruzioni per la Creazione di allarme nella console per creare un allarme secondo necessità. Per ulteriori informazioni, consulta Usare gli CloudWatch allarmi Amazon.

Per ulteriori informazioni, consulta Usare i CloudWatch parametri di Amazon.

Se non riesci a visualizzare i punti dati CloudWatch, leggi Risoluzione dei problemi relativi alle metriche della flotta.

Eliminazione

Per eliminare i parametri del parco istanze

Utilizza il comando CLI delete-fleet-metric per eliminare i parametri del parco istanze.

Per eliminare il parametro del parco istanze denominato high_temp_FM, esegui il seguente comando.

```
aws iot delete-fleet-metric --metric-name "high_temp_FM"
```

Per eliminare gli oggetti

Usa il comando CLI delete-thing per eliminare gli oggetti.

Per eliminare i dieci oggetti creati, esegui lo script seguente:

```
# Bash script. Type `bash` before running in other shells.

for ((i=0; i < 10; i++))
do
  thing=$(aws iot delete-thing --thing-name "TempSensor$i")
done
```

Per ripulire le metriche in CloudWatch

CloudWatch non supporta l'eliminazione delle metriche. I parametri scadono in base ai piani di conservazione. Per ulteriori informazioni, consulta Utilizzo dei CloudWatch parametri di Amazon.

Gestione dei parametri del parco istanze

Questo argomento mostra come utilizzare la AWS IoT console e AWS CLI gestire i parametri del parco veicoli.

Argomenti

- Gestione dei parametri del parco istanze (Console)
- Gestione dei parametri del parco istanze (CLI)
- Autorizzazione dell'assegnazione di tag delle risorse IoT

Gestione dei parametri del parco istanze (Console)

Le sezioni seguenti mostrano come utilizzare la AWS IoT console per gestire le metriche del parco veicoli. Assicurati di aver abilitato l'indicizzazione del parco istanze con origini dati e configurazioni associate prima di creare parametri del parco istanze.

Abilita l'indicizzazione del parco istanze

Se hai già abilitato l'indicizzazione del parco istanze, salta questa sezione.

Se non hai abilitato l'indicizzazione del parco istanze, segui queste istruzioni.

1. Apri la AWS IoT console all'indirizzo https://console.aws.amazon.com/iot/.
2. AWS IoT Nel menu, scegli Impostazioni.
3. Per visualizzare le impostazioni dettagliate, sulla pagina Impostazioni, scorri verso il basso fino alla sezione Indicizzazione del parco istanze.
4. Per aggiornare le impostazioni di indicizzazione del parco istanze, a destra della sezione Indicizzazione del parco istanza, seleziona Gestione dell'indicizzazione.
5. Sulla pagina Gestione dell'indicizzazione del parco istanze, aggiorna le impostazioni di indicizzazione del parco istanze in base alle tue esigenze.
   - Configurazione

     Per attivare l'indicizzazione degli oggetti, attiva Indicizzazione degli oggetti, quindi seleziona le origini dati da cui vuoi indicizzare.

     Per attivare l'indicizzazione del gruppo di oggetti, attiva Indicizzazione di gruppi di oggetti.
   - Campi personalizzati per l'aggregazione (facoltativo)

     I campi di ricerca personalizzati sono un elenco di coppie di nomi di campo e tipi di campo.

     Per aggiungere una coppia di campi personalizzata, scegli Aggiungi nuovo campo. Inserisci un nome di campo personalizzato come `attributes.temperature`, quindi seleziona un tipo di campo dal menu Tipo di campo. Nota che il nome di un campo personalizzato inizia con `attributes.` e verrà salvato come attributo per eseguire query di aggregazioni di oggetti.

     Per aggiornare e salvare l'impostazione, scegli Aggiorna.

Crea un parametro del parco istanze

1. Apri la AWS IoT console all'indirizzo https://console.aws.amazon.com/iot/.
2. AWS IoT Nel menu, scegli Gestisci, quindi scegli Fleet metrics.
3. Sulla pagina Parametri del parco istanza, scegli Crea un parametro del parco istanze e completa i passaggi di creazione.
4. Nel passaggio 1 Configurazione dei parametri del parco istanze

- Nella sezione Query, inserisci una stringa di query per specificare gli oggetti o i gruppi di oggetti che desideri che eseguano la ricerca aggregata. La stringa di query è costituita da un attributo e da un valore. Per Proprietà, scegli l'attributo desiderato oppure, se non viene visualizzato nell'elenco, inserisci l'attributo nel campo. Inserire il valore dopo `:`. Una stringa di query di esempio può essere `thingName:TempSensor*`. Per ogni stringa di query inserita, premi invio sulla tastiera. Se inserisci più stringhe di query, specifica la relazione selezionando e, o, e non, oppure o no tra di loro.
- In Proprietà del report, scegli Nome indice, Tipo di aggregazione, e Campo di aggregazione dalle rispettive liste. Quindi, seleziona i dati in cui desideri aggregare Seleziona dati, dove è possibile selezionare più valori di dati.
- Scegli Next (Successivo).

5. Nella fase 2 Specificazione delle proprietà del parametro del parco istanze
   - Nel campo Nome parametro del parco istanza, inserisci un nome per il parametro del parco istanze che stai creando.
   - Nel campo Descrizione - opzionale, inserisci una descrizione per il parametro del parco istanze che stai creando. Questo campo è facoltativo.
   - Nei campi Ore e Minuti, inserisci l'ora (con quale frequenza) in cui desideri che la metrica della flotta emetta i dati. CloudWatch
   - Scegli Next (Successivo).
6. Nella fase 3 Revisione e creazione
   - Revisiona le impostazioni della fase 1 e fase 2. Per modificare le impostazioni, scegli Modifica.
   - Scegli Crea parametro del parco istanze.

Dopo una creazione riuscita, il parametro del parco istanze è elencato sulla pagina Parametro del parco istanze.

Aggiorna una parametro del parco istanze

1. Sulla pagina Parametro del parco istanze, scegli il parametro del parco istanze che desideri aggiornare.
2. Sulla pagina Dettagli del parametro del parco istanze, scegli Modifica. In questo modo si aprono le fasi di creazione in cui è possibile aggiornare il parametro del parco istanze in una delle tre fasi.
3. Dopo aver completato l'aggiornamento del parametro del parco istanze, scegli Aggiorna parametro del parco istanze.

Elimina un parametro del parco istanze

1. Sulla pagina Parametro del parco istanze, scegli il parametro del parco istanze che desideri eliminare.
2. Nella pagina successiva che mostra i dettagli del parametro del parco istanze, scegli Elimina.
3. Nella finestra di dialogo, inserisci il nome del parametro del parco istanze per confermare l'eliminazione.
4. Scegli Elimina. Questa fase elimina definitivamente il parametro del parco istanze.

Gestione dei parametri del parco istanze (CLI)

Le sezioni seguenti mostrano come utilizzare le metriche della flotta AWS CLI per gestire. Assicurati di aver abilitato l'indicizzazione del parco istanze con origini dati e configurazioni associate prima di creare parametri del parco istanze. Per abilitare l'indicizzazione del parco istanze per oggetti o gruppi di oggetti, segui le istruzioni in Gestione dell'indicizzazione degli oggetti o Gestione dell'indicizzazione di gruppi di oggetti.

Crea un parametro del parco istanze

È possibile utilizzare il comando create-fleet-metric CLI per creare una metrica della flotta.

```
aws iot create-fleet-metric --metric-name "YourFleetMetricName" --query-string
 "*" --period 60 --aggregation-field "registry.version" --aggregation-type
 name=Statistics,values=sum
```

L'output di questo comando contiene il nome e Amazon Resource Name (ARN) del parametro del parco istanze. L'output sarà simile al seguente:

```
{
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "metricName": "YourFleetMetricName"
}
```

Elenco parametri del parco istanze

Puoi utilizzare il comando list-fleet-metric CLI per elencare tutte le metriche della flotta nel tuo account.

```
aws iot list-fleet-metrics
```

L'output di questo comando contiene tutti i parametri del parco istanze. L'output sarà simile al seguente:

```
{
    "fleetMetrics": [
        {
            "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/
YourFleetMetric1",
            "metricName": "YourFleetMetric1"
        },
        {
            "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/
YourFleetMetric2",
            "metricName": "YourFleetMetric2"
        }
    ]
}
```

Descrivi un parametro del parco istanze

È possibile utilizzare il comando describe-fleet-metric CLI per visualizzare informazioni più dettagliate su una metrica della flotta.

```
aws iot describe-fleet-metric --metric-name "YourFleetMetricName"
```

L'output del comando contiene le informazioni dettagliate sul parametro del parco istanze specificato. L'output sarà simile al seguente:

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625790642.355,
    "queryString": "*",
    "period": 60,
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "aggregationField": "registry.version",
    "version": 1,
    "aggregationType": {
        "values": [
            "sum"
        ],
        "name": "Statistics"
    },
```

```
    "indexName": "AWS_Things",
    "creationDate": 1625790642.355,
    "metricName": "YourFleetMetricName"
}
```

Aggiorna una parametro del parco istanze

È possibile utilizzare il comando update-fleet-metric CLI per aggiornare una metrica del parco veicoli.

```
aws iot update-fleet-metric --metric-name "YourFleetMetricName" --query-string
 "*" --period 120 --aggregation-field "registry.version" --aggregation-type
 name=Statistics,values=sum,count --index-name AWS_Things
```

Il comando update-fleet-metric non produce output. Puoi usare il comando describe-fleet-metric CLI per vedere il risultato.

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625792300.881,
    "queryString": "*",
    "period": 120,
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "aggregationField": "registry.version",
    "version": 2,
    "aggregationType": {
        "values": [
            "sum",
            "count"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625792300.881,
    "metricName": "YourFleetMetricName"
}
```

Elimina un parametro del parco istanze

Utilizza il comando delete-fleet-metric CLI per eliminare una metrica della flotta.

```
aws iot delete-fleet-metric --metric-name "YourFleetMetricName"
```

Questo comando non produce alcun output se l'eliminazione va a buon fine oppure se specifichi un parametro del parco istanze che non esiste.

Per ulteriori informazioni, consulta Risoluzione dei problemi dei parametri del parco istanze.

Autorizzazione dell'assegnazione di tag delle risorse IoT

Per un migliore controllo sulle metriche della flotta che puoi creare, modificare o utilizzare, puoi allegare tag alle metriche della flotta.

Per etichettare le metriche della flotta create utilizzando Console di gestione AWS or AWS CLI, devi includere l'`iot:TagResource`azione nella tua policy IAM per concedere le autorizzazioni all'utente. Se la tua policy IAM non lo include`iot:TagResource`, qualsiasi azione volta a creare una metrica della flotta con un tag restituirà un errore. `AccessDeniedException`

Per informazioni generali sull'etichettatura delle risorse, consulta Tagging your resources. AWS IoT

Esempio di policy IAM

Fai riferimento al seguente esempio di policy IAM che concede le autorizzazioni per l'etichettatura quando crei una metrica della flotta:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Action": [
    "iot:TagResource"
   ],
   "Effect": "Allow",
   "Resource": [
    "arn:aws:iot:*:*:fleetmetric/*"
   ]
  },
  {
   "Action": [
    "iot:CreateFleetMetric"
   ],
   "Effect": "Allow",
   "Resource": [
    "arn:aws:iot:*:*:index/*",
    "arn:aws:iot:*:*:fleetmetric/*"
```

```
    ]
   }
  ]
}
```

Per ulteriori informazioni, consulta Operazioni, risorse e chiavi di condizione per AWS IoT.

Distribuzione di file basata su MQTT

Un'opzione che potete utilizzare per gestire i file e trasferirli sui dispositivi del vostro parco AWS IoT dispositivi è la consegna dei file basata su MQTT. Con questa funzionalità nel AWS Cloud puoi creare uno stream che contiene più file, aggiornare i dati dello stream (l'elenco e le descrizioni dei file), ottenere i dati dello stream e altro ancora. AWS IoT La distribuzione di file basata su MQTT può trasferire dati in piccoli blocchi ai dispositivi IoT, utilizzando il protocollo MQTT con supporto per messaggi di richiesta e risposta in JSON o CBOR.

Per ulteriori informazioni sui modi di trasferire dati da e verso dispositivi IoT utilizzando AWS IoT, vedereConnect i dispositivi a AWS IoT.

Argomenti

- Che cos'è un flusso?
- Gestione di uno stream nel cloud AWS
- Utilizzo della distribuzione di file AWS IoT basata su MQTT nei dispositivi
- Un esempio di caso d'uso in FreeRTOS OTA

Che cos'è un flusso?

In AWS IoT, uno stream è una risorsa indirizzabile pubblicamente che è un'astrazione per un elenco di file che possono essere trasferiti su un dispositivo IoT. Generalmente un flusso contiene le seguenti informazioni:

- Un Amazon Resource Name (ARN) che identifica in modo univoco un flusso in un dato momento. Questo ARN ha il modello `arn:`*`partition`*`:iot:`*`region`*`:`*`account-ID`*`:stream/`*`stream ID`*.
- Un ID stream che identifica lo stream e viene utilizzato (e di solito richiesto) nei comandi AWS Command Line Interface (AWS CLI) o SDK.
- Una descrizione del flusso che fornisce una descrizione della risorsa di flusso.
- Una versione di flusso che identifica una versione particolare del flusso. Poiché i dati di flusso possono essere modificati immediatamente prima che i dispositivi avviino il trasferimento dei dati, la versione del flusso può essere utilizzata dai dispositivi per applicare un controllo di coerenza.
- Un elenco di file che possono essere trasferiti ai dispositivi. Per ogni file dell'elenco, il flusso registra un ID di file, la dimensione del file e le informazioni sull'indirizzo del file, che consiste, ad esempio, nel nome del bucket Amazon S3, nella chiave dell'oggetto e nella versione dell'oggetto.

- Un ruolo AWS Identity and Access Management (IAM) che concede alla distribuzione di file AWS IoT basata su MQTT l'autorizzazione a leggere i file di flusso archiviati nell'archiviazione dei dati.

AWS IoT La distribuzione di file basata su MQTT offre le seguenti funzionalità in modo che i dispositivi possano trasferire dati dal cloud: AWS

- Trasferimento dei dati tramite il protocollo MQTT.
- Supporto per i formati JSON o CBOR.
- La capacità di descrivere un flusso (API `DescribeStream`) per ottenere un elenco di file di flusso, la versione del flusso e le informazioni correlate.
- La possibilità di inviare dati in piccoli blocchi (API `GetStream`) in modo che i dispositivi con vincoli hardware possano ricevere i blocchi.
- Supporto per una dimensione di blocco dinamica per richiesta, al fine di supportare dispositivi con capacità di memoria diverse.
- Ottimizzazione per le richieste di flusso simultaneo quando più dispositivi richiedono blocchi di dati dallo stesso file di flusso.
- Amazon S3 come archiviazione dati per i file di flusso.
- Support per la pubblicazione di registri di trasferimento dati dalla consegna di file AWS IoT basata su MQTT a. CloudWatch

Per le quote di distribuzione dei file basate su MQTT, consultare AWS IoT Core Service Quotas nella Riferimenti generali di AWS.

Gestione di uno stream nel cloud AWS

AWS IoT fornisce AWS SDK e AWS CLI comandi che puoi utilizzare per gestire uno stream nel AWS cloud. È possibile utilizzare questi comandi per effettuare le seguenti operazioni:

- Creare un flusso. CLI / SDK
- Descrivere un flusso per ottenere le informazioni. CLI / SDK
- Elenca gli stream nel tuo. Account AWSCLI / SDK
- Aggiorna l'elenco dei file o la descrizione del flusso in un flusso. CLI / SDK
- Eliminare un flusso. CLI / SDK

> **Note**
>
> Al momento, i flussi non sono visibili nella Console di gestione AWS. È necessario utilizzare il AWS CLI o AWS SDK per gestire uno stream in. AWS IoT Inoltre, l'SDK per Embedded C è l'unico SDK che supporta i trasferimenti di file basati su MQTT.

Prima di utilizzare la distribuzione di file AWS IoT basata su MQTT dai dispositivi, è necessario assicurarsi che i dispositivi soddisfino le seguenti condizioni, come illustrato nelle sezioni successive:

- Una politica che riflette le autorizzazioni corrette richieste per la trasmissione di dati tramite MQTT.
- Il dispositivo può connettersi al Device Gateway. AWS IoT
- Una dichiarazione politica in cui si afferma che è possibile etichettare le risorse. Se `CreateStream` viene chiamato con tag, allora `iot:TagResource` è obbligatorio.

Prima di utilizzare la distribuzione di file AWS IoT basata su MQTT dai dispositivi, è necessario seguire i passaggi nelle sezioni successive per assicurarsi che i dispositivi siano autorizzati correttamente e possano connettersi al AWS IoT Device Gateway.

Concedere le autorizzazioni ai tuoi dispositivi

È possibile procedere come descritto in Creazione di una policy AWS IoT per creare una policy del dispositivo o utilizzare una policy del dispositivo esistente. Collegare la policy ai certificati associati ai dispositivi e aggiungere le seguenti autorizzazioni alla policy del dispositivo.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iot:Receive",
                "iot:Publish"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
${iot:Connection.Thing.ThingName}/streams/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/streams/*"
            ]
        }
    ]
}
```

Connect i tuoi dispositivi a AWS IoT

È necessario connettersi ai dispositivi che utilizzano la consegna di file AWS IoT basata su MQTT. AWS IoT AWS IoT La distribuzione di file basata su MQTT si integra con AWS IoT il AWS cloud, quindi i dispositivi devono connettersi direttamente all'endpoint del Data Plane. AWS IoT

> **Note**
>
> L'endpoint del piano AWS IoT dati è specifico della regione and. Account AWS È necessario utilizzare l'endpoint per la Account AWS e la regione in cui sono registrati i dispositivi. AWS IoT

Per ulteriori informazioni, consulta Connect a AWS IoT Core.

TagResource Utilizzo

L'azione `CreateStream` API crea un flusso per la distribuzione di uno o più file di grandi dimensioni in blocchi su MQTT.

Una chiamata `CreateStream` API riuscita richiede le seguenti autorizzazioni:

- `iot:CreateStream`
- `iot:TagResource`(se `CreateStream` è con tag)

La politica che supporta queste due autorizzazioni è mostrata di seguito:

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": [
            "iot:CreateStream",
            "iot:TagResource"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:stream/streamId"
    }
}
```

L'azione della dichiarazione `iot:TagResource` politica è necessaria per garantire che un utente non possa creare o aggiornare un tag su una risorsa senza le autorizzazioni appropriate. Senza l'azione specifica della dichiarazione politica di`iot:TagResource`, la chiamata `CreateStream` API restituirà un `AccessDeniedException` se la richiesta include tag.

Per ulteriori informazioni, consulta i seguenti link:

- CreateStream
- TagResource
- Tag

Utilizzo della distribuzione di file AWS IoT basata su MQTT nei dispositivi

Per avviare il processo di trasferimento dei dati, un dispositivo deve ricevere un set di dati iniziale, che includa almeno un ID di flusso. È possibile utilizzare un AWS IoT Lavori per pianificare le attività di trasferimento dei dati per i dispositivi includendo il set di dati iniziale nel documento del processo.

Quando un dispositivo riceve il set di dati iniziale, dovrebbe avviare l'interazione con la consegna di file basata su AWS IoT MQTT. Per scambiare dati con la distribuzione di file AWS IoT basata su MQTT, un dispositivo deve:

- Utilizzare il protocollo MQTT per effettuare la sottoscrizione a Argomenti di distribuzione dei file basati su MQTT.
- Invia richieste e attendi di ricevere le risposte utilizzando i messaggi MQTT.

Facoltativamente, è possibile includere un ID del file di flusso e una versione di flusso nel set di dati iniziale. L'invio di un ID del file di flusso a un dispositivo può semplificare la programmazione del firmware/software del dispositivo, poiché elimina la necessità di creare una richiesta `DescribeStream` dal dispositivo per ottenere questo ID. Il dispositivo può specificare la versione del flusso in una richiesta `GetStream` per applicare un controllo di coerenza nel caso in cui il flusso sia stato aggiornato in modo imprevisto.

Utilizzato DescribeStream per ottenere dati in streaming

AWS IoT La distribuzione di file basata su MQTT fornisce l'`DescribeStream`API per inviare dati di streaming a un dispositivo. I dati del flusso restituiti da questa API includono l'ID del flusso, la versione del flusso, la descrizione del flusso e un elenco di file di flusso, ognuno dei quali ha un ID di file e la dimensione del file in byte. Con queste informazioni, un dispositivo può selezionare file arbitrari per avviare il processo di trasferimento dei dati.

> **Note**
>
> Non è necessario utilizzare l'`DescribeStream`API se il dispositivo riceve tutti i file di stream richiesti IDs nel set di dati iniziale.

Attenersi ai seguenti passaggi per effettuare una richiesta `DescribeStream`.

1. Iscriversi al filtro argomento `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/description/json` “accettato”.
2. Iscriversi al filtro argomento `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/rejected/json` “rifiutato”.
3. Pubblicare una richiesta `DescribeStream` inviando un messaggio a `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/describe/json`.

4. Se la richiesta è stata accettata, il dispositivo riceve una risposta `DescribeStream` sul filtro argomento “accettato”.
5. Se la richiesta è stata rifiutata, il dispositivo riceve la risposta di errore nel filtro argomento “rifiutato”.

> **Note**
>
> Se si sostituisce `json` con `cbor` negli argomenti e nei filtri degli argomenti mostrati, il dispositivo riceve messaggi nel formato CBOR, che è più compatto di JSON.

DescribeStream richiesta

Una tipica richiesta `DescribeStream` in JSON è simile a quella nell'esempio seguente.

```
{
    "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039"
}
```

- (Facoltativo) “c” è il campo token client.

  Il token client non può superare i 64 byte. Un token client di dimensioni superiori a 64 byte causerà una risposta di errore e un messaggio di errore `InvalidRequest`.

DescribeStream risposta

Una risposta `DescribeStream` in JSON è simile a quella nell'esempio seguente.

```
{
    "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039",
    "s": 1,
    "d": "This is the description of stream ABC.",
    "r": [
        {
            "f": 0,
            "z": 131072
        },
        {
```

```
            "f": 1,
            "z": 51200
        }
    ]
}
```

- "c" è il campo token client. Questo viene restituito se è stato dato nella richiesta `DescribeStream`. Utilizza il token client per associare la risposta alla sua richiesta.
- "s" è la versione del flusso come numero intero. È possibile utilizzare questa versione per eseguire un controllo di coerenza con le richieste `GetStream`.
- "r" contiene un elenco dei file nel flusso.
  - "f" è l'ID del file di flusso come numero intero.
  - "z" è la dimensione del file di flusso in numero di byte.
- "d" contiene la descrizione del flusso.

Ottieni blocchi di dati da un file di flusso

Puoi utilizzare l'API `GetStream` in modo che un dispositivo possa ricevere file di flusso in blocchi di dati di piccole dimensioni, in modo che possa essere utilizzato da quei dispositivi che hanno vincoli per l'elaborazione di blocchi di grandi dimensioni. Per ricevere un intero file di dati, potrebbe essere necessario inviare o ricevere più richieste e risposte fino a quando tutti i blocchi di dati non vengono ricevuti ed elaborati.

GetStream richiesta

Attenersi ai seguenti passaggi per effettuare una richiesta `GetStream`.

1. Iscriversi al filtro argomento `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/data/json` "accettato".
2. Iscriversi al filtro argomento `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/rejected/json` "rifiutato".
3. Pubblicare una richiesta `GetStream` all'argomento `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/get/json`.
4. Se la richiesta è stata accettata, il dispositivo riceverà una o più risposte `GetStream` sul filtro argomento "accettato". Ogni messaggio di risposta contiene informazioni di base e un payload di dati per un singolo blocco.

5. Ripetere i passaggi 3 e 4 per ricevere tutti i blocchi di dati. È necessario ripetere questi passaggi se la quantità di dati richiesti è superiore a 128 KB. È necessario programmare il dispositivo per utilizzare più richieste `GetStream` per ricevere tutti i dati richiesti.
6. Se la richiesta è stata rifiutata, il dispositivo riceverà la risposta di errore nel filtro argomento "rifiutato".

> **Note**
>
> - Se sostituisci "json" con "cbor" negli argomenti e nei filtri degli argomenti mostrati, il tuo dispositivo riceverà messaggi nel formato CBOR, che è più compatto di JSON.
> - AWS IoT La consegna di file basata su MQTT limita la dimensione di un blocco a 128 KB. Se si effettua una richiesta per un blocco superiore a 128 KB, la richiesta non verrà eseguita correttamente.
> - È possibile effettuare una richiesta per più blocchi la cui dimensione totale è superiore a 128 KB (ad esempio, se si effettua una richiesta di 5 blocchi da 32 KB ciascuno per un totale di 160 KB di dati). In questo caso, la richiesta non ha esito negativo, ma il dispositivo deve effettuare più richieste per ricevere tutti i dati richiesti. Il servizio invierà blocchi aggiuntivi man mano che il dispositivo effettua richieste aggiuntive. Si consiglia di continuare con una nuova richiesta solo dopo che la risposta precedente è stata correttamente ricevuta ed elaborata.
> - Indipendentemente dalla dimensione totale dei dati richiesti, è consigliabile programmare il dispositivo per avviare i tentativi quando i blocchi non vengono ricevuti o non vengono ricevuti correttamente.

Una tipica richiesta `GetStream` in JSON è simile a quella nell'esempio seguente.

```
{
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
    "s": 1,
    "f": 0,
    "l": 4096,
    "o": 2,
    "n": 100,
    "b": "..."
}
```

- [facoltativo] “c” è il campo token client.

  Il token client non può superare i 64 byte. Un token client di dimensioni superiori a 64 byte causerà una risposta di errore e un messaggio di errore `InvalidRequest`.

- [facoltativo] “s” è il campo della versione di flusso (un numero intero).

  La distribuzione dei file basata su MQTT applica un controllo di coerenza basato su questa versione richiesta e sulla versione più recente del flusso nel cloud. Se la versione del flusso inviata da un dispositivo in una richiesta `GetStream` non corrisponde alla versione più recente del flusso nel cloud, il servizio invia una risposta di errore e un messaggio di errore `VersionMismatch`. In genere, un dispositivo riceve la versione di flusso prevista (più recente) nel set di dati iniziale o nella risposta a `DescribeStream`.

- "f” è l'ID del file di flusso (un numero intero compreso tra 0 e 255).

  L'ID del file di stream è necessario quando crei o aggiorni uno stream utilizzando AWS CLI o SDK. Se un dispositivo richiede un file di flusso con un ID che non esiste, il servizio invia una risposta di errore e un messaggio di errore `ResourceNotFound`.

- "l” è la dimensione del blocco dati in byte (un numero intero compreso tra 256 e 131.072).

  Fare riferimento a Crea una bitmap per una richiesta GetStream per istruzioni su come utilizzare i campi bitmap per specificare quale parte del file di flusso verrà restituita nella risposta `GetStream`. Se un dispositivo specifica una dimensione di blocco fuori intervallo, il servizio invia una risposta di errore e un messaggio di errore `BlockSizeOutOfBounds`.

- [facoltativo] “o” è l'offset del blocco nel file di flusso (un numero intero compreso tra 0 e 98.304).

  Fare riferimento a Crea una bitmap per una richiesta GetStream per istruzioni su come utilizzare i campi bitmap per specificare quale parte del file di flusso verrà restituita nella risposta `GetStream`. Il valore massimo di 98.304 è basato su un limite di dimensioni del file di flusso di 24 MB e 256 byte per la dimensione minima del blocco. Se il valore non viene specificato, viene usato il valore predefinito 0.

- [facoltativo] “n” è il numero di blocchi richiesti (un numero intero compreso tra 0 e 98.304).

  Il campo “n” specifica (1) il numero di blocchi richiesti, o (2) quando viene utilizzato il campo bitmap (“b”), un limite al numero di blocchi che verranno restituiti dalla richiesta bitmap. Questo secondo utilizzo è facoltativo. Se non è definito, il valore predefinito è 131072/. *`DataBlockSize`*

- [facoltativo] “b” è una bitmap che rappresenta i blocchi richiesti.

Utilizzando una bitmap, il dispositivo può richiedere blocchi non consecutivi, il che rende più comoda la gestione dei tentativi in seguito a un errore. Fare riferimento a Crea una bitmap per una richiesta GetStream per istruzioni su come utilizzare i campi bitmap per specificare quale parte del file di flusso verrà restituita nella risposta `GetStream`. Per questo campo, converti la bitmap in una stringa che rappresenta il valore della bitmap in notazione esadecimale. La bitmap deve essere inferiore a 12.288 byte.

> **Important**
>
> Una delle opzioni "n" o "b" deve essere specificata. Se nessuna delle due opzioni è specificata, la richiesta `GetStream` potrebbe non essere valida quando la dimensione del file è inferiore a 131072 byte (128 KB).

GetStream risposta

Una risposta `GetStream` in JSON assomiglia a questo esempio per ogni blocco di dati richiesto.

```
{
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
    "f": 0,
    "l": 4096,
    "i": 2,
    "p": "..."
}
```

- “`c`” è il campo token client. Questo viene restituito se è stato dato nella richiesta `GetStream`. Utilizza il token client per associare la risposta alla sua richiesta.
- “`f`” è l'ID del file di flusso a cui appartiene il payload del blocco di dati corrente.
- “`l`” è la dimensione del payload del blocco dati in byte.
- “`i`” è l'ID del blocco di dati contenuto nel payload. I blocchi di dati sono numerati a partire da 0.
- “p” contiene il payload del blocco dati. Questo campo è una stringa, che rappresenta il valore del blocco di dati in codifica Base64.

Crea una bitmap per una richiesta GetStream

È possibile utilizzare il campo bitmap (b) in una richiesta `GetStream` per ottenere blocchi non consecutivi da un file di flusso. Ciò aiuta i dispositivi con capacità RAM limitata a gestire i problemi di distribuzione della rete. Un dispositivo può richiedere solo i blocchi che non sono stati ricevuti o non sono stati ricevuti correttamente. La bitmap determina quali blocchi del file di flusso verranno restituiti. Per ogni bit impostato su 1 nella bitmap, verrà restituito un blocco corrispondente del file di flusso.

Di seguito è riportato un esempio di come specificare una bitmap e i relativi campi di supporto in una richiesta `GetStream`. Ad esempio, desideri ricevere un file di flusso in blocchi di 256 byte (la dimensione del blocco). Pensa a ogni blocco di 256 byte come a un numero che specifica la sua posizione nel file, a partire da 0. Quindi il blocco 0 è il primo blocco di 256 byte nel file, il blocco 1 è il secondo e così via. Desideri richiedere i blocchi 20, 21, 24 e 43 dal file.

Offset di blocco

Poiché il primo blocco è il numero 20, specifica l'offset (campo o) come 20 per risparmiare spazio nella bitmap.

Numero di blocchi

Per garantire che il dispositivo non riceva più blocchi di quelli in grado di gestire con risorse di memoria limitate, è possibile specificare il numero massimo di blocchi da restituire in ciascun messaggio inviato dalla distribuzione di file basata su MQTT. Nota che questo valore viene ignorato se la bitmap stessa specifica meno di questo numero di blocchi o se la dimensione totale dei messaggi di risposta inviati dalla distribuzione di file basata su MQTT supera il limite del servizio di 128 KB per la richiesta `GetStream`.

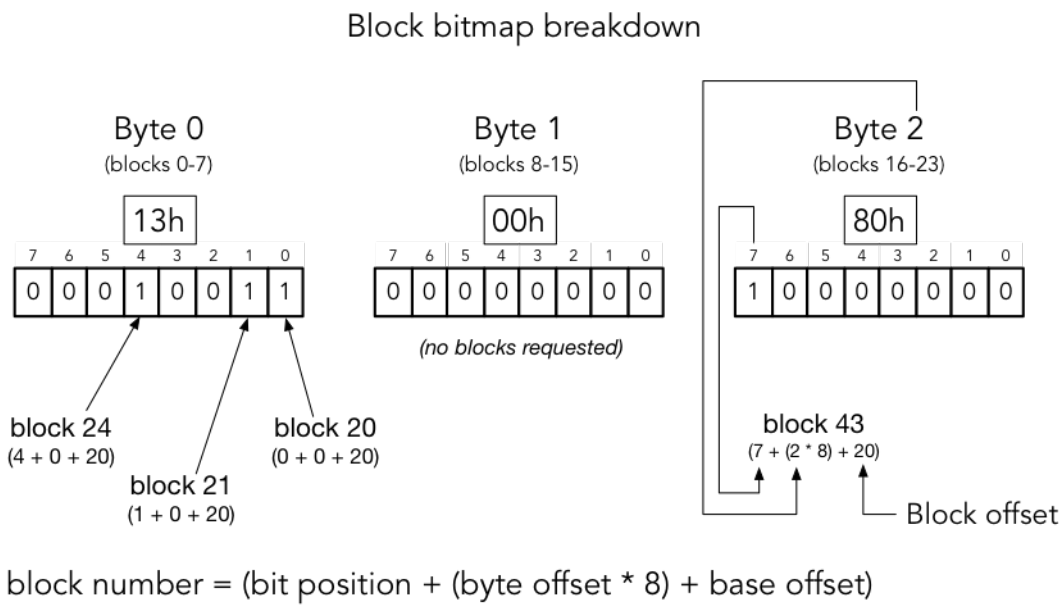
Bitmap di blocco

La bitmap stessa è una matrice di byte non firmati espressi in notazione esadecimale e inclusi nella richiesta `GetStream` come rappresentazione della stringa del numero. Ma per costruire questa stringa, iniziamo pensando alla bitmap come una lunga sequenza di bit (un numero binario). Se un bit in questa sequenza è impostato su 1, il blocco corrispondente dal file di flusso verrà inviato nuovamente al dispositivo. Per questo esempio, vogliamo ricevere i blocchi 20, 21, 24 e 43, quindi dobbiamo impostare i bit 20, 21, 24 e 43 nella nostra bitmap. Possiamo usare l'offset del blocco per risparmiare spazio, quindi dopo aver sottratto l'offset da ogni numero di blocco, vogliamo impostare i bit 0, 1, 4 e 23, come nell'esempio seguente.

```
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

Prendendo un byte (8 bit) alla volta, questo è convenzionalmente scritto come: “0b00010011”, “0b00000000” e “0b10000000”. Il bit 0 viene visualizzato nella nostra rappresentazione binaria alla fine del primo byte e il bit 23 all'inizio dell'ultimo. Questo può creare confusione a meno che non si conoscano le convenzioni. Il primo byte contiene i bit 7-0 (in questo ordine), il secondo byte contiene i bit 15-8, il terzo byte contiene i bit 23-16 e così via. Nella notazione esadecimale, questo si converte in “0x130080”.

> **Tip**
>
> È possibile convertire il binario standard in notazione esadecimale. Prendi quattro cifre binarie alla volta e convertile nel loro equivalente esadecimale. Ad esempio, “0001” diventa “1”, “0011” diventa “3” e così via.



Mettendo tutto questo insieme, il JSON per la nostra richiesta `GetStream` avrà il seguente aspetto.

```
{
    "c" : "1",
    "s" : 1,
    "l" : 256,
    "f" : 1,
    "o" : 20,
    "n" : 32,
```

```
    "b" : "130080"
}
```

- "c" è il campo token client.
- "s" è la versione del flusso prevista.
- "l" è la dimensione del payload del blocco dati in byte.
- "f" è l'ID dell'indice del file di origine.
- "o" è l'offset del blocco.
- "n" è il numero di blocchi.
- "b" è la bitmap BlockId mancante a partire dall'offset. Questo valore deve essere codificato base64.

Gestione degli errori derivanti dalla consegna di file basata su MQTT AWS IoT

Una risposta di errore che viene inviata a un dispositivo per entrambi `DescribeStream` e `GetStream` APIs contiene un token client, un codice di errore e un messaggio di errore. Una tipica risposta di errore è simile a quella nell'esempio seguente.

```
{
    "o": "BlockSizeOutOfBounds",
    "m": "The block size is out of bounds",
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380"
}
```

- "o" è il codice che indica il motivo dell'errore. Consulta i codici di errore più avanti in questa sezione per maggiori dettagli.
- "m" è il messaggio di errore che contiene i dettagli dell'errore.
- "c" è il campo token client. Questo può essere restituito se è stato dato nella richiesta `DescribeStream`. Utilizza il token client per associare la risposta alla sua richiesta.

  Il campo token client non è sempre incluso in una risposta di errore. Quando il token client fornito nella richiesta non è valido o non è corretto, non viene restituito nella risposta di errore.

> **Note**
>
> Per compatibilità con le versioni precedenti, i campi nella risposta agli errori potrebbero essere in forma non abbreviata. Ad esempio, il codice di errore potrebbe essere designato da campi “codice” o “o” e il campo token client può essere designato dai campi “clientToken” o “c”. È consigliabile utilizzare il modulo abbreviazione mostrato sopra.

InvalidTopic

L'argomento MQTT del messaggio di flusso non è valido.

InvalidJson

La richiesta del flusso non è un documento JSON valido.

InvalidCbor

La richiesta del flusso non è un documento CBOR valido.

InvalidRequest

La richiesta è generalmente identificata come non corretta. Per ulteriori informazioni, vedere il messaggio di errore.

Non autorizzato

La richiesta non è autorizzata ad accedere ai file di dati di flusso nel supporto di archiviazione, come Amazon S3. Per ulteriori informazioni, vedere il messaggio di errore.

BlockSizeOutOfBounds

La dimensione del blocco è fuori intervallo. Fare riferimento alla sezione “Distribuzione di file basata su MQTT” in Service Quotas di AWS IoT Core.

OffsetOutOfBounds

L'offset è fuori intervallo. Fai riferimento alla sezione “Distribuzione di file basata su MQTT” in Service Quotas di AWS IoT Core.

BlockCountLimitExceeded

Il numero dei blocchi di richiesta è fuori intervallo. Fai riferimento alla sezione “Distribuzione di file basata su MQTT” in Service Quotas di AWS IoT Core.

BlockBitmapLimitExceeded

La dimensione della bitmap richiesta è fuori intervallo. Fai riferimento alla sezione “Distribuzione di file basata su MQTT” in Service Quotas di AWS IoT Core.

ResourceNotFound

Il flusso, i file, le versioni del file o i blocchi richiesti non sono stati trovati. Consulta il messaggio di errore per ulteriori dettagli.

VersionMismatch

La versione del flusso nella richiesta non corrisponde alla versione del flusso nella funzione della distribuzione dei file basata su MQTT. Ciò indica che i dati del flusso sono stati modificati da quando la versione del flusso è stata inizialmente ricevuta dal dispositivo.

ETagMancata corrispondenza

L'S3 ETag nello stream non corrisponde alla versione ETag dell'oggetto S3 più recente.

InternalError

Si è verificato un errore interno nella distribuzione dei file basata su MQTT.

Un esempio di caso d'uso in FreeRTOS OTA

L'agente FreeRTOS OTA over-the-air () AWS IoT utilizza la distribuzione di file basata su MQTT per trasferire le immagini del firmware FreerTOS sui dispositivi FreerTOS. Per inviare il set di dati iniziale a un dispositivo, utilizza il servizio AWS IoT Job per pianificare un processo di aggiornamento OTA sui dispositivi FreerTOS.

Per un'implementazione di riferimento di un client della distribuzione di file basata su MQTT, consulta Codici dell'agente FreeRTOS OTA nella documentazione di FreeRTOS.

Device Advisor

Device Advisor è una funzionalità di test completamente gestita e basata sul cloud per la convalida dei dispositivi IoT durante lo sviluppo del software dei dispositivi. Device Advisor fornisce test predefiniti che è possibile utilizzare per convalidare i dispositivi IoT per una connettività affidabile e sicura prima di distribuirli in produzione. AWS IoT Core I test predefiniti di Device Advisor consentono di convalidare il software del dispositivo in base alle best practice per l'utilizzo di TLS, MQTT, Device Shadow, e Processi IoT. È inoltre possibile scaricare i rapporti di qualificazione firmati da inviare al Partner Network AWS per ottenere il tuo dispositivo qualificato per il Catalogo dei dispositivi dei partner dei servizi AWS senza la necessità di inviare il dispositivo e attendere che venga testato.

> **Note**
>
> Device Advisor è supportato nelle regioni us-east-1, us-west-2, ap-northeast-1 e eu-west-1.
> Device Advisor supporta dispositivi e client che utilizzano i protocolli MQTT e MQTT over WebSocket Secure (WSS) per pubblicare e sottoscrivere messaggi. Tutti i protocolli supportano e. IPv4 IPv6
> Device Advisor supporta i certificati server RSA.

Qualsiasi dispositivo progettato per connettersi AWS IoT Core può sfruttare Device Advisor. Puoi accedere a Device Advisor dalla AWS IoT console o utilizzando l'SDK AWS CLI o. Quando sei pronto per testare il tuo dispositivo, registralo AWS IoT Core e configura il software del dispositivo con l'endpoint Device Advisor. Quindi scegli i test precompilati, configurali, esegui i test sul tuo dispositivo e ottieni i risultati del test insieme a registri dettagliati o a un rapporto di qualificazione.

Device Advisor è un endpoint di test nel AWS cloud. È possibile testare i dispositivi configurandoli per connettersi all'endpoint di test fornito da Device Advisor. Dopo aver configurato un dispositivo per la connessione all'endpoint di test, puoi visitare la console di Device Advisor o utilizzare l' AWS SDK per scegliere i test da eseguire sui tuoi dispositivi. Device Advisor gestisce quindi l'intero ciclo di vita di un test, incluso il provisioning delle risorse, la programmazione del processo di test, la gestione della macchina a stato, la registrazione del comportamento del dispositivo, la registrazione dei risultati e la fornitura dei risultati finali sotto forma di report di test.

Protocolli TLS

Il protocollo Transport Layer Security (TLS) viene utilizzato per crittografare i dati riservati su reti non sicure, ad esempio Internet. Il protocollo TLS è il successore del protocollo Secure Sockets Layer (SSL).

Device Advisor supporta i seguenti protocolli TLS:

- TLS 1.3 (consigliato)
- TLS 1.2

Protocolli, mappature delle porte e autenticazione

Il protocollo di comunicazione del dispositivo viene utilizzato da un dispositivo o da un client per connettersi al broker di messaggi utilizzando un endpoint del dispositivo. Nella tabella seguente vengono elencati i protocolli supportati dagli endpoint Device Advisor, nonché i metodi di autenticazione e le porte utilizzati.

Mappature tra protocolli, autenticazione e porte

Protocollo	Operazioni supportate	Autenticazione	Porta	Nome del protocollo ALPN
MQTT su WebSocket	Pubblicazione, sottoscrizione	Signature Version 4	443	N/D
MQTT	Pubblicazione, sottoscrizione	Certificato client X.509	8883	`x-amzn-mqtt-ca`
MQTT	Pubblicazione, sottoscrizione	Certificato client X.509	443	N/D

Questo capitolo contiene le sezioni seguenti:

- Configurazione
- Nozioni di base su Device Advisor nella console
- Flusso di lavoro di Device Advisor
- Flusso di lavoro della console dettagliato di Device Advisor
- Flusso di lavoro della console per test di lunga durata
- Endpoint VPC di Device Advisor (AWS PrivateLink)

- Case test di Device Advisor

Configurazione

Prima di utilizzare Device Advisor per la prima volta, completa le seguenti attività:

Creare un oggetto IoT

Innanzitutto, crea un oggetto IoT e collega un certificato a tale oggetto. Per un tutorial su come creare oggetti, consultare Creazione di un oggetto.

Creazione di un ruolo IAM da utilizzare come ruolo del dispositivo

> **Note**
>
> È possibile creare rapidamente il ruolo del dispositivo utilizzando la console Device Advisor. Per ulteriori informazioni su come configurare il ruolo del dispositivo con la console Device Advisor, consultare Nozioni di base su Device Advisor nella console.

1. Vai alla AWS Identity and Access Management console e accedi al file Account AWS che usi per i test di Device Advisor.
2. Nel pannello di navigazione a sinistra scegli Policies (Policy).
3. Scegli Crea policy.
4. Sotto Create Policy (Crea policy), effettua le operazioni seguenti:

    a. Per Service (Servizio), scegli IoT.

    b. In Azioni, esegui una delle seguenti operazioni:

    - (Consigliato) Seleziona le azioni in base alla policy collegata all'oggetto IoT o al certificato creati nella sezione precedente.
    - Cerca le seguenti azioni nella casella Filtra le azioni e selezionale:

        - `Connect`
        - `Publish`
        - `Subscribe`
        - `Receive`

- `RetainPublish`

c. In Risorse, limita il client, l'argomento e le risorse argomento. La limitazione di queste risorse è una best practice di sicurezza. Per limitare le risorse, esegui le seguenti operazioni:

i. Scegli Specifica ARN della risorsa client per l'operazione Connect.

ii. Scegli Aggiungi ARN, quindi esegui una delle seguenti operazioni:

> **Note**
>
> Il clientId è l'ID client MQTT utilizzato dal dispositivo per interagire con Device Advisor.

- Specifica Regione, accountID e clientID nell'editor ARN visivo.
- Inserisci manualmente gli Amazon Resource Names (ARNs) degli argomenti IoT con cui desideri eseguire i casi di test.

iii. Scegliere Aggiungi.

iv. Scegli Specifica l'ARN della risorsa dell'argomento per la ricezione e un'altra operazione.

v. Scegli Aggiungi ARN, quindi esegui una delle seguenti operazioni:

> **Note**
>
> Il nome argomento è l'argomento MQTT in cui il dispositivo pubblica i messaggi.

- Specifica Regione, accountID e Nome argomento nell'editor ARN visivo.
- Inserisci manualmente gli ARNs argomenti IoT con cui desideri eseguire i tuoi casi di test.

vi. Scegliere Aggiungi.

vii. Scegli Specifica ARN della risorsa topicFilter per l'azione Sottoscrivere.

viii. Scegli Aggiungi ARN, quindi esegui una delle seguenti operazioni:

> **Note**
>
> Il nome argomento è l'argomento MQTT a cui il dispositivo effettua la sottoscrizione.

- Specifica Regione, accountID e Nome argomento nell'editor ARN visivo.
- Inserisci manualmente gli ARNs argomenti IoT con cui desideri eseguire i tuoi casi di test.

ix. Scegliere Aggiungi.

5. Scegliere Next: Tags (Successivo: Tag).
6. Scegliere Next:Review (Successivo: Rivedi).
7. In Verifica policy, immetti un Nome per la policy.
8. Scegli Crea policy.
9. Nel pannello di navigazione a sinistra seleziona Roles (Ruoli).
10. Selezionare Crea ruolo.
11. In Seleziona un'entità attendibile, scegli Policy di attendibilità personalizzata.
12. Immetti la seguente policy di attendibilità nella casella Policy di attendibilità personalizzata. Per prevenire il problema "confused deputy", aggiungi le chiavi di contesto di condizione globali `aws:SourceArn` e `aws:SourceAccount` alla policy.

> **Important**
>
> È necessaria la conformità di `aws:SourceArn` con `format: arn:aws:iotdeviceadvisor:`*`region:account-id`*`:*`. Assicurati che *`region`* corrisponda alla regione AWS IoT e che *`account-id`* corrisponda all'ID dell'account cliente. Per ulteriori informazioni consulta la pagina relativa alla prevenzione del problema "confused deputy" tra servizi.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowAwsIoTCoreDeviceAdvisor",
            "Effect": "Allow",
            "Principal": {
                "Service": "iotdeviceadvisor.amazonaws.com"
        },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
            },
                "ArnLike": {
                    "aws:SourceArn":
 "arn:aws:iotdeviceadvisor:*:123456789012:suitedefinition/*"
            }
        }
        }
    ]
}
```

13. Scegli Next (Successivo).
14. Seleziona la policy creata nella Fase 4.
15. (Opzionale) In Imposta il limite delle autorizzazioni, scegli Utilizza un limite delle autorizzazioni per controllare il numero massimo di autorizzazioni del ruolo, quindi seleziona la policy creata.
16. Scegli Next (Successivo).
17. Immetti Role name (Nome del ruolo) e Role description (Descrizione del ruolo).
18. Scegli Crea ruolo.

Crea una policy gestita su misura per un utente IAM per l'uso di Device Advisor

1. Accedi alla console IAM all'indirizzo https://console.aws.amazon.com/iam/. Se viene richiesto di effettuare l'accesso, immetti le tue credenziali AWS per accedere.
2. Nel riquadro di navigazione sinistro, scegli Policy.
3. Seleziona Create policy (Crea policy), quindi scegli la scheda JSON.
4. Aggiungi le autorizzazioni necessarie per utilizzare Device Advisor. Il documento della policy è disponibile nell'argomento relativo alle best practice per la sicurezza.
5. Scegli Esamina la policy.

6. Immetti il Name (Nome) e la Description (Descrizione).
7. Scegli Crea policy.

Crea un utente IAM per utilizzare Device Advisor

> **Note**
>
> È consigliabile creare un utente IAM da utilizzare durante l'esecuzione di test di Device Advisor. L'esecuzione dei test Device Advisor da un utente amministratore può comportare rischi per la sicurezza e non è consigliata.

1. Accedi alla console IAM all'indirizzo https://console.aws.amazon.com/iam/Se richiesto, inserisci AWS le tue credenziali per accedere.
2. Nel pannello di navigazione a sinistra, seleziona Users (Utenti).
3. Scegli Add User (Aggiungi utente).
4. Immetti un User name (Nome utente).
5. Gli utenti necessitano di un accesso programmatico se desiderano interagire con utenti AWS esterni a. Console di gestione AWS Il modo per concedere l'accesso programmatico dipende dal tipo di utente che accede. AWS

   Per fornire agli utenti l’accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell’accesso programmatico?	Per	Come
IAM	(Consigliato) Utilizza le credenziali della console come credenziali temporane e per firmare le richieste programmatiche a,, o. AWS CLI AWS SDKs AWS APIs	Segui le istruzioni per l’interfaccia che desideri utilizzare. • Per la AWS CLI, consulta Login for AWS local development nella Guida per l'AWS Command Line Interface utente.

Quale utente necessita dell'accesso programmatico?	Per	Come
		• Per AWS SDKs, consulta Login for AWS local development nella AWS SDKs and Tools Reference Guide.
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporanee per firmare le richieste programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, vedere Configurazione dell'uso AWS IAM Identity Center nella AWS CLI Guida per l'utente.AWS Command Line Interface • Per AWS SDKs gli strumenti e AWS APIs, consulta l'autenticazione di IAM Identity Center nella Guida di riferimento AWS SDKs and Tools.
IAM	Utilizza credenziali temporanee per firmare le richieste programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	Seguendo le istruzioni riportate in Utilizzo delle credenziali temporanee con le AWS risorse nella Guida per l'utente IAM.

Quale utente necessita dell'accesso programmatico?	Per	Come
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli strumenti AWS SDKs e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli strumenti e agli AWS SDKs strumenti. • Per AWS APIs, consulta la sezione Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

6. Scegli Successivo: autorizzazioni.
7. Per fornire l'accesso, aggiungi autorizzazioni agli utenti, gruppi o ruoli:

   • Utenti e gruppi in: AWS IAM Identity Center

   Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center .

   • Utenti gestiti in IAM tramite un provider di identità:

   Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Create a role for a third-party identity provider (federation) della Guida per l'utente IAM.

- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Segui le istruzioni riportate nella pagina Create a role for an IAM user della Guida per l'utente IAM.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente IAM.

8. Nella casella di ricerca, immetti il nome della policy gestita dal cliente creata. Quindi, seleziona la casella di controllo per Nome della policy.
9. Scegliere Next: Tags (Successivo: Tag).
10. Scegliere Next:Review (Successivo:Rivedi).
11. Seleziona Create user (Crea utente).
12. Scegli Close (Chiudi).

Device Advisor richiede l'accesso alle tue AWS risorse (oggetti, certificati ed endpoint) per tuo conto. L'utente IAM deve disporre delle autorizzazioni necessarie. Device Advisor pubblicherà anche i log su Amazon CloudWatch se alleghi la politica di autorizzazioni necessaria al tuo utente IAM.

Configurazione del dispositivo

Device Advisor utilizza l'estensione TLS (SNI) per applicare le configurazioni di TLS. I dispositivi devono utilizzare questa estensione quando si collegano e passare un nome server identico all'endpoint di test di Device Advisor.

Device Advisor consente la connessione TLS quando un test è nello stato `Running`, ma nega la connessione TLS prima e dopo ogni esecuzione di un test. Per questo motivo, è consigliabile utilizzare il meccanismo di riconnessione del dispositivo per un'esperienza di test completamente automatizzata con Device Advisor. È possibile eseguire gruppi di test che includono più casi di test, ad esempio connessione TLS, connessione MQTT e pubblicazione MQTT. Se si eseguono più casi di test, è opportuno che il dispositivo tenti di connettersi all'endpoint di test ogni cinque secondi. È quindi possibile automatizzare l'esecuzione di più casi di test in sequenza.

> **Note**
>
> Per preparare il software del dispositivo per il test, è consigliabile utilizzare una SDK in grado di connettersi a AWS IoT Core. Occorre quindi aggiornare l'SDK con l'endpoint di test di Device Advisor fornito per Account AWS.

Device Advisor supporta due tipi di endpoint: endpoint a livello di account ed endpoint a livello di dispositivo. Scegli l'endpoint che meglio si adatta al tuo caso d'uso. Per eseguire contemporaneamente più gruppi di test per dispositivi differenti, utilizza un endpoint a livello di dispositivo.

Eseguire il seguente comando per ottenere l'endpoint a livello di dispositivo:

Per i clienti MQTT che utilizzano certificati client X.509:

```
aws iotdeviceadvisor get-endpoint --thing-arn your-thing-arn
```

or

```
aws iotdeviceadvisor get-endpoint --certificate-arn your-certificate-arn
```

Per i WebSocket clienti che utilizzano la versione 4 di MQTT che utilizzano Signature:

```
aws iotdeviceadvisor get-endpoint --device-role-arn your-device-role-arn --
authentication-method SignatureVersion4
```

Per eseguire una suite di test alla volta, scegli un endpoint a livello di account. Esegui il seguente comando per ottenere l'endpoint a livello di account:

```
aws iotdeviceadvisor get-endpoint
```

Nozioni di base su Device Advisor nella console

Questo tutorial ti aiuta a iniziare a usare AWS IoT Core Device Advisor la console. Device Advisor offre funzionalità quali i test obbligatori e i report di qualifica firmati. È possibile utilizzare questi test

e report per qualificare ed elencare i dispositivi nel Catalogo dei dispositivi dei Partner AWS, come illustrato in AWS IoT Core qualification program.

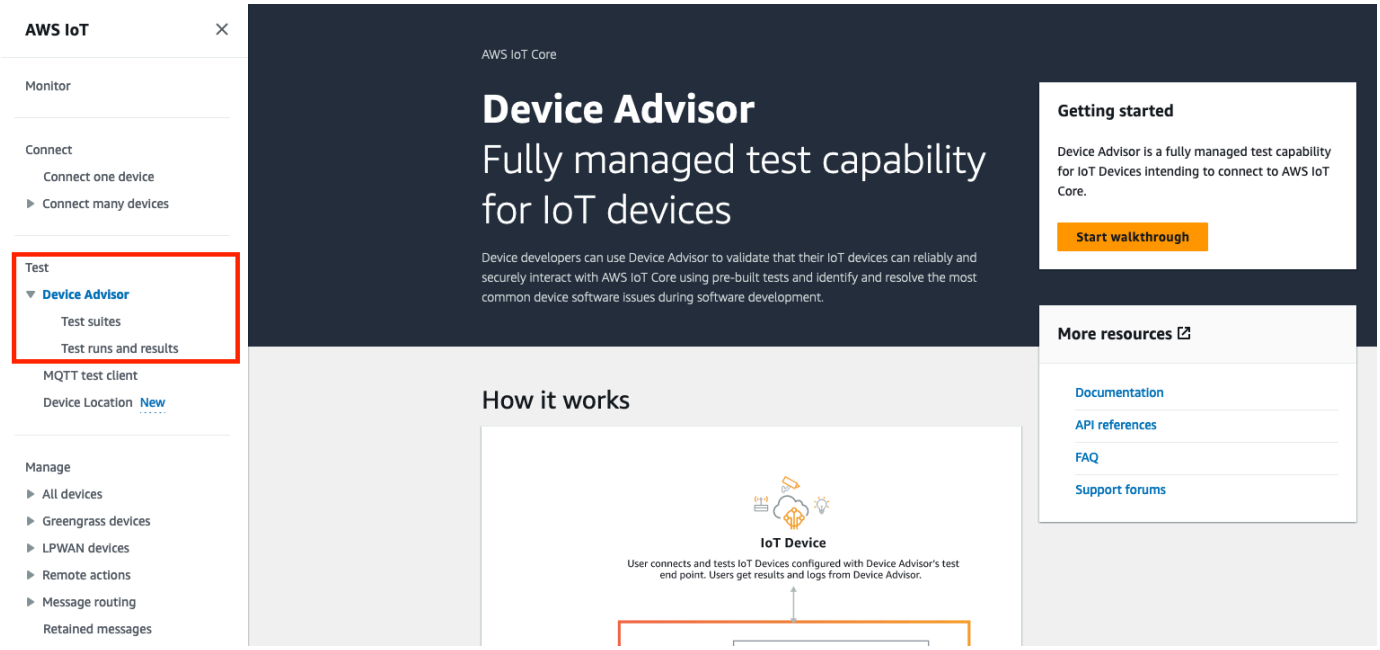
Per ulteriori informazioni sull'utilizzo di Device Advisor, consulta Flusso di lavoro di Device Advisor e Flusso di lavoro della console dettagliato di Device Advisor.

Per completare questo tutorial, attieniti alla procedura illustrata in Configurazione.

> **Note**
>
> Device Advisor è supportato nelle seguenti versioni Regioni AWS:
>
> - Stati Uniti orientali (Virginia settentrionale)
> - Stati Uniti occidentali (Oregon)
> - Asia Pacifico (Tokyo)
> - Europa (Irlanda)

Nozioni di base

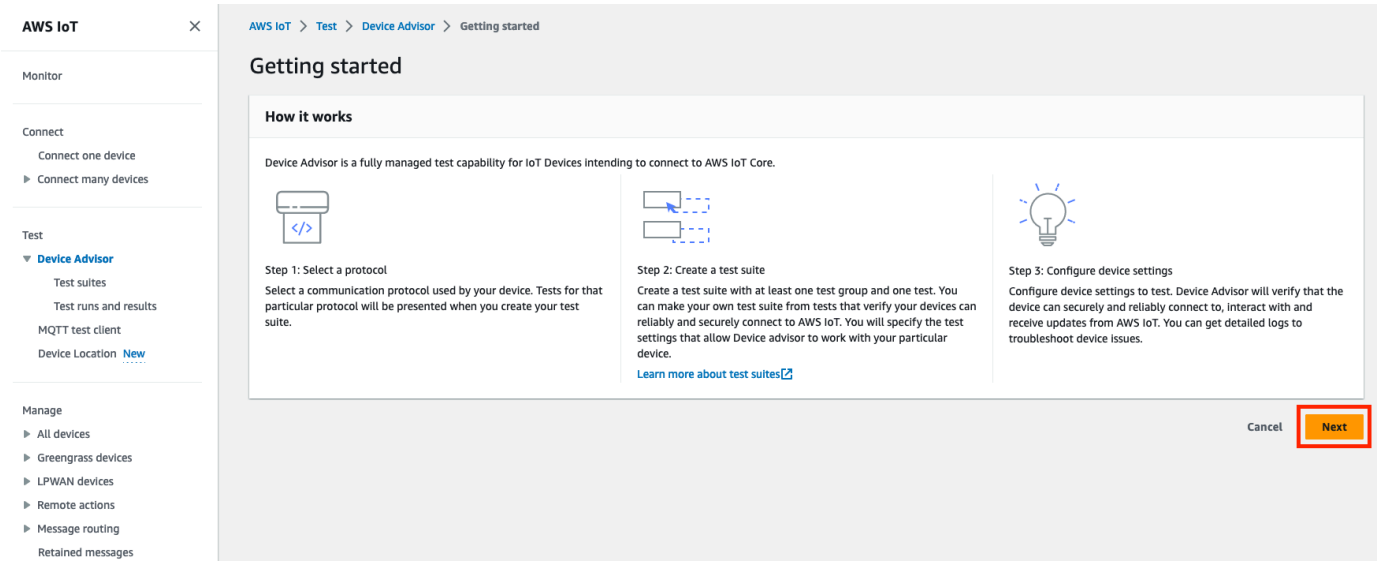
1. Nel pannello di navigazione della console AWS IoT in Test, scegli Device Advisor. Quindi, scegli il pulsante Avvia la spiegazione passo per passo sulla console.



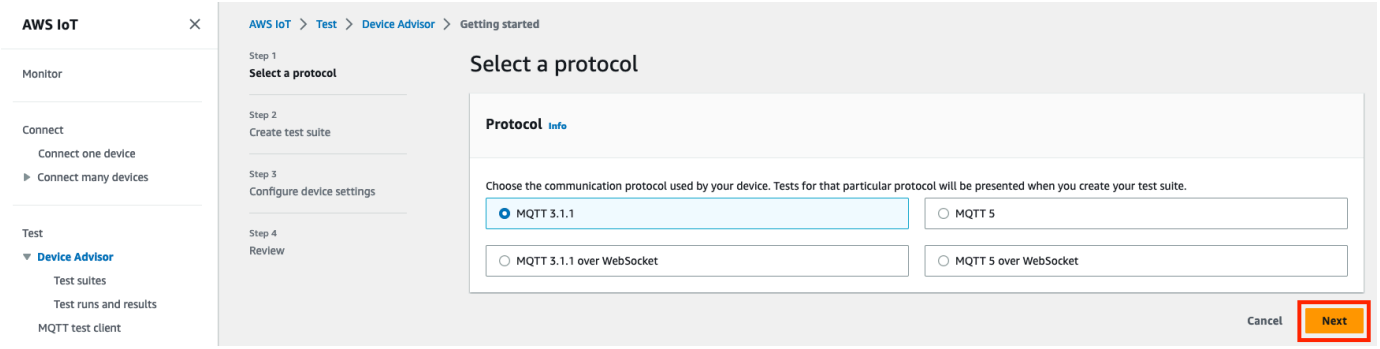
2. Nella pagina Nozioni di base su Device Advisor viene fornita una panoramica dei passaggi richiesti per creare una suite di test ed eseguire test sul dispositivo. Qui è anche possibile trovare

l'endpoint di test di Device Advisor per il proprio account. Per eseguire la connessione a questo endpoint di test, è necessario configurare il firmware o il software sul dispositivo utilizzato per il test.

Per completare questo tutorial, devi prima creare un oggetto e un certificato. Dopo aver esaminato le informazioni nella sezione Come funziona, scegli Avanti.

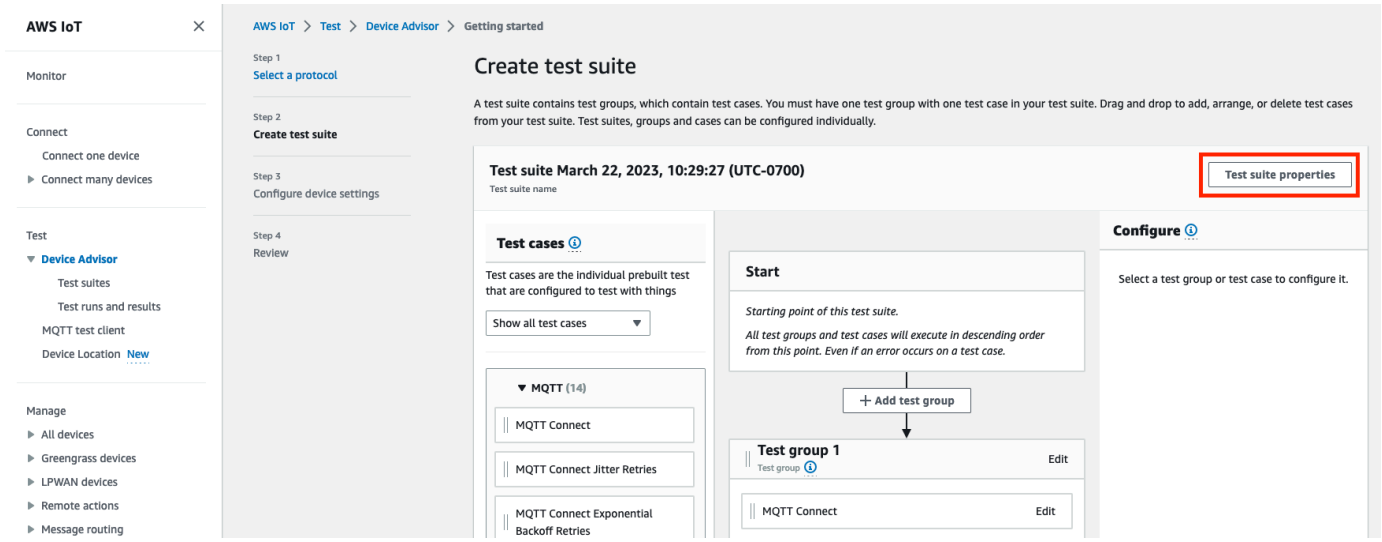


3. Nella Fase 1: Selezione di un protocollo, seleziona un protocollo dalle opzioni elencate. Quindi, seleziona Successivo.



4. In Step 2 (Fase 2), è possibile creare e configurare una suite di test personalizzata. Una suite di test personalizzata deve avere almeno un gruppo di test e ogni gruppo di test deve avere almeno un test case. Abbiamo aggiunto il test case MQTT Connect (Collegamento MQTT) per iniziare.

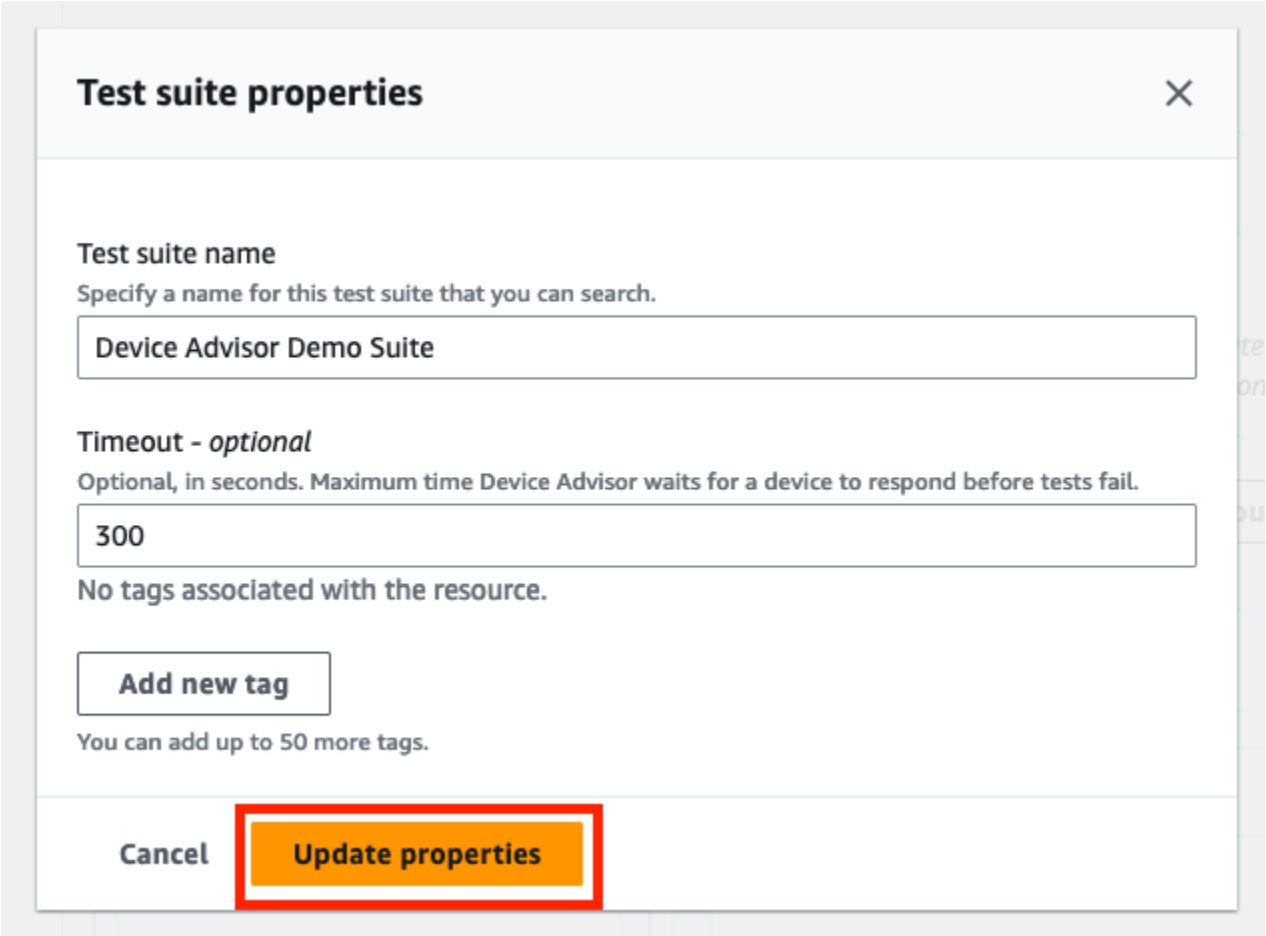
   Scegli Test suite properties (Proprietà della suite di test).



Fornisci le proprietà della suite di test quando crei la tua suite di test. È possibile configurare le seguenti proprietà a livello di suite:

- Test suite name (Nome della suite di test): Inserisci un nome per la suite di test.
- Timeout (facoltativo): il timeout (in secondi) per ogni caso di test nella suite di test corrente. Se non specifichi un valore di timeout, viene utilizzato il valore predefinito.
- Tags (Tag) (facoltativo): Aggiungi tag alla suite di test.

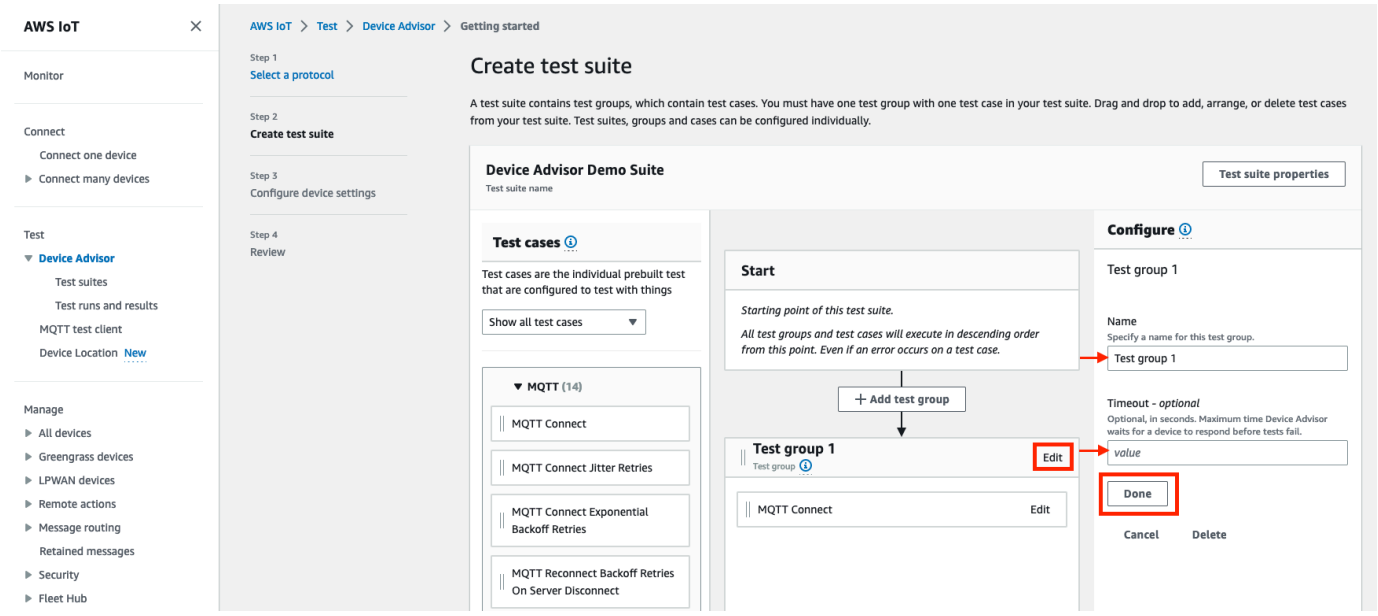
Al termine, scegli Update properties (Aggiorna proprietà).



5. (Facoltativo) Per aggiornare la configurazione del gruppo di suite di test, scegli il pulsante Modifica accanto al nome del gruppo di test.

   - Name (Nome): inserisci un nome personalizzato per il gruppo della suite di test.
   - Timeout (facoltativo): il timeout (in secondi) per ogni caso di test nella suite di test corrente. Se non specifichi un valore di timeout, viene utilizzato il valore predefinito.

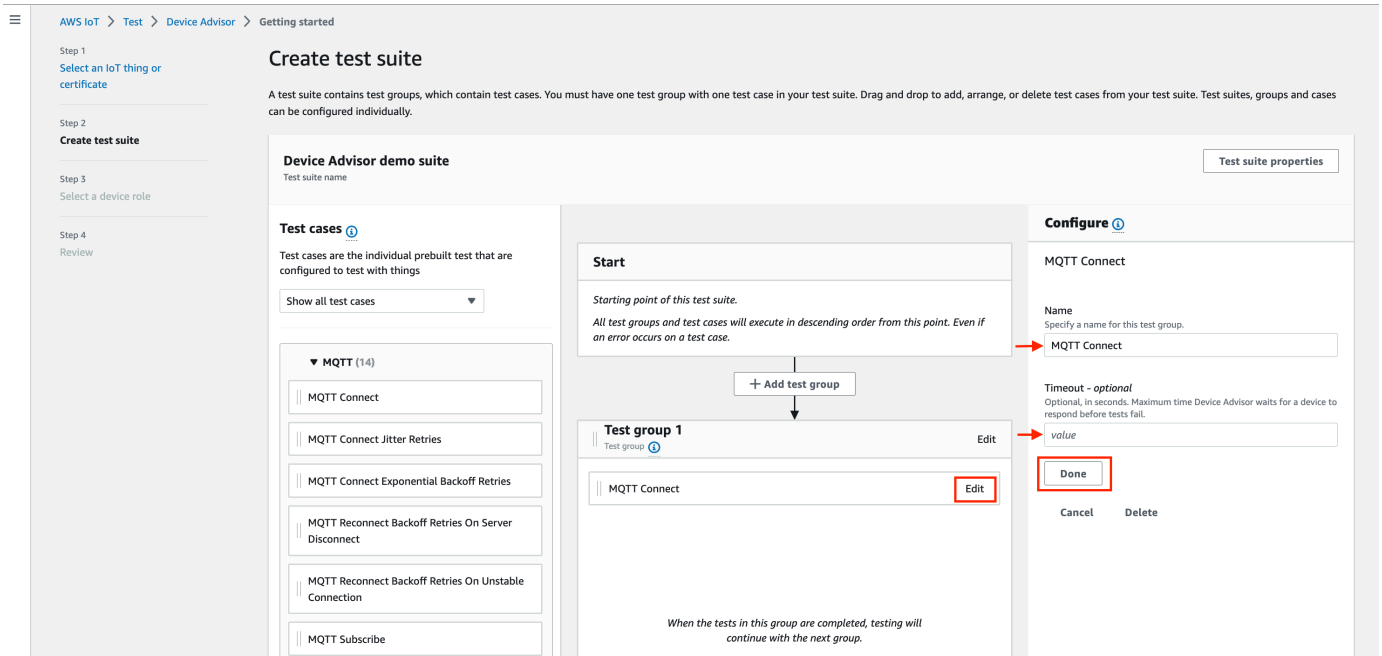
   Al termine, scegli Fine per continuare.



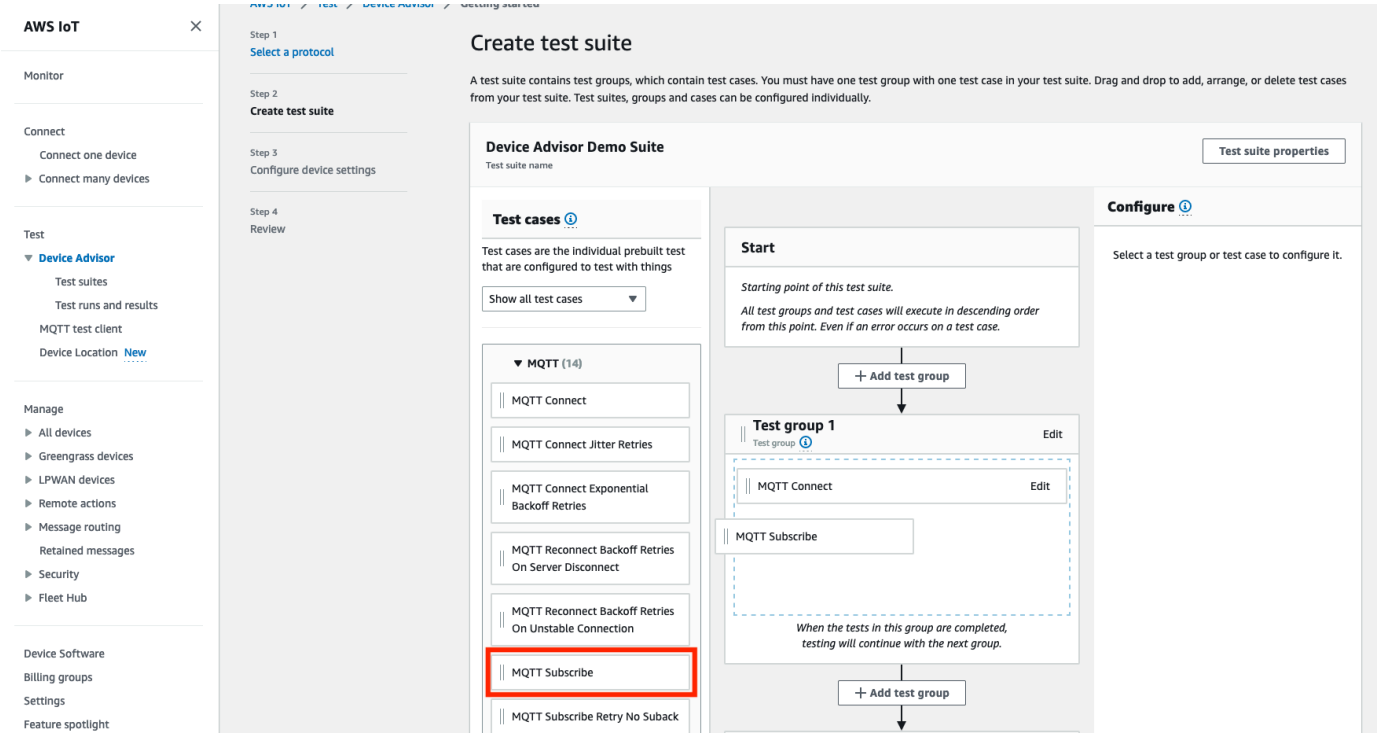
6. (Facoltativo) Per aggiornare la configurazione del caso di test per un caso di test, scegli il pulsante Modifica accanto al nome del caso di test.

   - Name (Nome): inserisci un nome personalizzato per il gruppo della suite di test.
   - Timeout (facoltativo): il timeout (in secondi) per il caso di test selezionato. Se non specifichi un valore di timeout, viene utilizzato il valore predefinito.

   Al termine, scegli Fine per continuare.

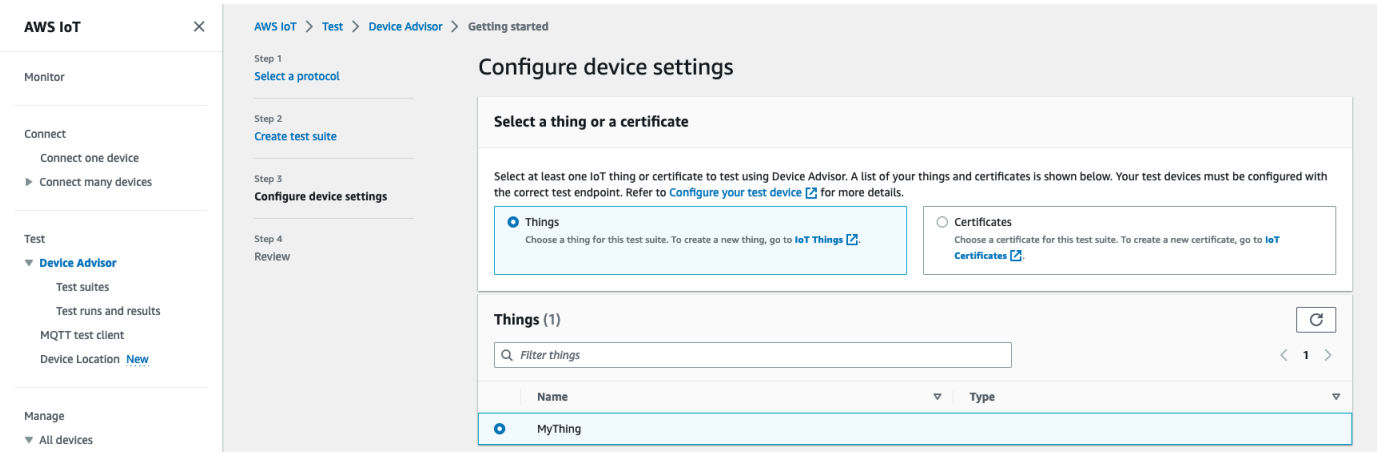


7. (Facoltativo) Per aggiungere altri gruppi di test alla suite di test, scegli Aggiungi un gruppo di test e segui le istruzioni nella Fase 5.
8. (Facoltativo) Per aggiungere altri casi di test, trascina i casi di test nella sezione Casi di test in uno qualsiasi dei gruppi di test.



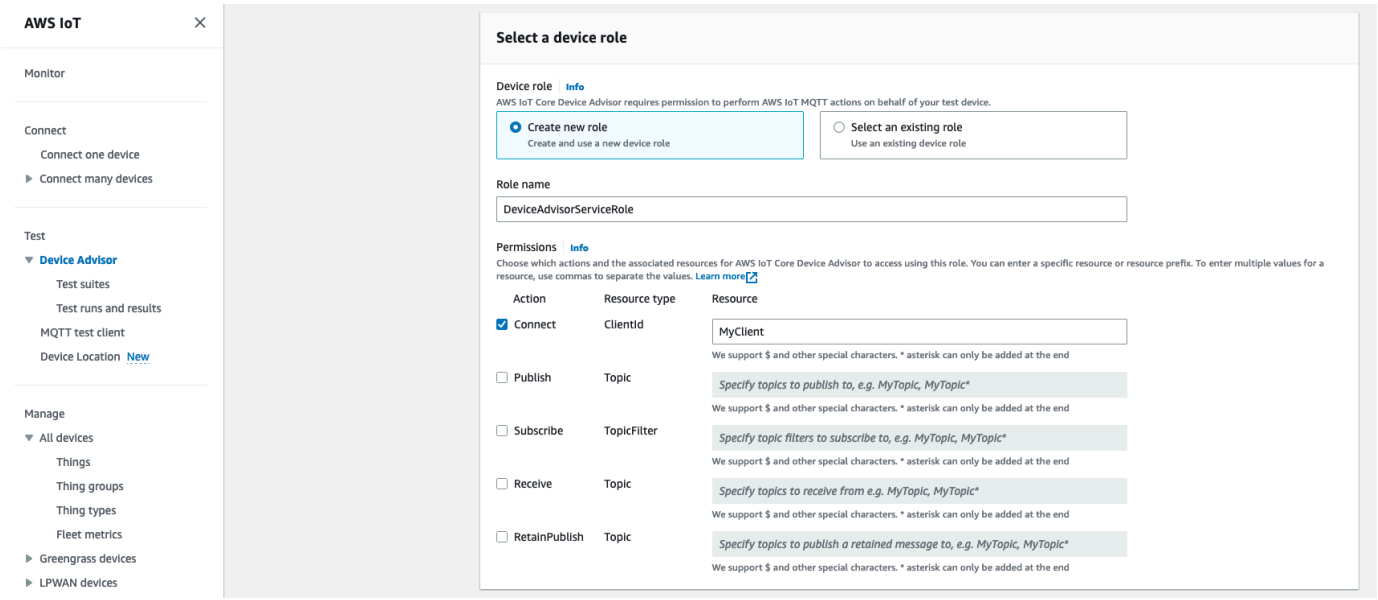
9. Puoi modificare l'ordine dei gruppi di test e dei casi di test. Per apportare modifiche, trascina i casi di test elencati verso l'alto o verso il basso nell'elenco. Device Advisor esegue i test nell'ordine in cui sono stati elencati.

   Dopo aver configurato la suite di test, scegli Next (Successivo).
10. Nel passaggio 3, seleziona un AWS IoT elemento o un certificato da testare utilizzando Device Advisor. Se non sono presenti elementi o certificati esistenti, consulta Configurazione.

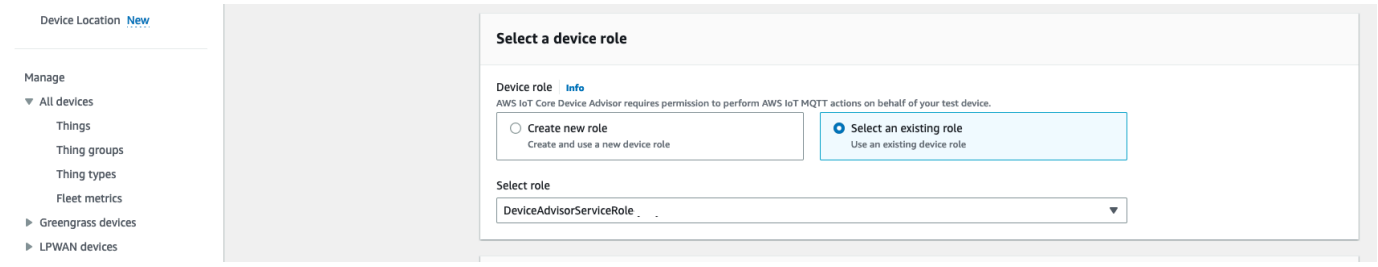


11. È possibile configurare un ruolo del dispositivo utilizzato da Device Advisor per eseguire azioni AWS IoT MQTT per conto del dispositivo di test. Solo per il caso di test Connessione MQTT, l'operazione Connettersi viene selezionata automaticamente. Questo perché il ruolo del dispositivo richiede questa autorizzazione per eseguire la suite di test. Per altri casi di test, vengono selezionate le operazioni corrispondenti.

    Fornisci i valori delle risorse per ciascuna operazione selezionata. Ad esempio, per l'azione Connettersi, fornisci l'ID client utilizzato dal dispositivo per connettersi all'endpoint Device Advisor. Puoi fornire più valori con valori separati da virgole e valori prefisso con un carattere jolly (*). Ad esempio, per fornire l'autorizzazione per pubblicare su qualsiasi argomento che inizia con `MyTopic`, immetti **`MyTopic*`** come valore della risorsa.

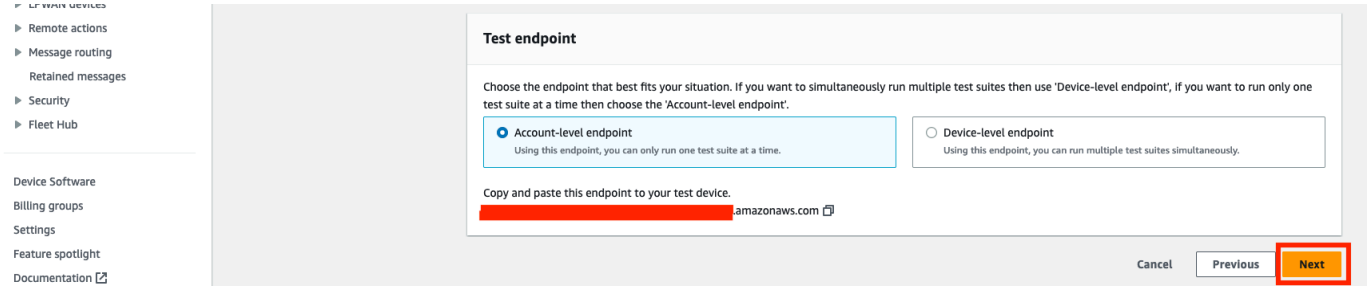


    Per utilizzare un ruolo del dispositivo creato in precedenza da Impostazione, scegli Seleziona ruolo esistente. Quindi scegli il ruolo del dispositivo in Seleziona ruolo.



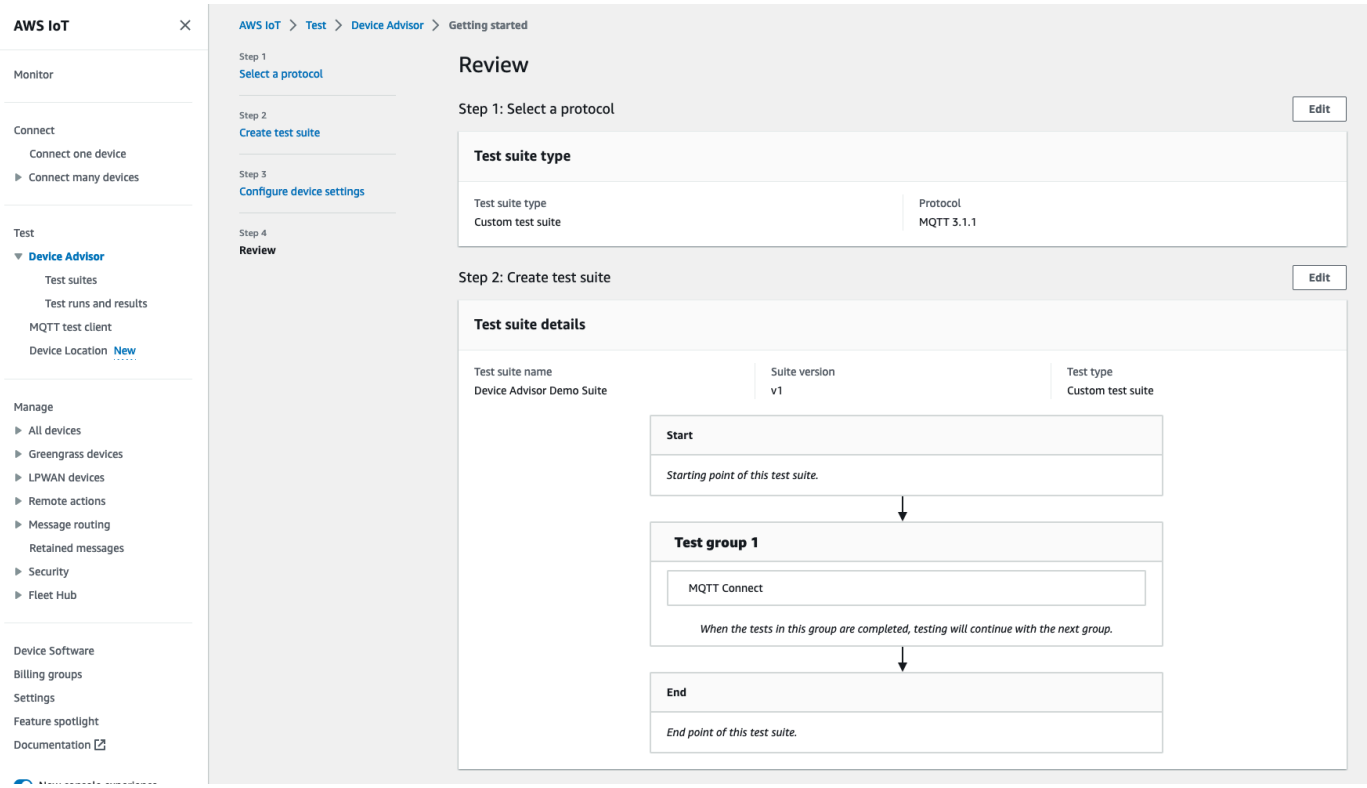
    Configura il ruolo del dispositivo utilizzando una delle due opzioni fornite, quindi scegli Avanti.

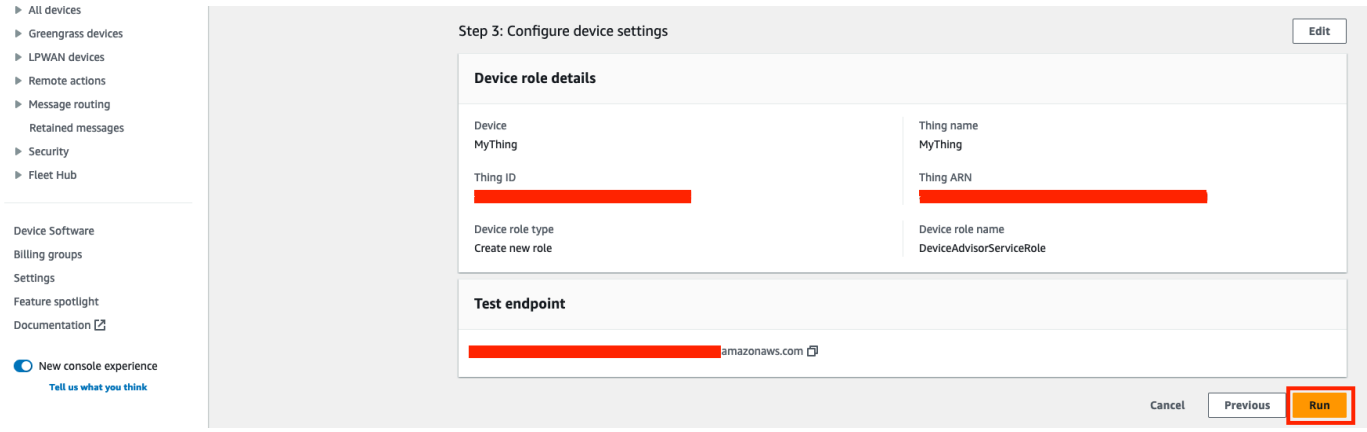
12. Nella sezione Endpoint di test, seleziona l'endpoint più adatto al caso d'uso. Per eseguire più suite di test contemporaneamente con la stessa suite Account AWS, seleziona Endpoint a livello di dispositivo. Per eseguire una suite di test alla volta, seleziona Endpoint a livello di account.



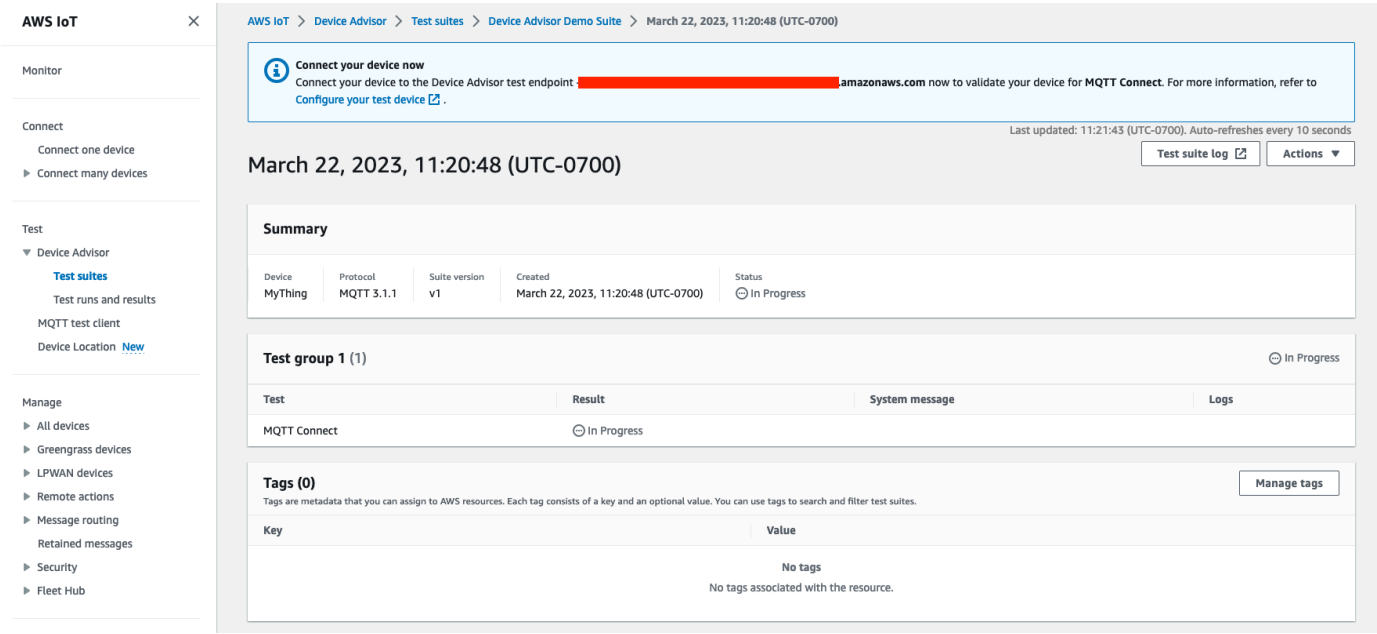
13. Nella Fase 4 viene illustrata una panoramica del dispositivo di test, dell'endpoint di test, della suite di test e del ruolo del dispositivo di test selezionati e configurati. Per apportare modifiche a una sezione, scegli il pulsante Modifica relativo alla sezione che desideri modificare. Dopo aver confermato la configurazione del test, scegli Esegui per creare la suite di test ed eseguire i test.

> **Note**
>
> Per risultati ottimali, puoi collegare il dispositivo di test selezionato all'endpoint di test di Device Advisor prima di avviare l'esecuzione della suite di test. Si consiglia di disporre di un meccanismo creato per il dispositivo per provare a connettersi all'endpoint di test ogni cinque secondi per uno o due minuti.





14. Nel pannello di navigazione, in Test, scegli Device Advisor, quindi seleziona Esecuzioni di test e risultati. Seleziona l'esecuzione di una suite di test per visualizzare i dettagli e i log.



15. Per accedere ai CloudWatch log di Amazon per la suite, esegui:

    - Scegli Test suite log per visualizzare CloudWatch i log relativi all'esecuzione della suite di test.
    - Scegli Test case log per qualsiasi test case per visualizzare i log specifici del test case CloudWatch .

16. In base ai risultati del test, risolvi i problemi del tuo dispositivo fino a quando tutti i test vengono superati.

Flusso di lavoro di Device Advisor

In questo tutorial viene descritto come creare una suite di test personalizzata ed eseguire test sul dispositivo che si desidera testare nella console. Dopo aver completato i test, è possibile visualizzare i risultati del test e i registri dettagliati.

Prerequisiti

Prima di iniziare questo tutorial, completa la procedura illustrata in Configurazione.

Crea una definizione di suite di test

Innanzitutto, installa un SDK. AWS

`rootGroup` sintassi

Un gruppo root è una stringa JSON che specifica quali casi di test includere nella suite di test. Inoltre, consente di specificare le eventuali configurazioni necessarie per tali casi di test. Utilizza il gruppo root per strutturare e ordinare la suite di test in base alle tue esigenze. La gerarchia di una suite di test è:

```
test suite # test group(s) # test case(s)
```

Una suite di test deve avere almeno un gruppo di test e ogni gruppo di test deve avere almeno un test case. Device Advisor esegue i test nell'ordine in cui vengono definiti i gruppi di test e i test case.

Ogni gruppo radice segue questa struttura di base:

```
{
    "configuration": {  // for all tests in the test suite
        "": ""
    }
    "tests": [{
        "name": ""
        "configuration": {  // for all sub-groups in this test group
            "": ""
        },
        "tests": [{
            "name": ""
```

```
            "configuration": {  // for all test cases in this test group
                "": ""
            },
            "test": {
                "id": ""
                "version": ""
            }
        }]
    }]
}
```

Nel gruppo root, definisci la suite di test con un `name`, `configuration` e i `tests` contenuti nel gruppo. Il gruppo `tests` contiene le definizioni di singoli test. Definisci ogni test con un `name`, `configuration` e un blocco `test` che definisce i casi di test per tale test. Infine, ogni caso di test è definito con un `id` e una `version`.

Per informazioni su come utilizzate i campi `"id"` e `"version"` per ogni caso di test (blocco `test`), consultare Case test di Device Advisor. Questa sezione contiene anche informazioni sulle impostazioni `configuration` disponibili.

Il blocco seguente è un esempio di configurazione del gruppo root. Queste configurazioni specificano i casi di test MQTT Connect Happy Case e MQTT Connect Exponential Backoff Retries, insieme alle descrizioni dei campi di configurazione.

```
{
    "configuration": {},  // Suite-level configuration
    "tests": [            // Group definitions should be provided here
      {
        "name": "My_MQTT_Connect_Group",  // Group definition name
        "configuration": {}               // Group definition-level configuration,
        "tests": [                        // Test case definitions should be provided
 here
        {
            "name": "My_MQTT_Connect_Happy_Case",  // Test case definition name
            "configuration": {
                "EXECUTION_TIMEOUT": 300        // Test case definition-level
 configuration, in seconds
            },
            "test": {
                "id": "MQTT_Connect",              // test case id
                "version": "0.0.0"                 // test case version
```

```
            }
        },
        {
            "name": "My_MQTT_Connect_Jitter_Backoff_Retries",  // Test case definition
 name
            "configuration": {
                "EXECUTION_TIMEOUT": 600                // Test case definition-level
 configuration,  in seconds
            },
            "test": {
                "id": "MQTT_Connect_Jitter_Backoff_Retries",  // test case id
                "version": "0.0.0"                                  // test case version
            }
        }]
    }]
}
```

È necessario fornire la configurazione del gruppo radice quando si crea la definizione della suite di test. Salva il `suiteDefinitionId` restituito nell'oggetto di risposta. Puoi utilizzare questo ID per recuperare le informazioni sulla definizione della suite di test ed eseguire la suite di test.

Ecco un esempio di Java SDK:

```
response = iotDeviceAdvisorClient.createSuiteDefinition(
        CreateSuiteDefinitionRequest.builder()
            .suiteDefinitionConfiguration(SuiteDefinitionConfiguration.builder()
                .suiteDefinitionName("your-suite-definition-name")
                .devices(
                    DeviceUnderTest.builder()
                        .thingArn("your-test-device-thing-arn")
                        .certificateArn("your-test-device-certificate-arn")
                        .deviceRoleArn("your-device-role-arn") //if using SigV4 for
 MQTT over WebSocket
                        .build()
                )
                .rootGroup("your-root-group-configuration")
                .devicePermissionRoleArn("your-device-permission-role-arn")
                .protocol("MqttV3_1_1 || MqttV5 || MqttV3_1_1_OverWebSocket ||
 MqttV5_OverWebSocket")
                .build()
            )
            .build()
)
```

Ottieni una definizione di suite di test

Dopo aver creato la definizione della suite di test, riceverai `suiteDefinitionId` nell'oggetto risposta dell'operazione API `CreateSuiteDefinition`.

Quando l'operazione restituisce il `suiteDefinitionId`, è possibile che vengano visualizzati nuovi campi `id` all'interno di ciascun gruppo e definizione del caso di test all'interno del gruppo root. Puoi usarli IDs per eseguire un sottoinsieme della definizione della tua suite di test.

Esempio di Java SDK:

```
response = iotDeviceAdvisorClient.GetSuiteDefinition(
    GetSuiteDefinitionRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
        .build()
)
```

Ottieni un endpoint di test

Utilizza l'operazione API `GetEndpoint` per ottenere l'endpoint di test utilizzato dal dispositivo. Seleziona l'endpoint più adatto al tuo test. Per eseguire contemporaneamente più suite di test, utilizza l'endpoint a livello di dispositivo fornendo un `thing ARN`, `certificate ARN` o `device role ARN`. Per eseguire una singola suite di test, non fornire argomenti all' GetEndpoint operazione di scelta dell'endpoint a livello di account.

Esempi di SDK:

```
response = iotDeviceAdvisorClient.getEndpoint(GetEndpointRequest.builder()
.certificateArn("your-test-device-certificate-arn")
.thingArn("your-test-device-thing-arn")
.deviceRoleArn("your-device-role-arn") //if using SigV4 for MQTT over WebSocket

.build())
```

Avvia una suite di test

Dopo che hai creato una definizione della suite di test e configurato il dispositivo di test per la connessione all'endpoint di test Device Advisor, esegui la suite di test con l'API `StartSuiteRun`.

Per i clienti MQTT, utilizza `certificateArn` o `thingArn` per eseguire la suite di test. Se entrambi sono configurati, il certificato viene utilizzato se appartiene all'oggetto.

Per MQTT over WebSocket customer, utilizzare `deviceRoleArn` per eseguire la suite di test. Se il ruolo specificato è diverso da quello descritto nella definizione della suite di test, il ruolo specificato sostituisce il ruolo definito.

Per `.parallelRun()`, utilizza `true` se l'endpoint a livello di dispositivo viene utilizzato per eseguire più suite di test in parallelo utilizzando un Account AWS.

Esempi di SDK:

```
response = iotDeviceAdvisorClient.startSuiteRun(StartSuiteRunRequest.builder()
.suiteDefinitionId("your-suite-definition-id")
.suiteRunConfiguration(SuiteRunConfiguration.builder()
    .primaryDevice(DeviceUnderTest.builder()
        .certificateArn("your-test-device-certificate-arn")
        .thingArn("your-test-device-thing-arn")
        .deviceRoleArn("your-device-role-arn") //if using SigV4 for MQTT over WebSocket

        .build())
    .parallelRun(true | false)
    .build())
.build())
```

Salva la `suiteRunId` dalla risposta. Lo utilizzerai per recuperare i risultati dell'esecuzione di questa suite di test.

Ottieni un'esecuzione della suite di test

Dopo aver avviato un'esecuzione di suite di test, è possibile verificarne lo stato di avanzamento e i risultati con l'API `GetSuiteRun`.

Esempi di SDK:

```
// Using the SDK, call the GetSuiteRun API.

response = iotDeviceAdvisorClient.GetSuiteRun(
GetSuiteRunRequest.builder()
    .suiteDefinitionId("your-suite-definition-id")
    .suiteRunId("your-suite-run-id")
.build())
```

Interrompi l'esecuzione di una suite di test

Per interrompere l'esecuzione di una suite di test ancora in corso, puoi chiamare l'operazione API `StopSuiteRun`. Dopo aver chiamato l'operazione API `StopSuiteRun`, il servizio avvia il processo di pulizia. Mentre il servizio esegue il processo di pulizia, lo stato di esecuzione della suite di test viene aggiornato a `Stopping`. Il processo di pulizia può richiedere alcuni minuti. Una volta completato il processo, lo stato di esecuzione della suite di test viene aggiornato a `Stopped`. Dopo che un'esecuzione di test si è completamente interrotta, puoi avviare un'altra esecuzione di suite di test. Puoi controllare periodicamente lo stato di esecuzione della suite utilizzando l'operazione API `GetSuiteRun`, come illustrato nella sezione precedente.

Esempi di SDK:

```
// Using the SDK, call the StopSuiteRun API.

response = iotDeviceAdvisorClient.StopSuiteRun(
StopSuiteRun.builder()
    .suiteDefinitionId("your-suite-definition-id")
    .suiteRunId("your-suite-run-id")
.build())
```

Ottieni un rapporto di qualificazione per una suite di test di qualificazione riuscita

Se esegui una suite di test di qualifica completata correttamente, puoi recuperare un report di qualifica con l'operazione API `GetSuiteRunReport`. Utilizza questo report di qualifica per qualificare il dispositivo con il Programma di idoneità di AWS IoT Core . Per determinare se la suite di test è una suite di test di qualificazione, verifica se il parametro `intendedForQualification` è impostato su `true`. Dopo aver chiamato l'operazione API `GetSuiteRunReport`, puoi scaricare il report dall'URL restituito per un massimo di 90 secondi. Se trascorrono più di 90 secondi dalla precedente chiamata dell'operazione `GetSuiteRunReport`, chiama nuovamente l'operazione per recuperare un nuovo URL valido.

Esempi di SDK:

```
// Using the SDK, call the getSuiteRunReport API.

response = iotDeviceAdvisorClient.getSuiteRunReport(
```

```
    GetSuiteRunReportRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
        .suiteRunId("your-suite-run-id")
        .build()
)
```

Flusso di lavoro della console dettagliato di Device Advisor

In questo tutorial, creerai una suite di test personalizzata ed eseguirai test sul dispositivo che desideri testare nella console. Dopo aver completato i test, è possibile visualizzare i risultati del test e i registri dettagliati.

Esercitazioni

- Prerequisiti
- Crea una definizione di suite di test
- Avvia una suite di test
- Interrompi l'esecuzione di una suite di test (facoltativo)
- Visualizza i dettagli e i registri di esecuzione della suite
- Scarica un rapporto di AWS IoT qualificazione

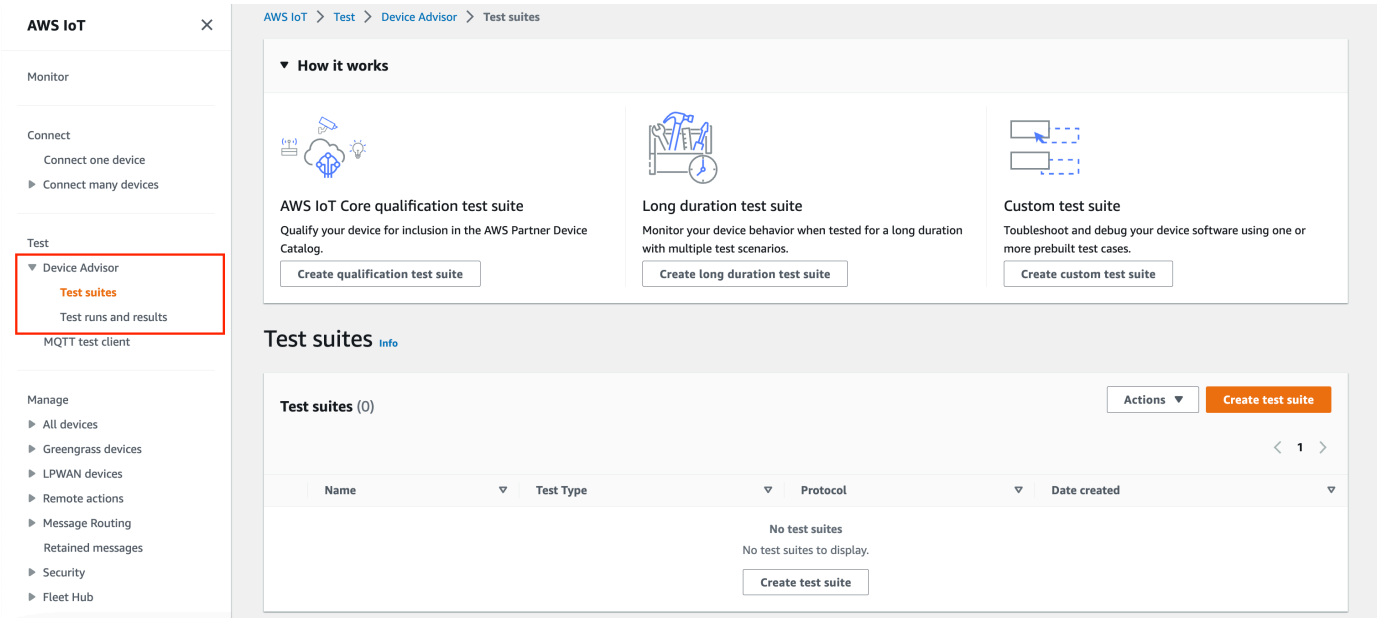
Prerequisiti

Per completare questo tutorial, è necessario creare un oggetto e un certificato.

Crea una definizione di suite di test

Crea una suite di test in modo da poterla eseguire sui tuoi dispositivi ed eseguire la verifica.

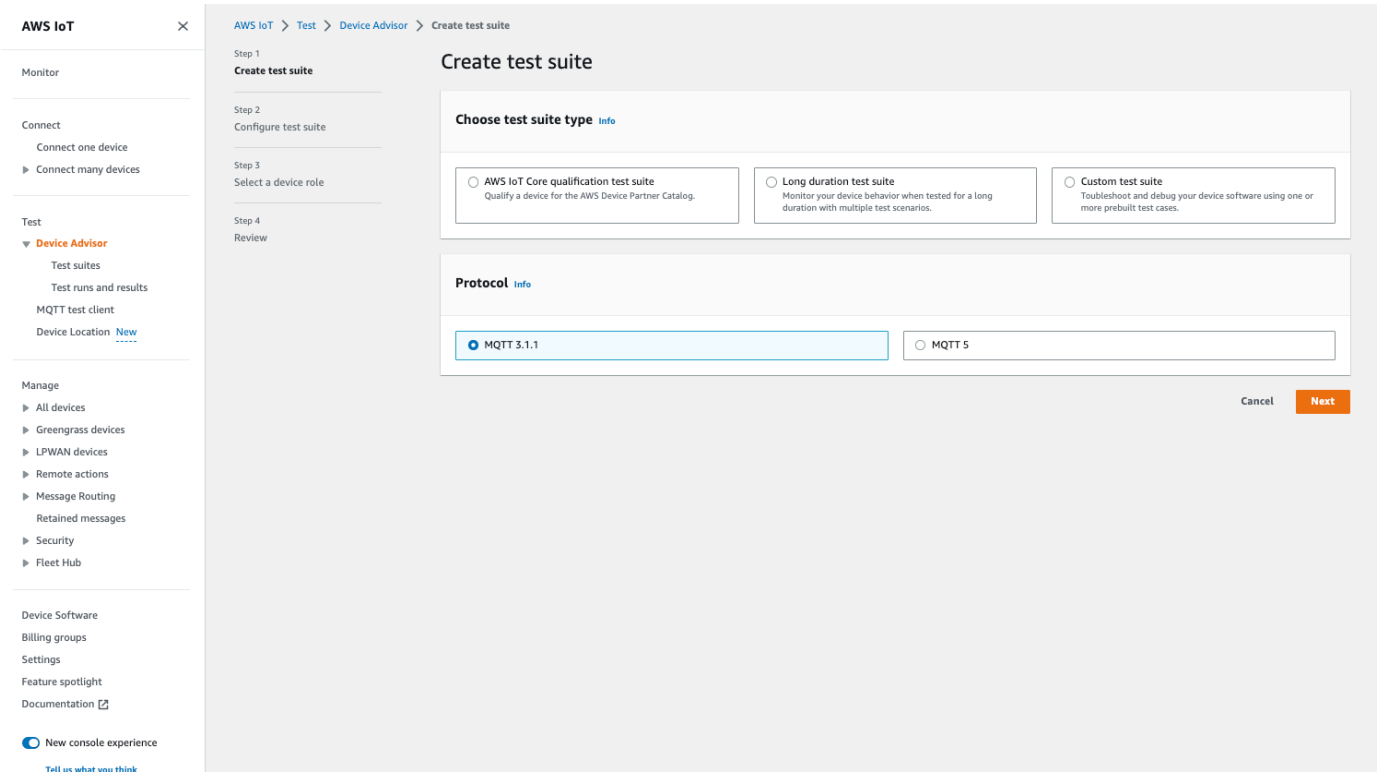
1. Nella console AWS IoT, nel pannello di navigazione, espandi Test, Device Advisor e scegli Test suites (Suite di test).



Scegli Create test suite (Crea suite di test).

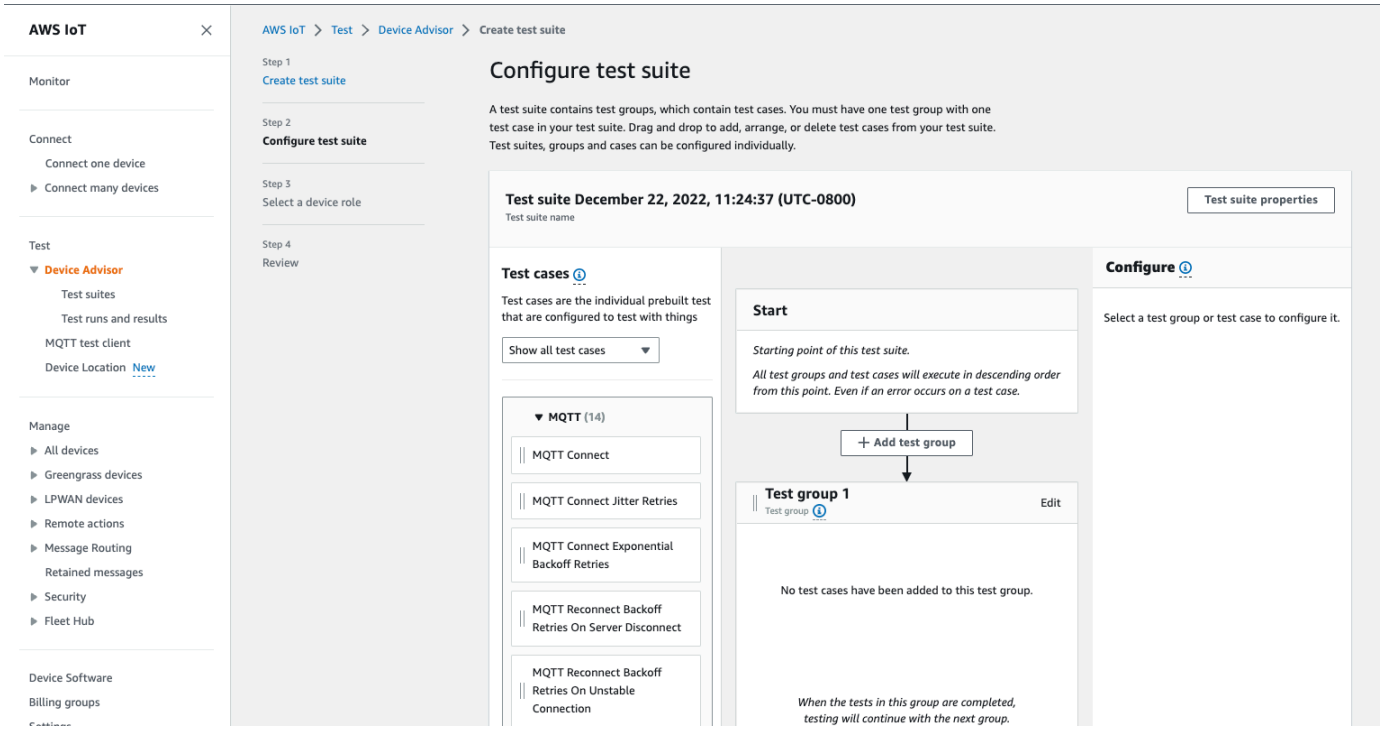
2. Seleziona `Use the AWS Qualification test suite` o `Create a new test suite`.

Per il protocollo, scegli MQTT 3.1.1 o MQTT 5.

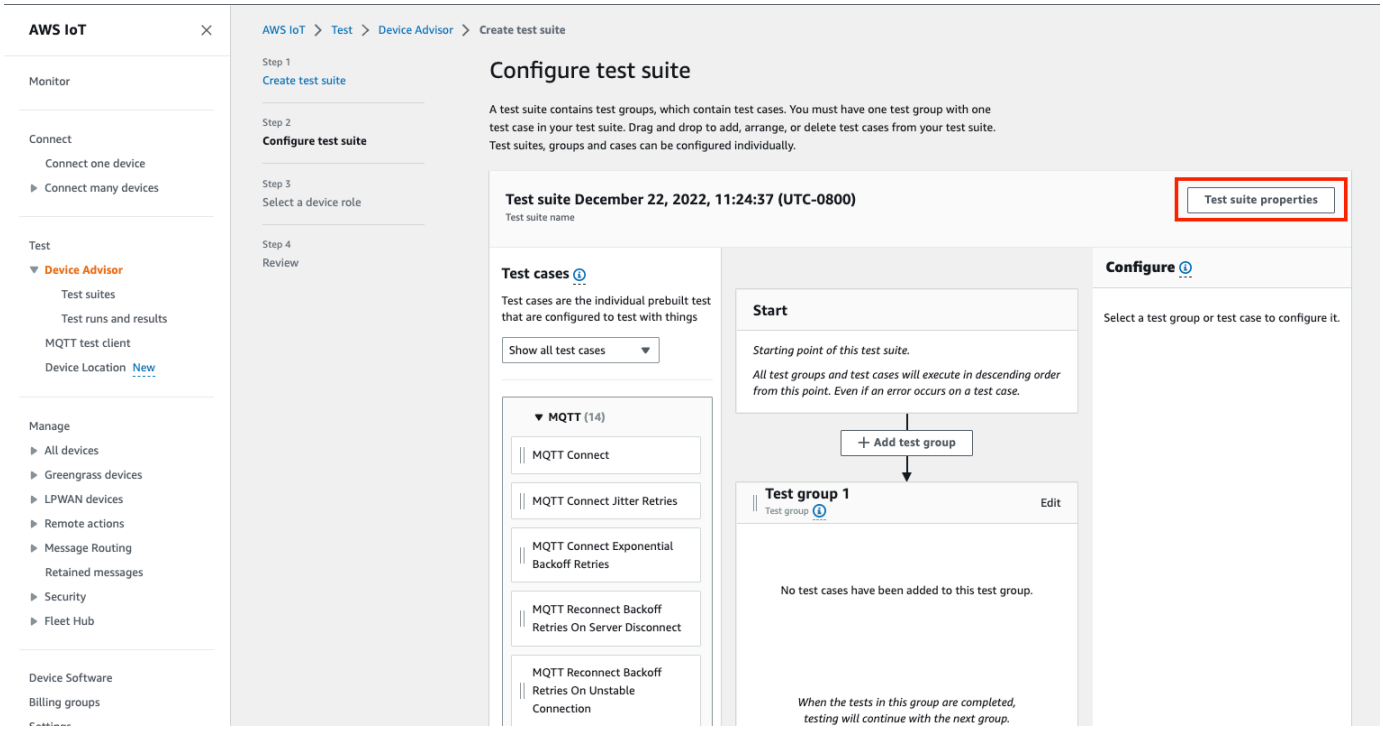


Seleziona questa opzione `Use the AWS Qualification test suite` e inserisci il tuo dispositivo nel Partner Device Catalog. AWS Scegliendo questa opzione, preselezioni i test case necessari per la qualificazione del dispositivo al programma di qualificazione AWS IoT Core . I gruppi di test e i test case non possono essere aggiunti o rimossi. Dovrai comunque configurare le proprietà della suite di test.

Scegli `Create a new test suite` per creare e configurare una suite di test personalizzata. Si consiglia di iniziare con questa opzione per il test iniziale e la risoluzione dei problemi. Una suite di test personalizzata deve avere almeno un gruppo di test e ogni gruppo di test deve avere almeno un test case. Ai fini di questo tutorial, selezioneremo questa opzione e sceglieremo Next (Successivo).

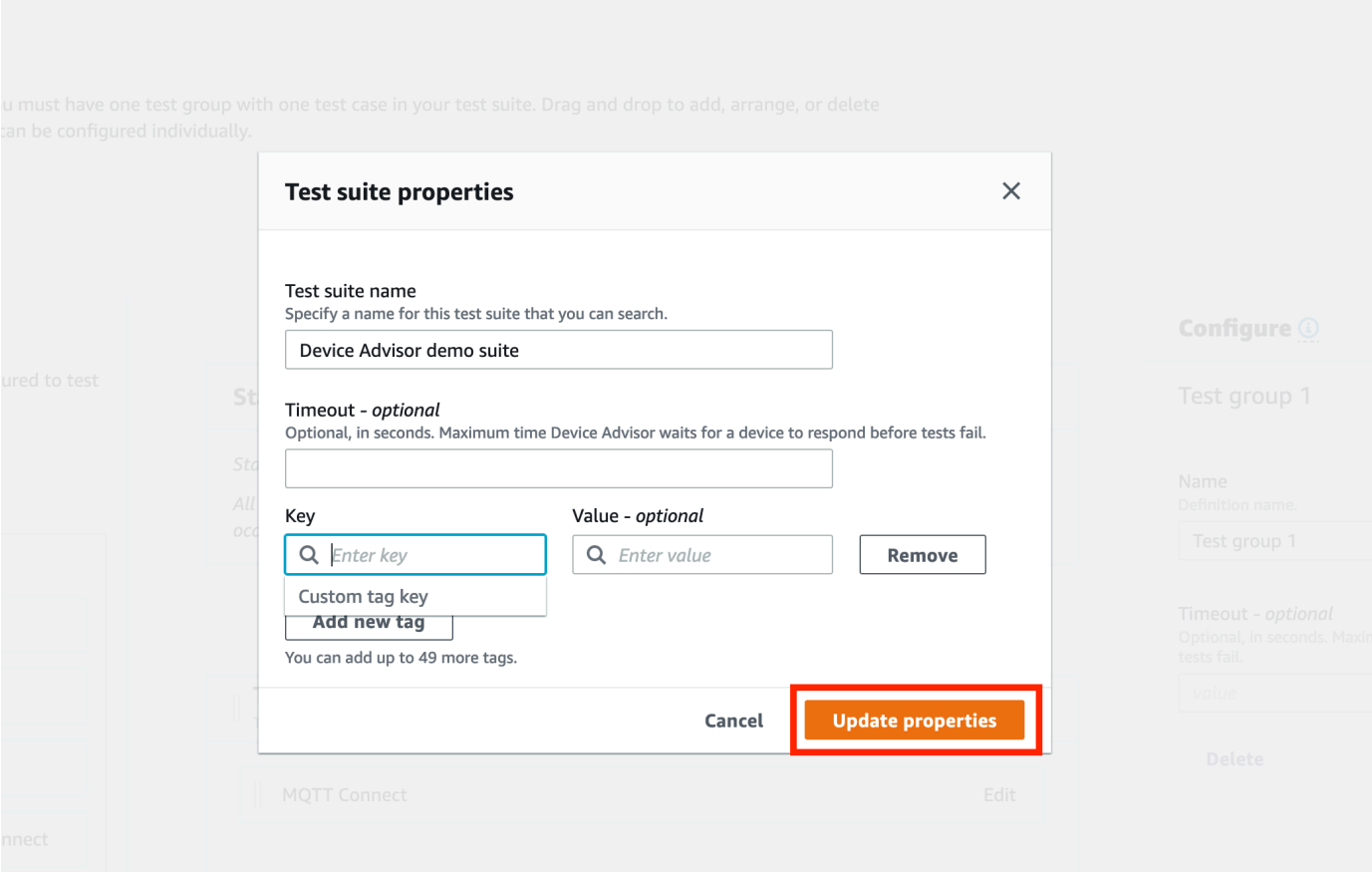


3. Scegli Test suite properties (Proprietà della suite di test). È necessario creare le proprietà della suite di test quando crei la suite di test.



Sotto Test suite properties (Proprietà della suite di test), compila quanto segue:

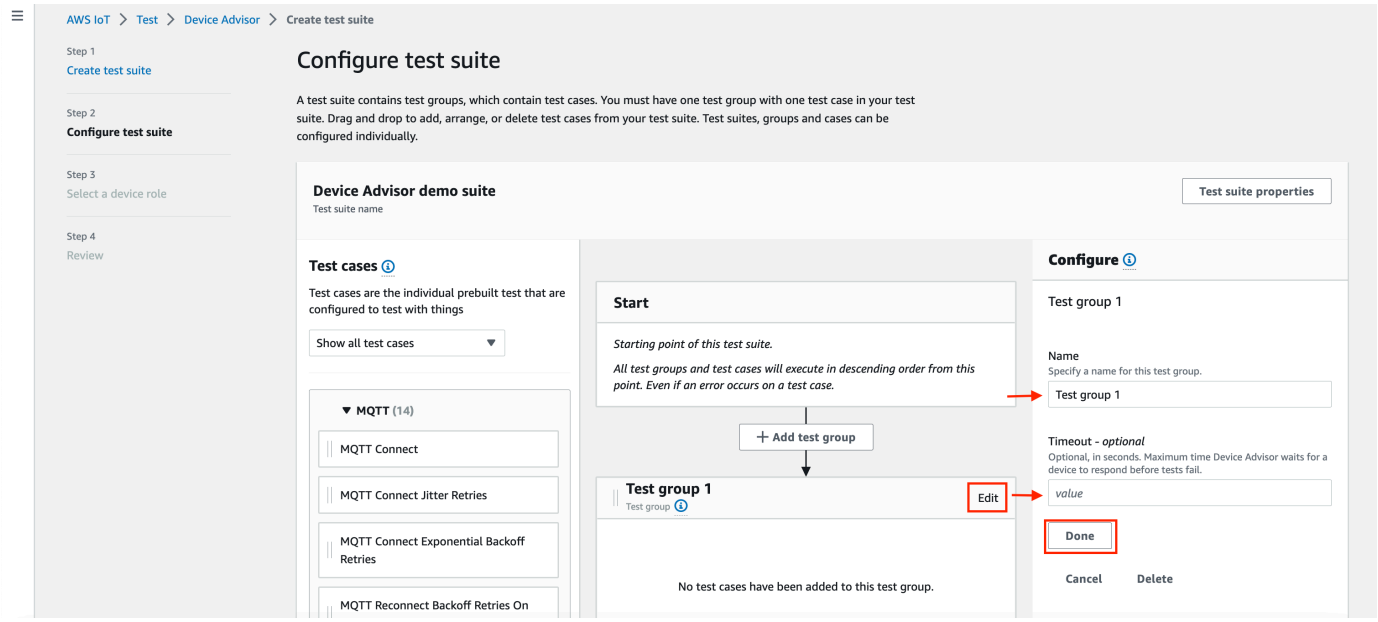
- Test suite name (Nome della suite di test): è possibile creare la suite con un nome personalizzato.
- Timeout (facoltativo): il timeout in secondi per ogni test case nella suite di test corrente. Se non specifichi un valore di timeout, viene utilizzato il valore predefinito.
- Tags (Tag) (facoltativo): Aggiungi tag alla suite di test.



Al termine, scegli Update properties (Aggiorna proprietà).

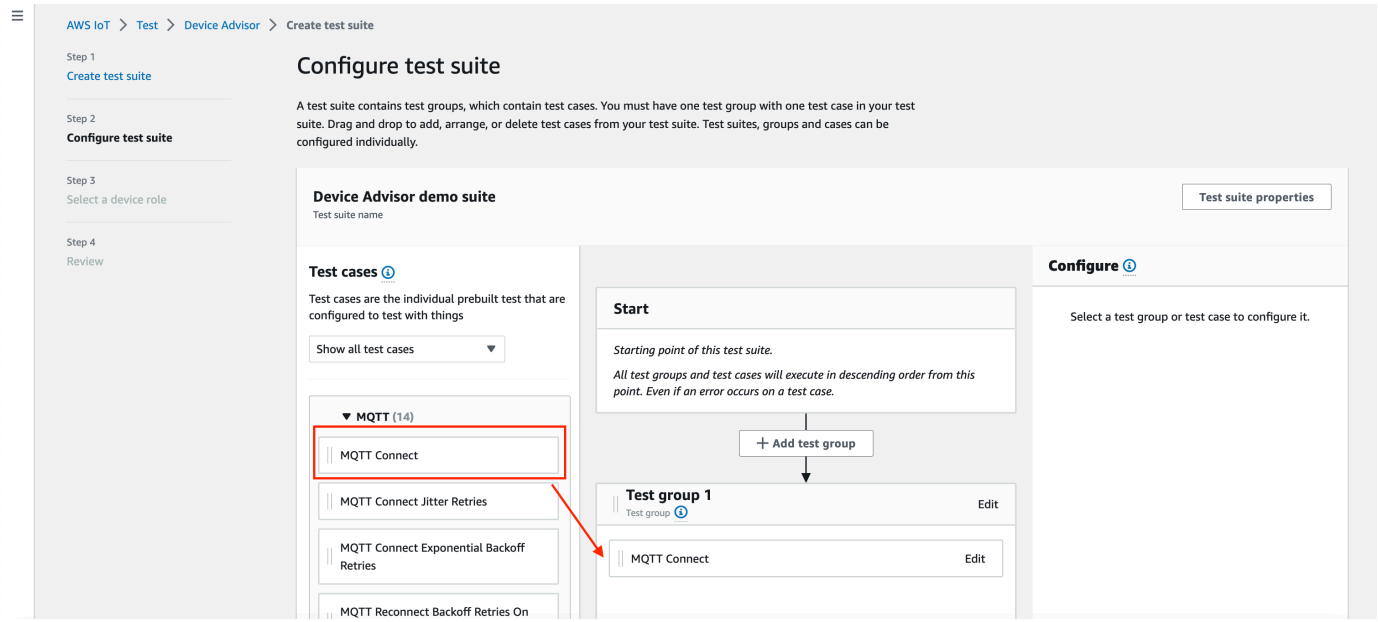
4. Per modificare la configurazione a livello di gruppo, in `Test group 1`, scegli Edit (Modificare). Immetti poi un Name (Nome) per assegnare al gruppo un nome personalizzato.

   Facoltativamente, è anche possibile inserire un valore di Timeout in secondi sotto il gruppo di test selezionato. Se non specifichi un valore di timeout, viene utilizzato il valore predefinito.



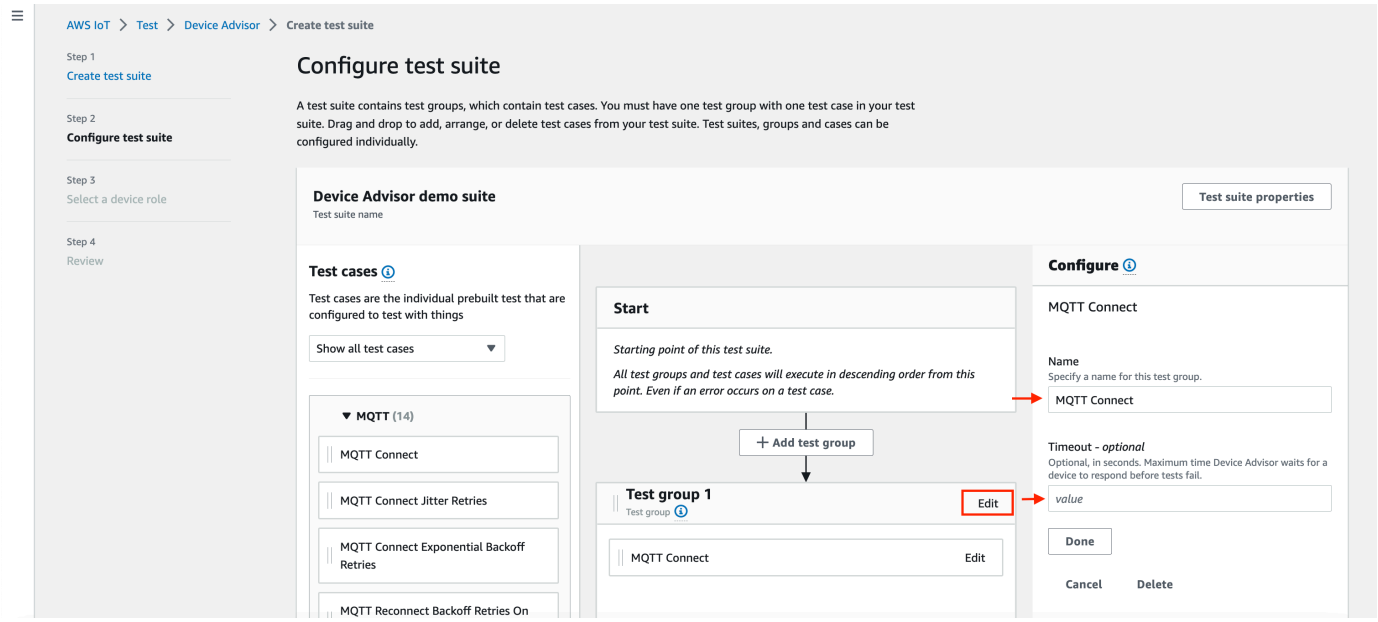
Seleziona Fatto.

5. Trascina uno dei test case disponibili da Test cases (Test case) nel gruppo di test.



6. Per modificare la configurazione a livello di test case per il test case aggiunto nel gruppo di test, scegli Edit (Modifica). Immetti poi un Name (Nome) per assegnare al gruppo un nome personalizzato.

   Facoltativamente, è anche possibile inserire un valore di Timeout in secondi sotto il gruppo di test selezionato. Se non specifichi un valore di timeout, viene utilizzato il valore predefinito.



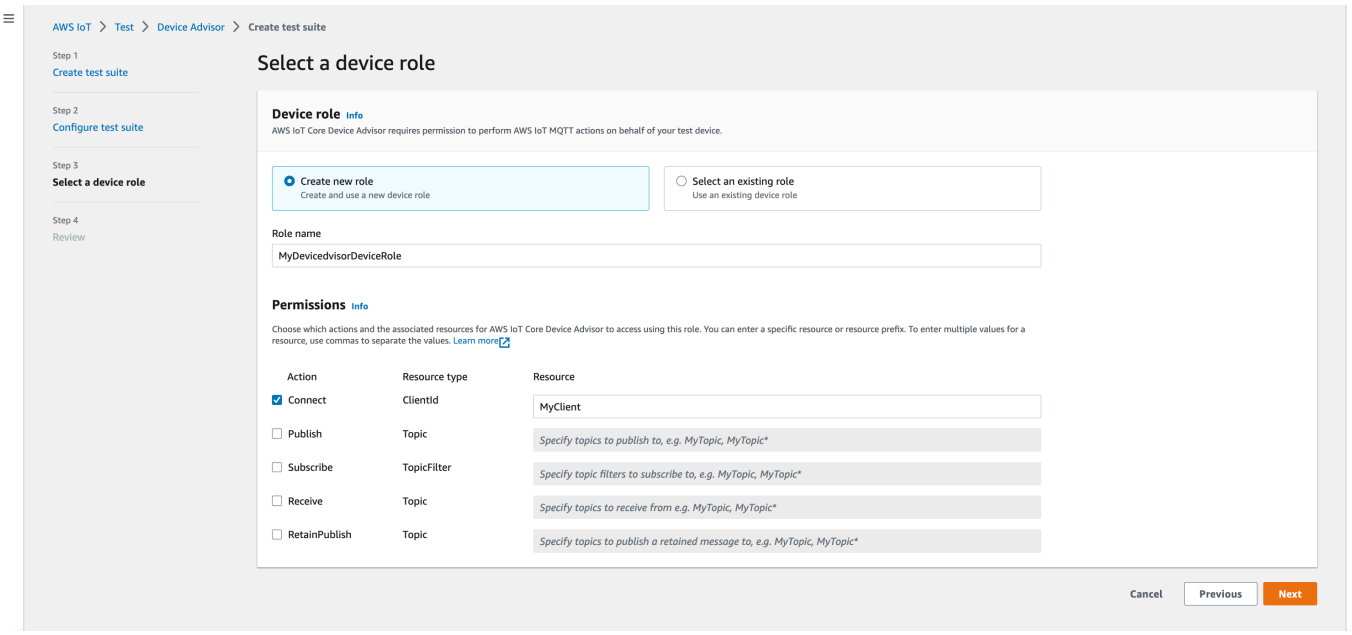
Seleziona Fatto.

> **Note**
>
> Per aggiungere altri gruppi di test alla suite di test, scegli Add test group (Aggiungi gruppo di test). Segui i passaggi precedenti per creare e configurare più gruppi di test o aggiungere più test case a uno o più gruppi di test. I gruppi di test e i test case possono essere riordinati scegliendo e trascinando un test case nella posizione desiderata. Device Advisor esegue i test nell'ordine in cui vengono definiti i gruppi di test e i test case.

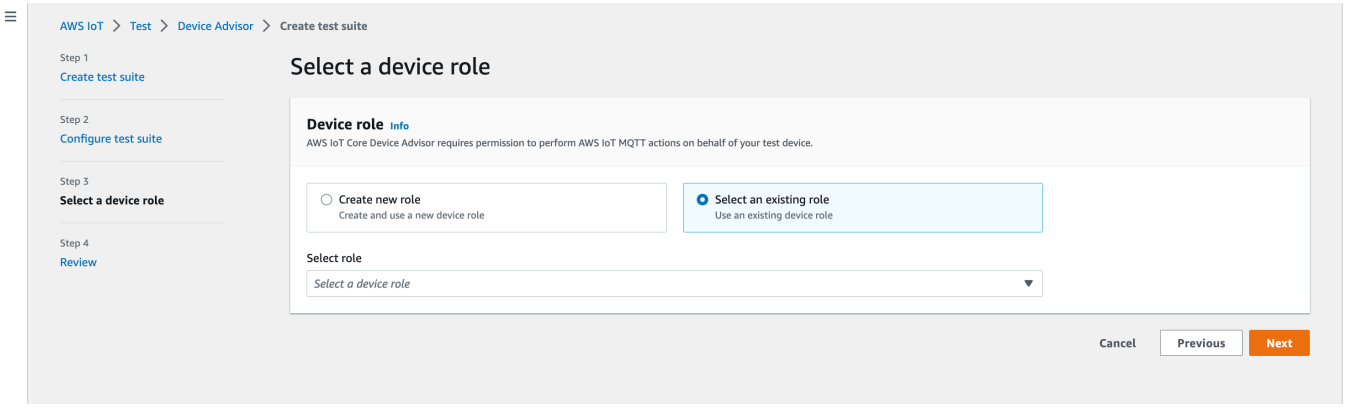
7. Scegli Next (Successivo).
8. Nella fase 3, configura un ruolo del dispositivo che Device Advisor utilizzerà per eseguire azioni AWS IoT MQTT per conto del dispositivo di test.

   Se hai selezionato il test case MQTT Connect solo in Step 2 (Fase 2), l'operazione Connect (Collegarsi) verrà controllata automaticamente poiché è richiesta l'autorizzazione sul ruolo del dispositivo per eseguire questa suite di test. Se hai selezionato altri test case, verranno controllate le operazioni richieste corrispondenti. Assicurarsi che siano forniti i valori delle risorse per ciascuna operazione. Ad esempio, per l'operazione Connect (Collegarsi), fornisci l'ID client con cui il dispositivo si connetterà all'endpoint Device Advisor. È possibile fornire più valori utilizzando virgole per separare i valori e fornire valori di prefisso utilizzando anche un carattere

jolly (*). Ad esempio, per fornire l'autorizzazione per la pubblicazione su qualsiasi argomento che inizia con `MyTopic`, è possibile fornire "`MyTopic*`" come valore della risorsa.



Se hai già creato un ruolo del dispositivo in precedenza e desideri utilizzare tale ruolo, seleziona Select an existing role (Seleziona un ruolo esistente) e scegli il ruolo del tuo dispositivo in Select role (Seleziona ruolo).



Configura il ruolo del dispositivo utilizzando una delle due opzioni fornite e scegli Next (Successivo).

9. In Step 4 (Fase 4), assicurati che la configurazione fornita in ciascuno dei passaggi sia accurata. Per modificare la configurazione fornita per un particolare passaggio, scegli Edit (Modifica) per il passaggio corrispondente.

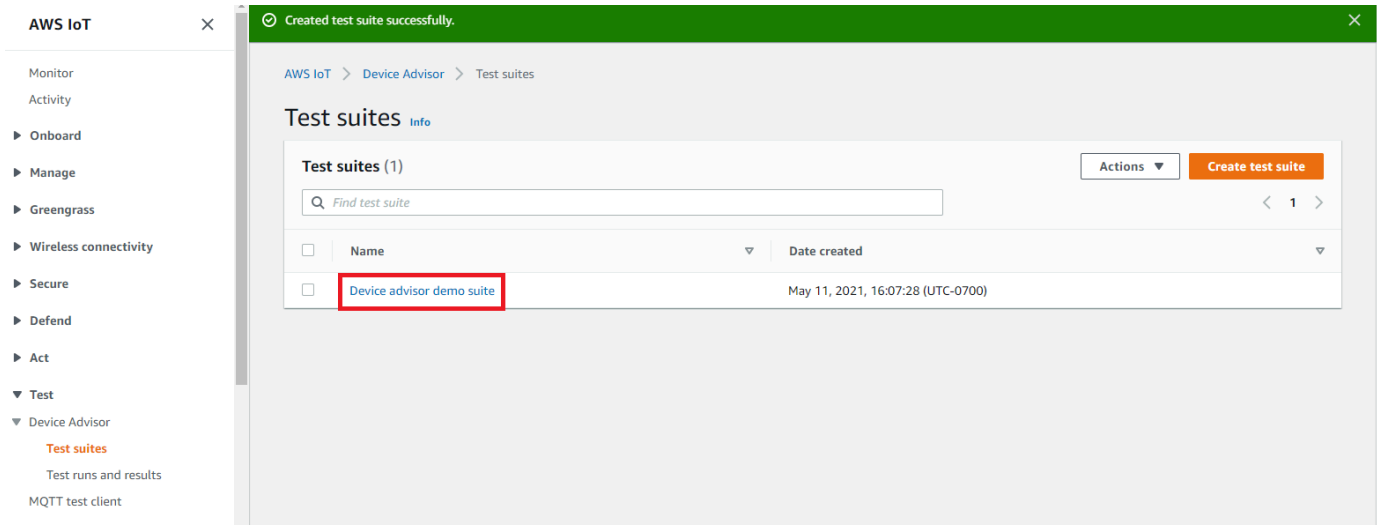
   Dopo aver verificato la configurazione, scegli Create test suite (Crea suite di test).

La suite di test dovrebbe essere creata con successo e verrai reindirizzato alle Test suites (Suite di test) dove è possibile visualizzare tutte le suite di test che sono state create.

Se la creazione della suite di test non è riuscita, assicurati che la suite di test, i gruppi di test, i test case e ruolo del dispositivo siano stati configurati in base alle istruzioni.

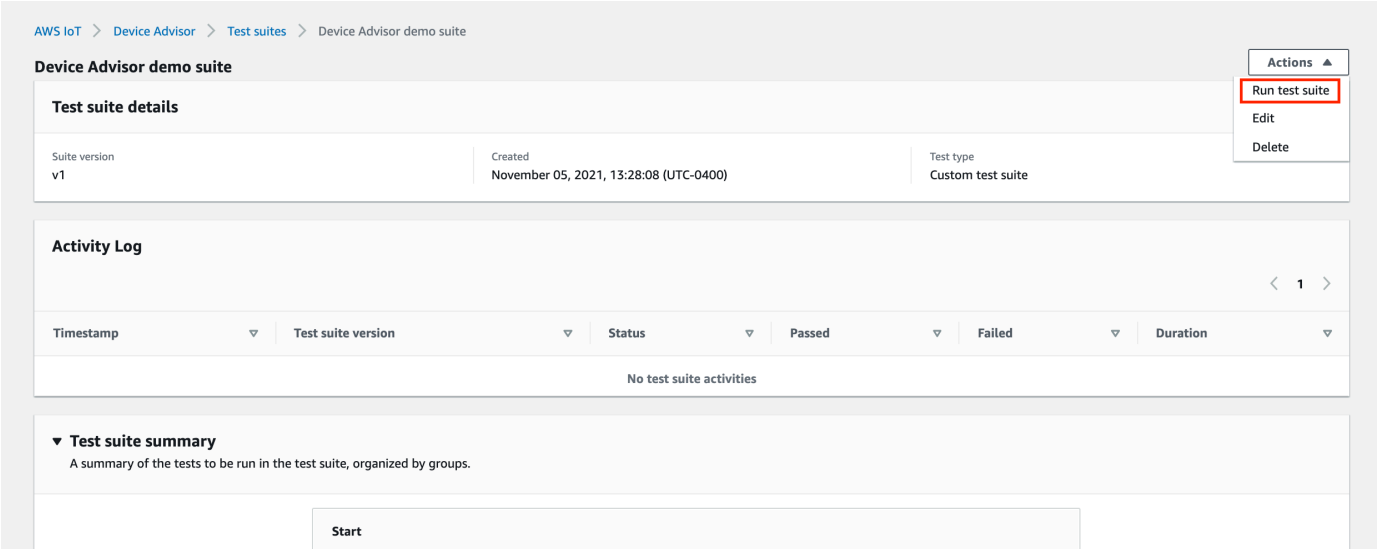
Avvia una suite di test

1. Nell'AWS IoT console, nel pannello di navigazione, espandi Test, Device Advisor, quindi scegli Test suites (Suite di test).
2. Scegli la suite di test per la quale desideri visualizzare i dettagli.



La pagina dei dettagli della suite di test mostra tutte le informazioni relative alla suite di test.

3. Scegli Actions (Operazioni), quindi Run test suite (Esegui suite di test).

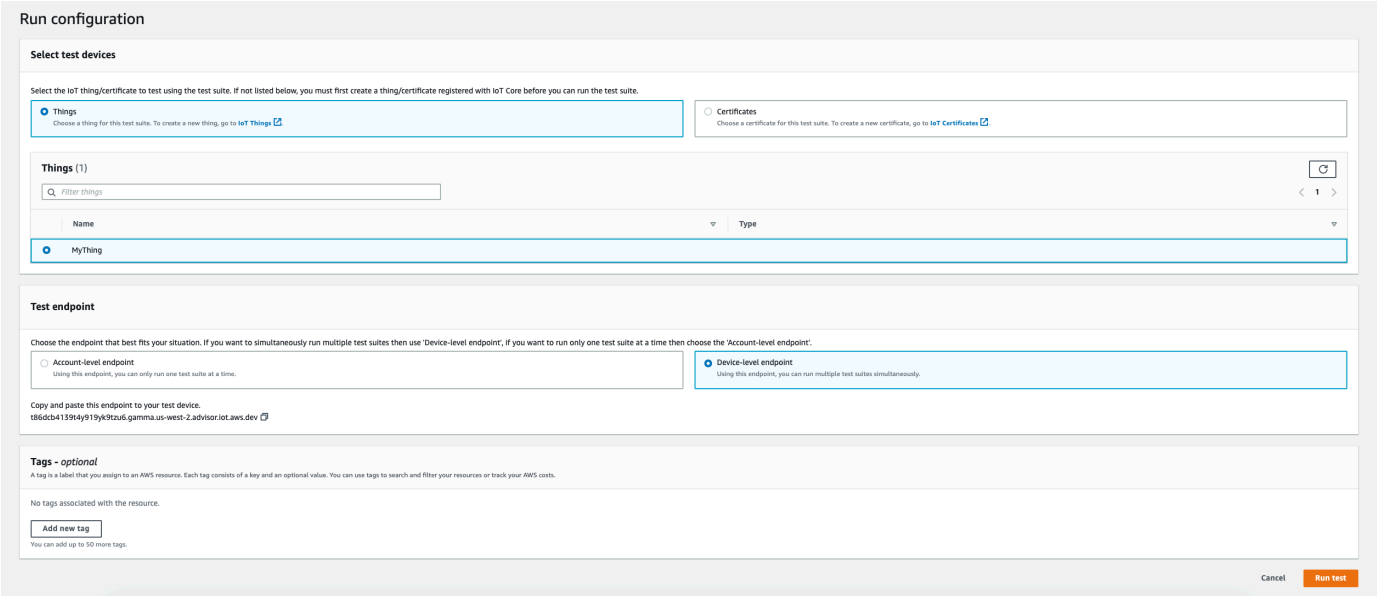


4. In Run configuration, dovrai selezionare un AWS IoT elemento o un certificato da testare utilizzando Device Advisor. Se non disponi di elementi o certificati esistenti, [crea innanzitutto AWS IoT Core delle risorse](#).

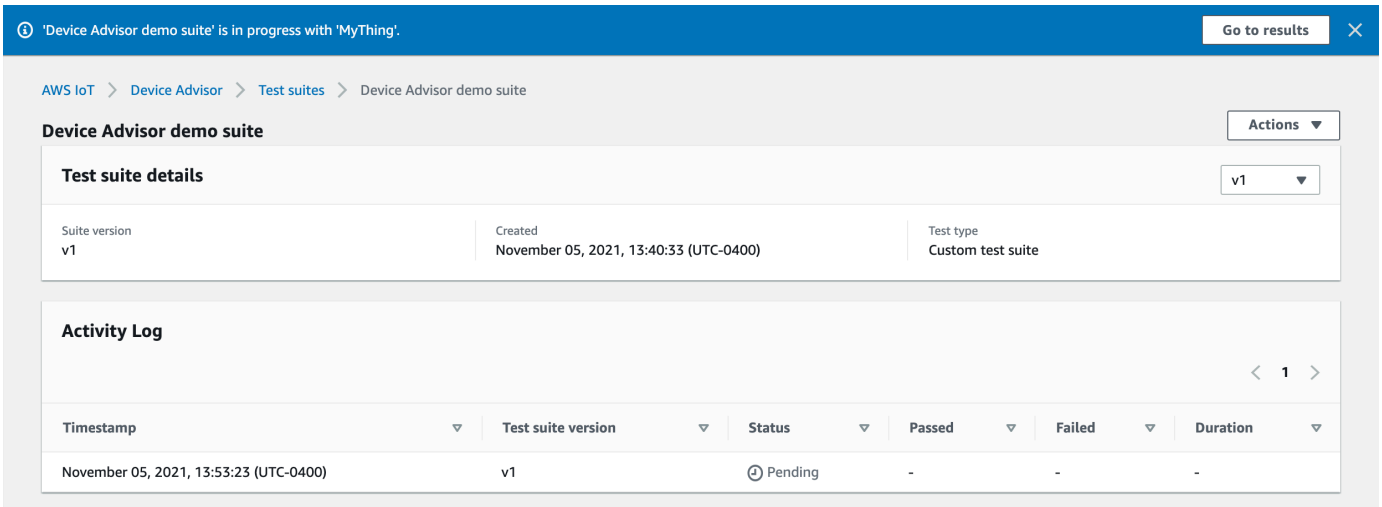
   Nella sezione Test endpoint (Endpoint di test) seleziona l'endpoint più adatto al tuo caso. Se prevedi di eseguire più suite di test contemporaneamente utilizzando lo stesso AWS account in futuro, seleziona Endpoint a livello di dispositivo. In caso contrario, se prevedi di eseguire una sola suite di test alla volta, seleziona Account-level endpoint (Endpoint a livello di account).

   Configurare il dispositivo di test con l'endpoint di test del Device Advisor selezionato.

   Dopo aver selezionato un oggetto o un certificato e aver scelto un endpoint Device Advisor, scegli Run test (Esegui test).

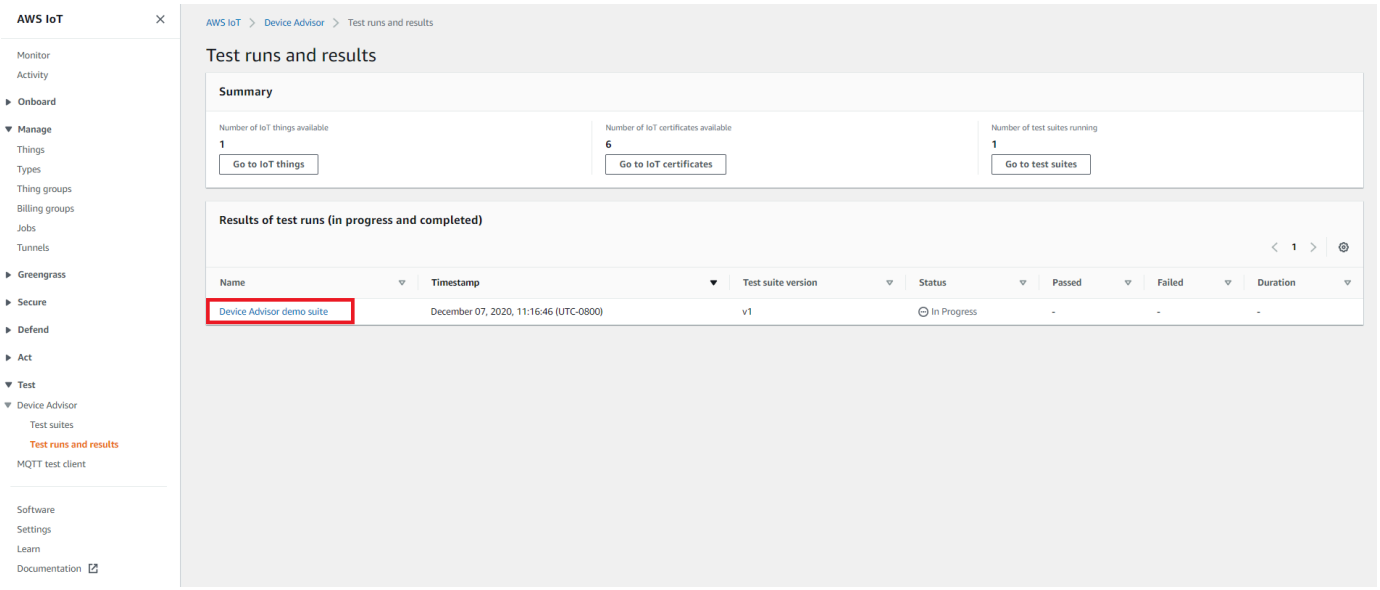


5. Scegli Go to results (Accedi ai risultati) nel banner superiore per visualizzare i dettagli dell'esecuzione del test.

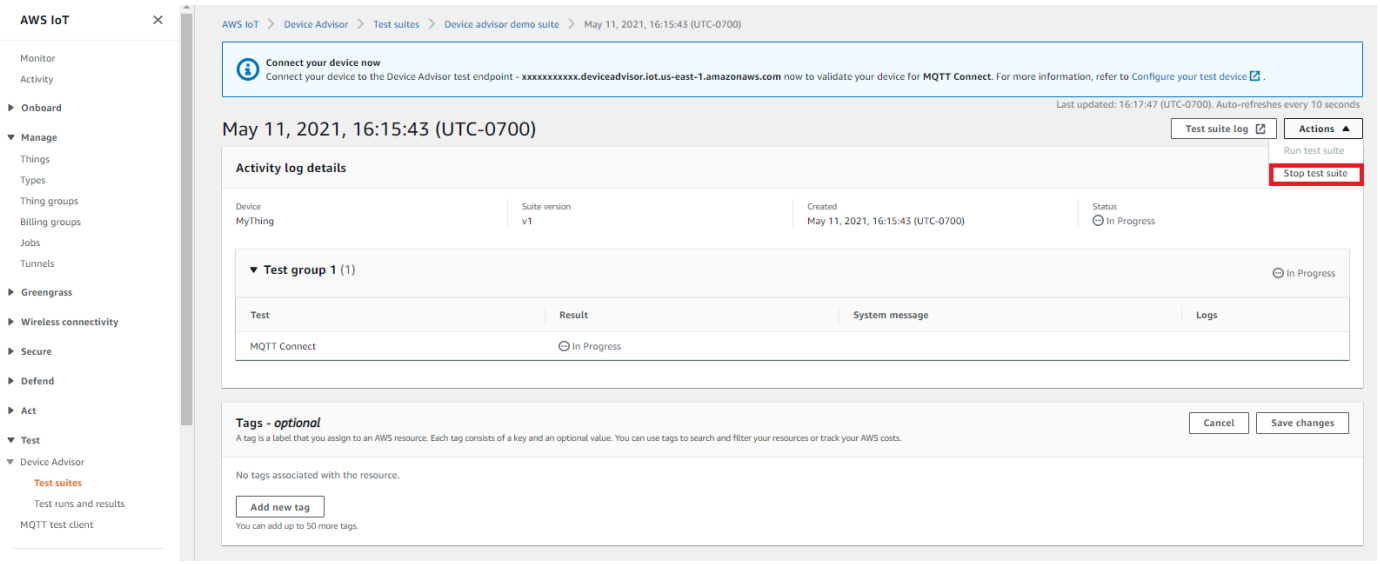


Interrompi l'esecuzione di una suite di test (facoltativo)

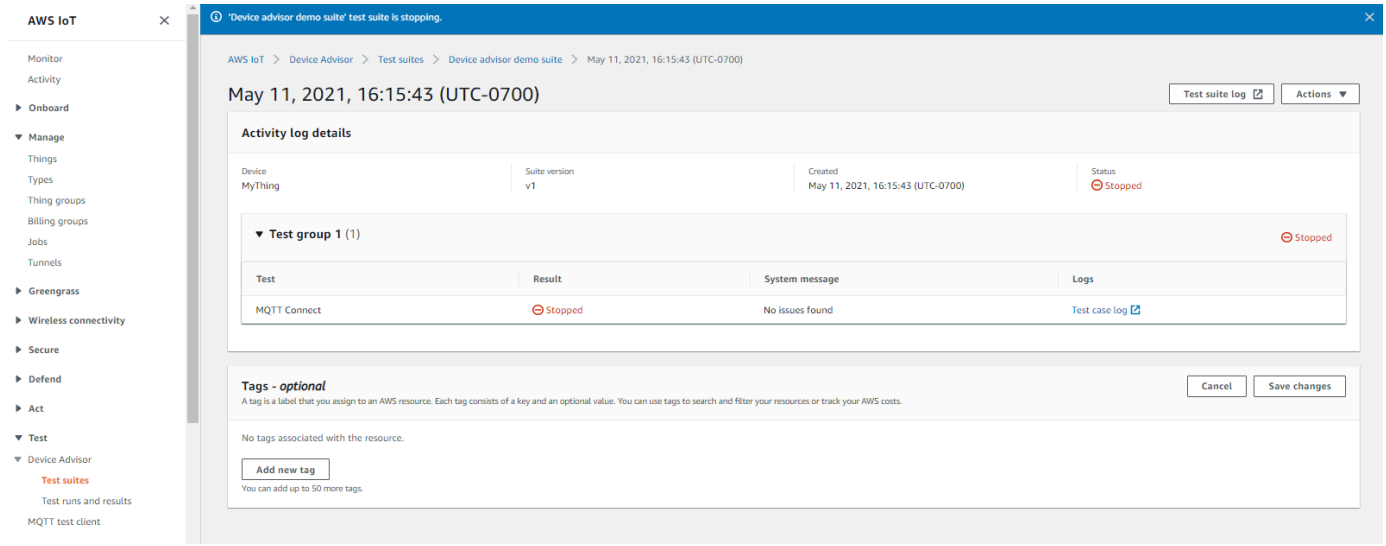
1. Nell'AWS IoT console, nel pannello di navigazione, espandi Test, Device Advisor, quindi scegli Test runs and results (Esecuzioni di test e risultati).
2. Scegli la suite di test in corso che desideri arrestare.



3. Scegli Actions (Operazioni), quindi Stop test suite (Arresto delle suite di test).



4. Il processo di pulizia può richiedere alcuni minuti per il completamento. Mentre il processo di pulizia è in corso, lo stato di esecuzione del test sarà `STOPPING`. Attendi che il processo di pulizia sia completo e che lo stato della suite di test passi a `STOPPED` prima di iniziare una nuova esecuzione della suite.

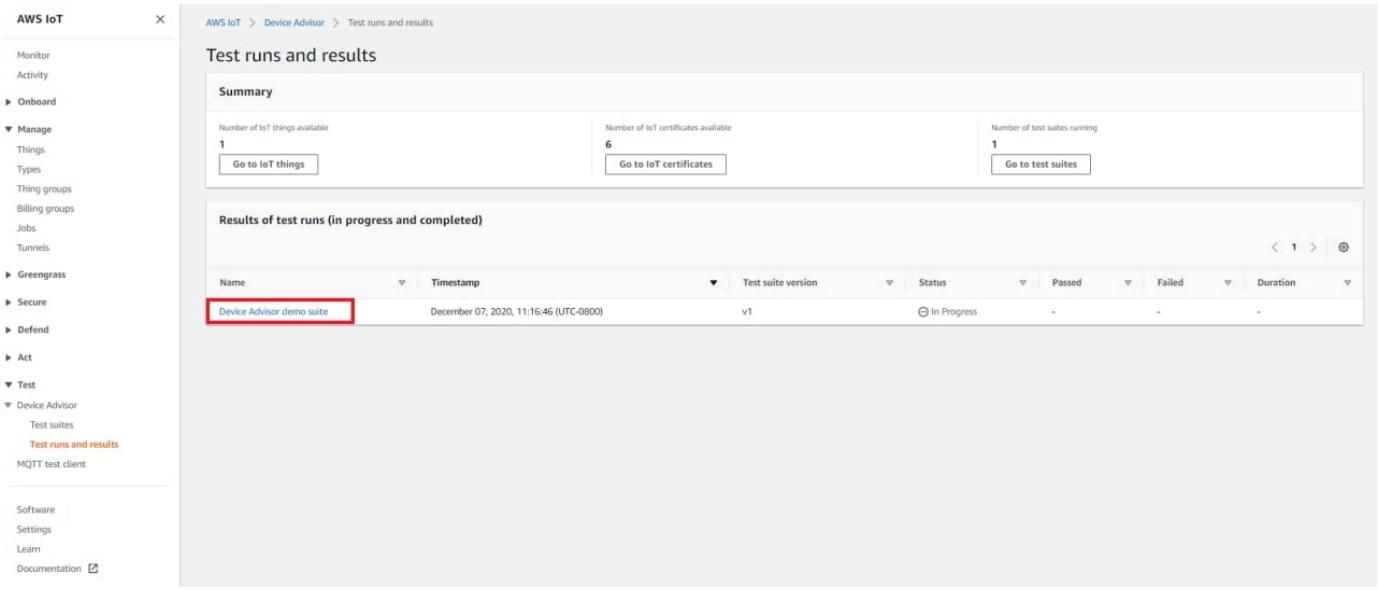


Visualizza i dettagli e i registri di esecuzione della suite

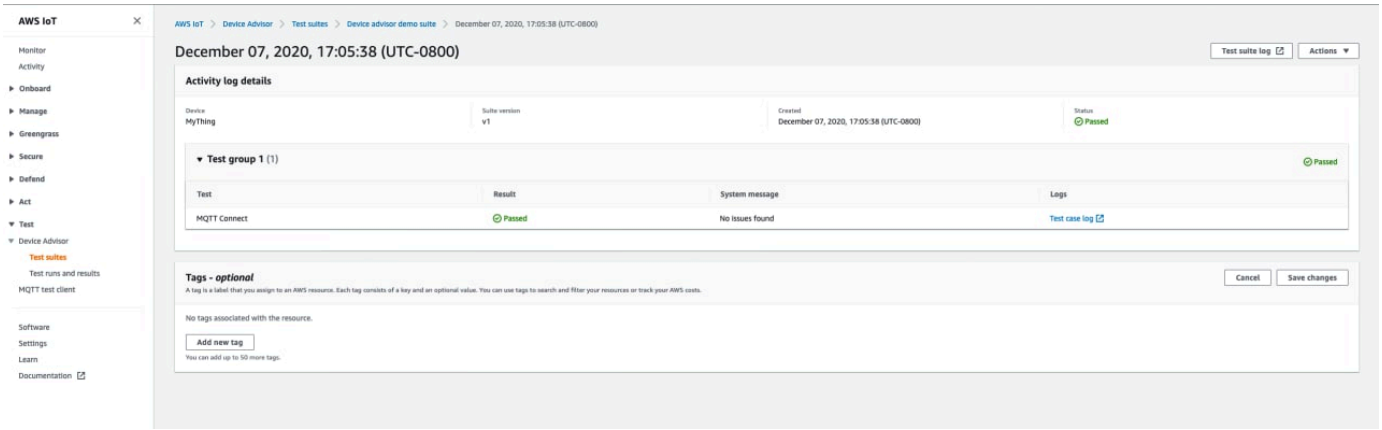
1. Nell'AWS IoT console, nel pannello di navigazione, espandi Test, Device Advisor quindi scegli Test runs and results (Esecuzioni di test e risultati).

   Questa pagina mostra:

   - Numero di oggetti IoT
   - Numero di certificati IoT
   - Numero di suite di test attualmente in esecuzione
   - Tutte le esecuzioni della suite di test che sono state create
2. Scegli la suite di test per la quale desideri visualizzare i dettagli e registri di esecuzione.



La pagina di riepilogo dell'esecuzione visualizza lo stato dell'esecuzione corrente della suite di test. Questa pagina viene aggiornata automaticamente ogni 10 secondi. Si consiglia di disporre di un meccanismo creato per il dispositivo per provare a connettersi al nostro endpoint di test ogni cinque secondi per uno o due minuti. Dopodiché puoi eseguire più casi di test in sequenza in modo automatico.



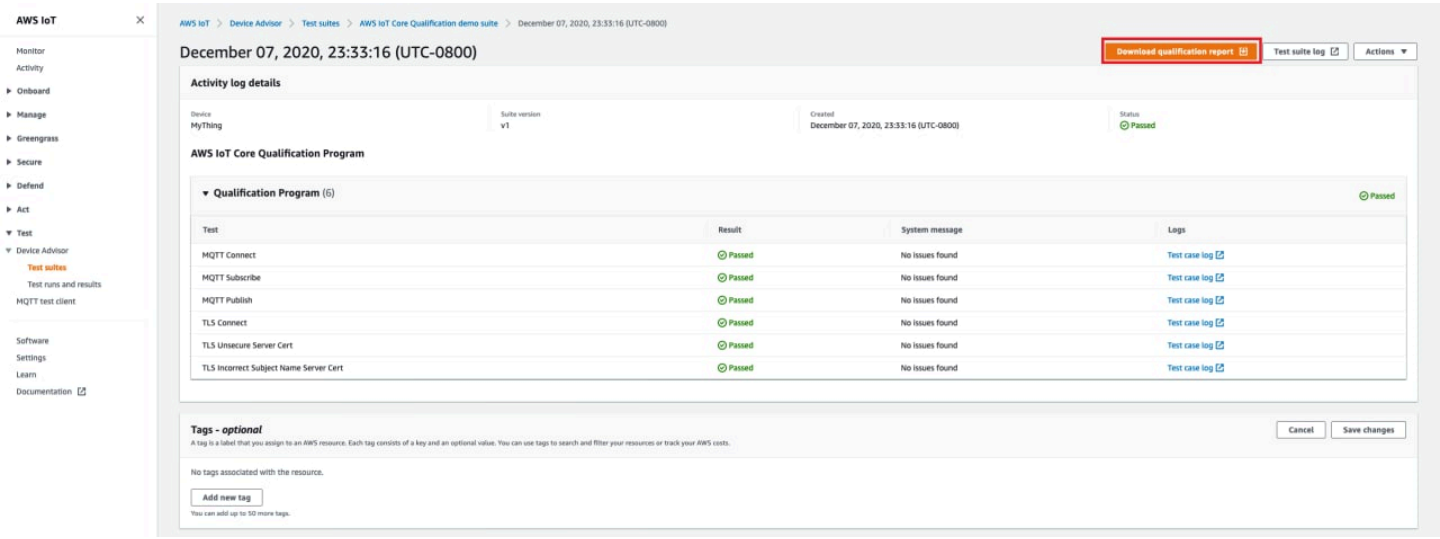
3. Per accedere ai CloudWatch log relativi all'esecuzione della suite di test, scegliete Test suite log.

   Per accedere ai CloudWatch log di qualsiasi test case, scegli Test case log.

4. In base ai risultati del test, risolvi i problemi del tuo dispositivo fino a quando tutti i test vengono superati.

Scarica un rapporto di AWS IoT qualificazione

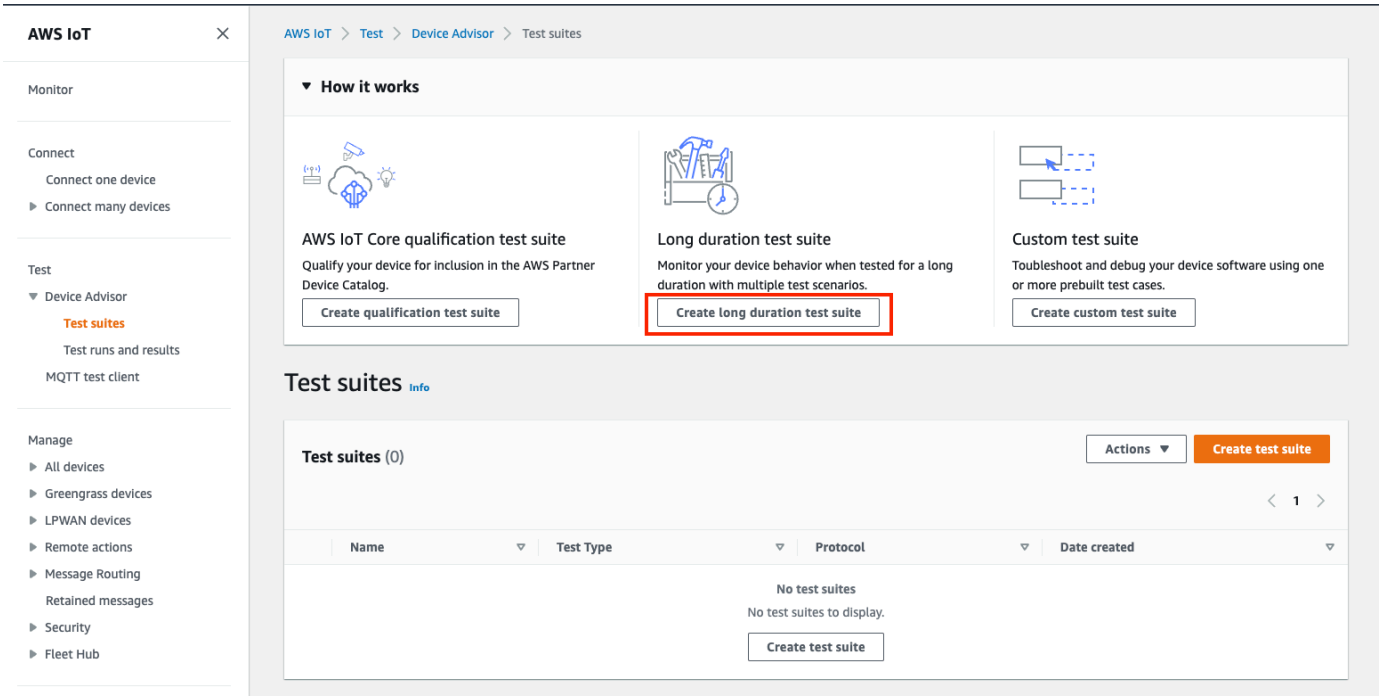
Se hai scelto l'opzione Usa la suite di test di AWS IoT qualificazione durante la creazione di una suite di test e sei riuscito a eseguire una suite di test di qualificazione, puoi scaricare un rapporto di qualificazione selezionando Scarica rapporto di qualificazione nella pagina di riepilogo dell'esecuzione del test.



Flusso di lavoro della console per test di lunga durata

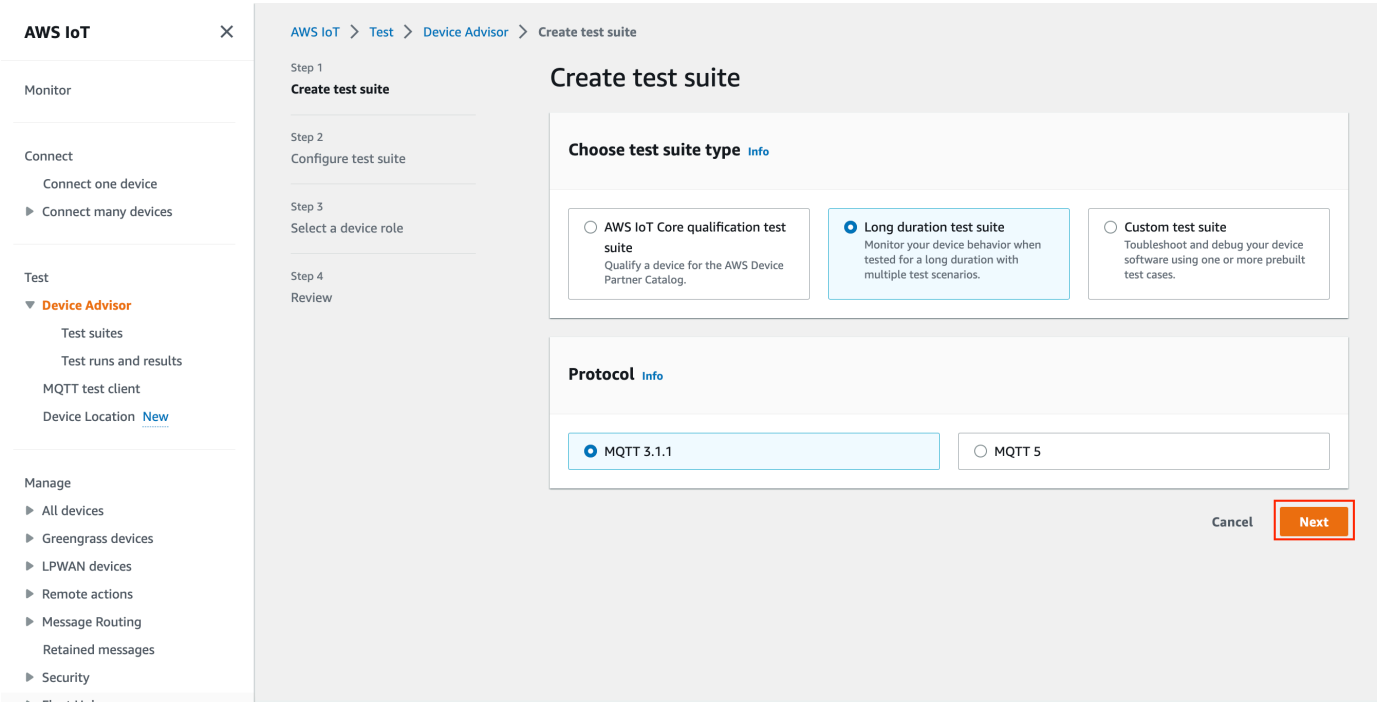
Questo tutorial consente di iniziare a utilizzare test di lunga durata su Device Advisor utilizzando la console. Per completare il tutorial, attenersi alle fasi descritte in Configurazione.

1. Nel pannello di navigazione Console AWS IoT, espandi Test, Device Advisor, quindi Test suites (Suite di test). Nella pagina, seleziona Create long duration test suite (Crea suite di test di lunga durata).



2. Nella pagina Create test suite (Crea suite di test), seleziona Long duration test suite (Suite di test di lunga durata) e scegli Next (Avanti).

   Per il protocollo, scegli MQTT 3.1.1 o MQTT 5.



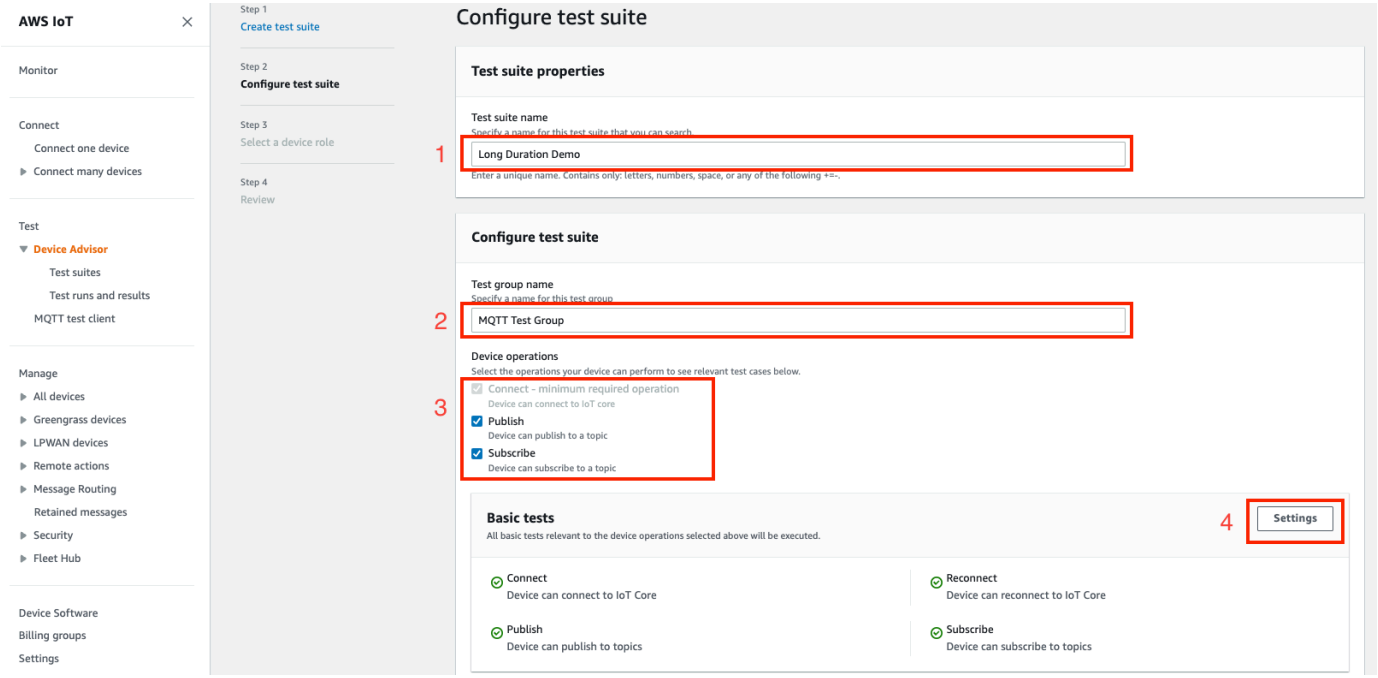
3. Nella pagina Configure test suite (Configura la suite di test), esegui le seguenti operazioni:

   a. Aggiorna il campo Test suite name (Nome suite di test).

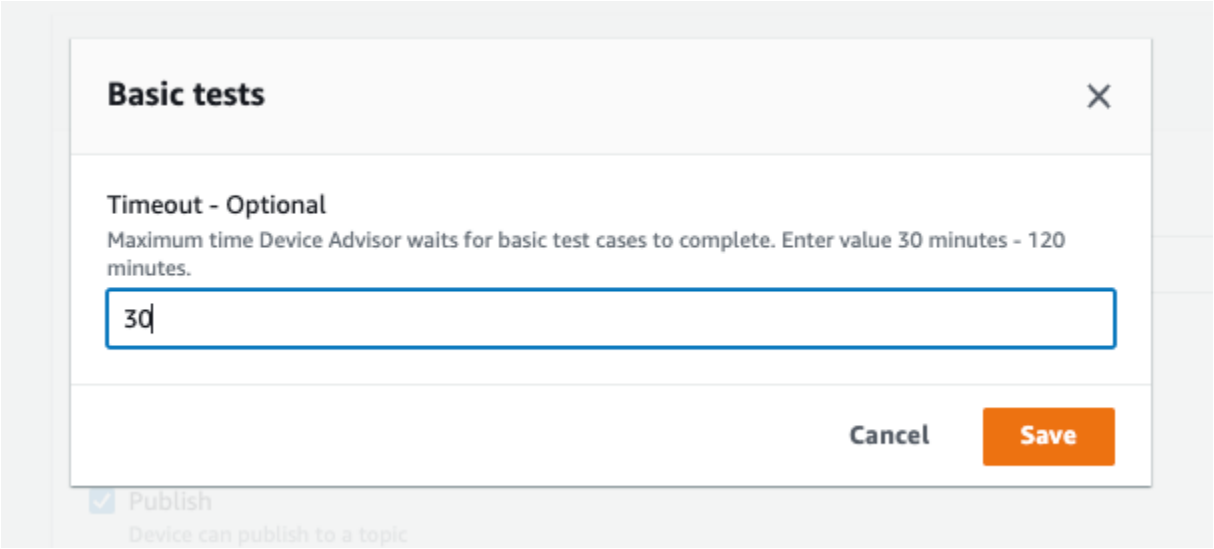
b. Aggiorna il campo Test group name (Nome del gruppo di test).

c. Scegli le Device operations (Operazioni del dispositivo) che il dispositivo può eseguire. Verranno selezionati i test da eseguire.

d. Seleziona l'opzione Settings (Impostazioni).



4. (Facoltativo) Inserisci il periodo di tempo massimo per cui Device Advisor deve attendere il completamento dei test di base. Seleziona Salva.



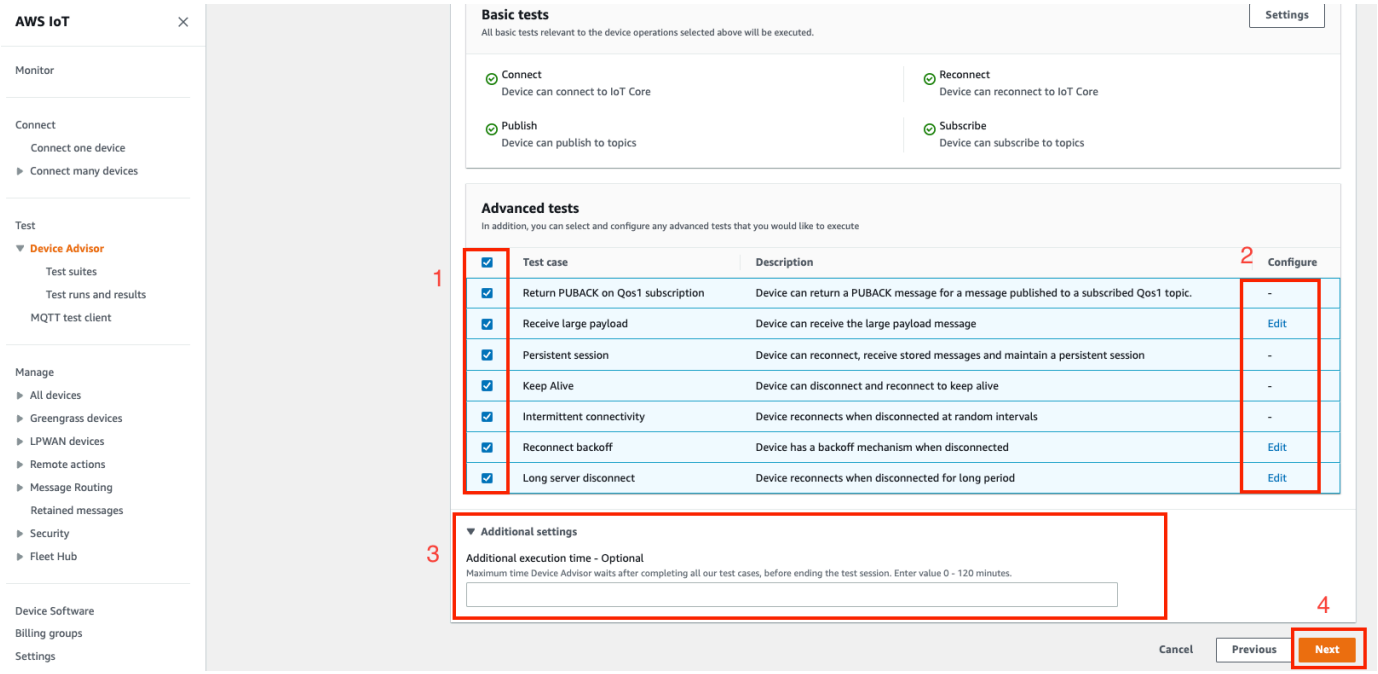
5. Esegui le seguenti operazioni nelle sezioni Advanced tests (Test avanzati) e Additional settings (Impostazioni aggiuntive).

a. Seleziona o deseleziona Advanced tests (Test avanzati) che desideri eseguire come parte di questo test.

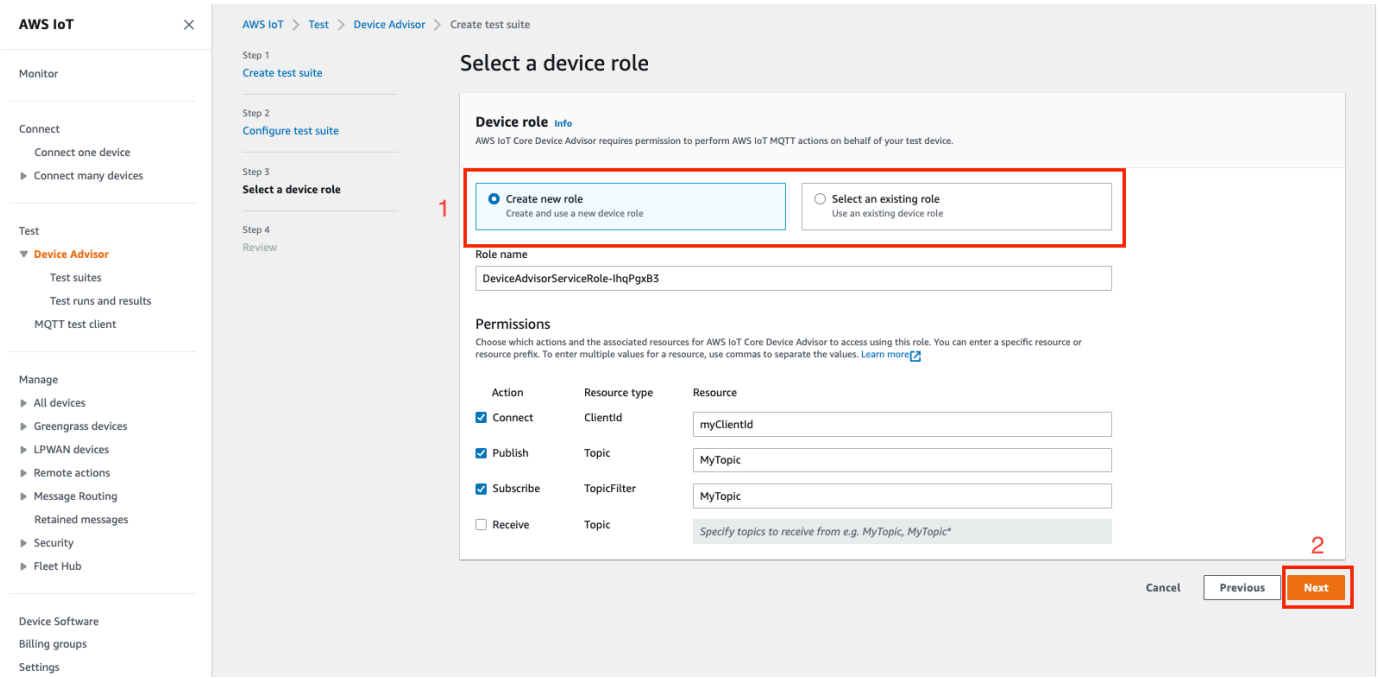
b. Modifica le configurazioni per i test, se necessario.

c. Configura l'opzione Additional execution time (Tempo di esecuzione aggiuntivo) nella sezione Additional settings (Impostazioni aggiuntive).

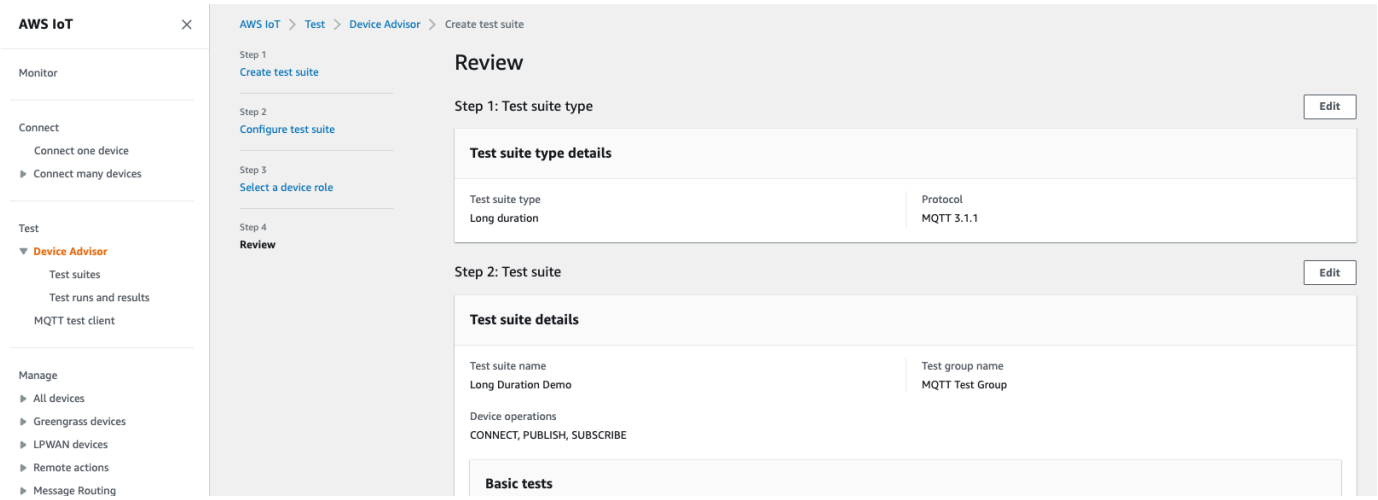
d. Scegli Next (Avanti) per eseguire il passaggio successivo.

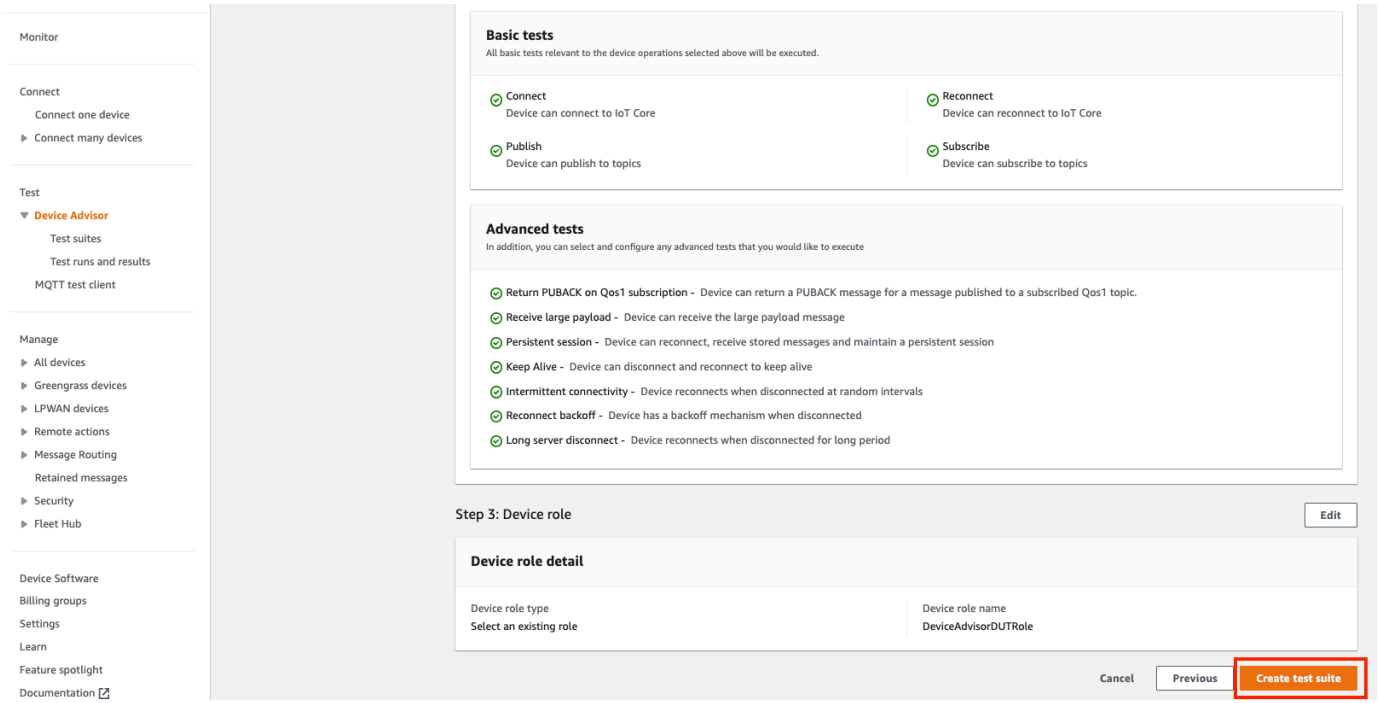


6. In questo passaggio, seleziona Create a new role (Crea un nuovo ruolo) o Select an existing role (Seleziona un ruolo esistente). Per informazioni dettagliate, vedi Creazione di un ruolo IAM da utilizzare come ruolo del dispositivo.

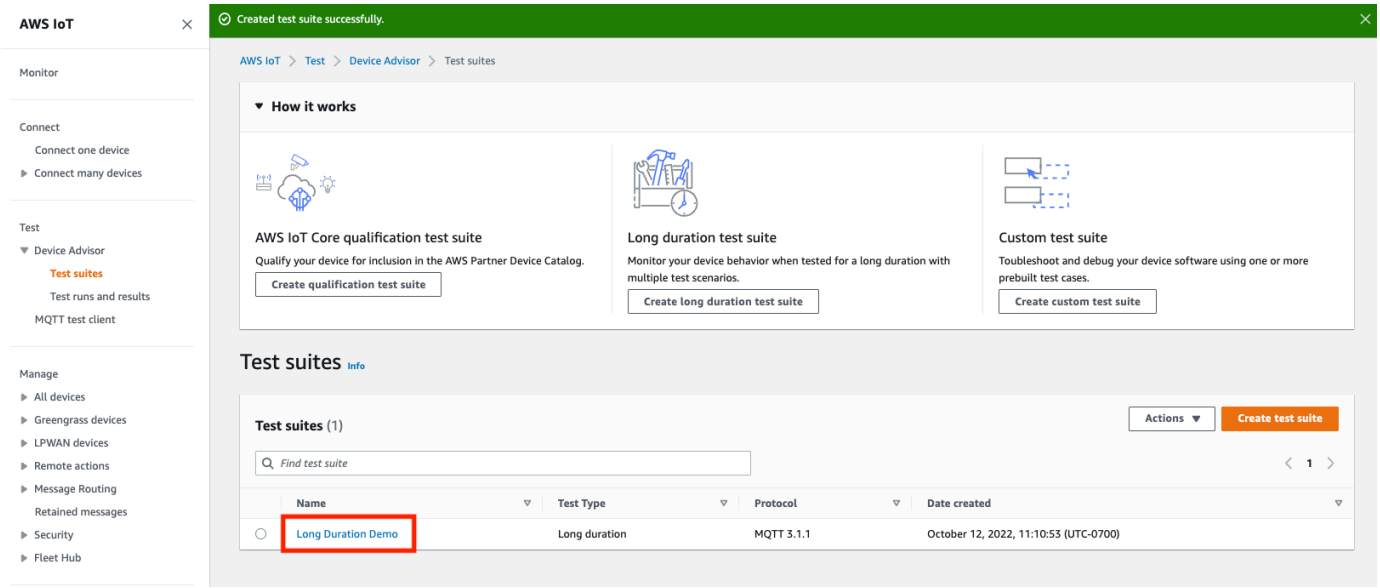


7. Esamina tutte le configurazioni create fino a questo passaggio e seleziona Create test suite (Crea suite di test).

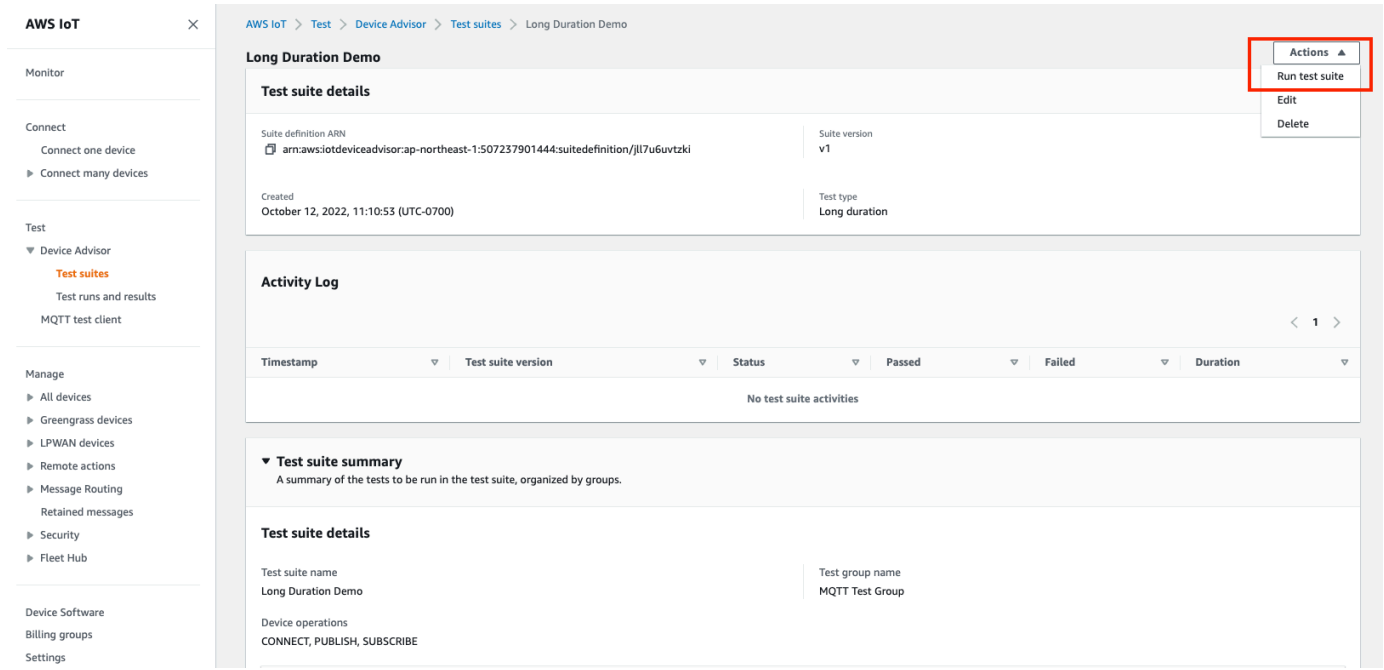




8. La suite di test creata si trova nella sezione Test suites (Suite di test). Seleziona la suite per visualizzare i dettagli.



9. Per eseguire la suite di test creata, seleziona Actions (Operazioni), quindi Run test suite (Esegui suite di test).

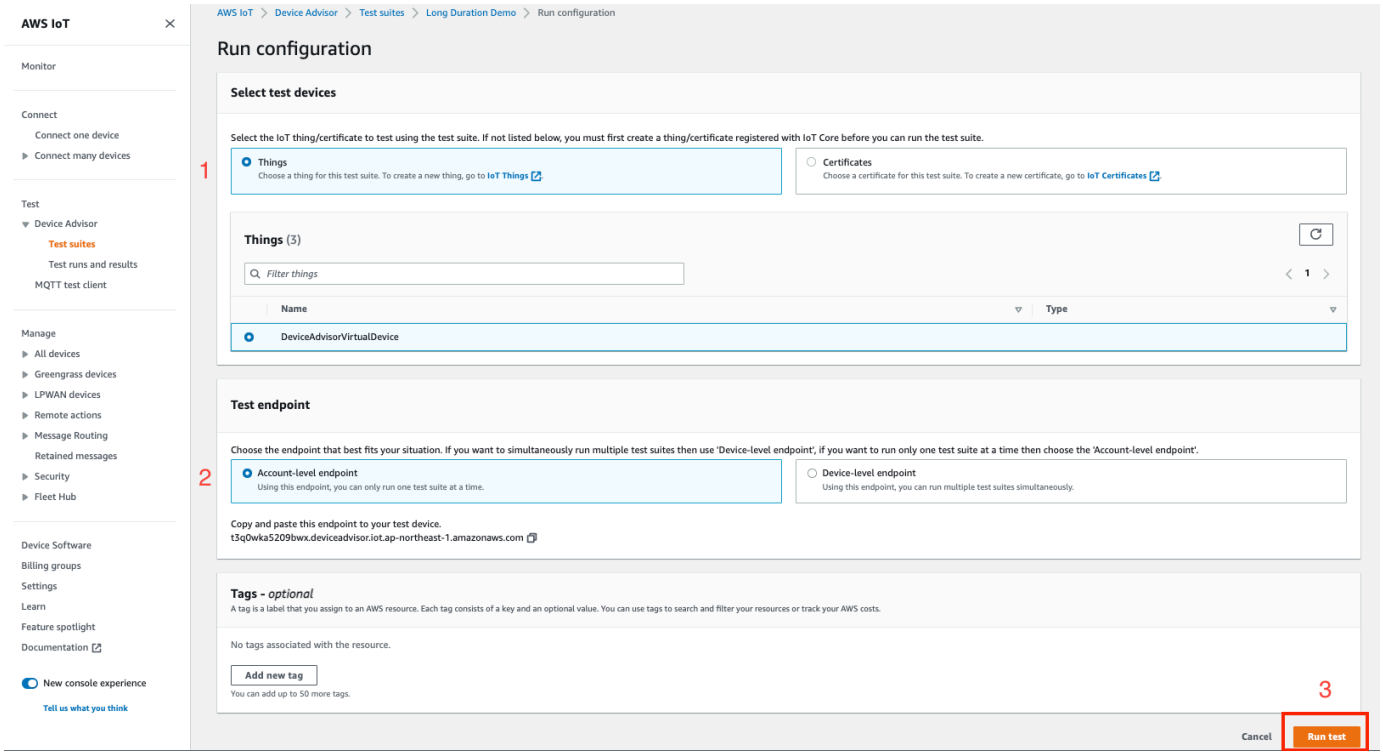


10. Scegli le opzioni di configurazione nella pagina Run configuration (Esegui configurazione).

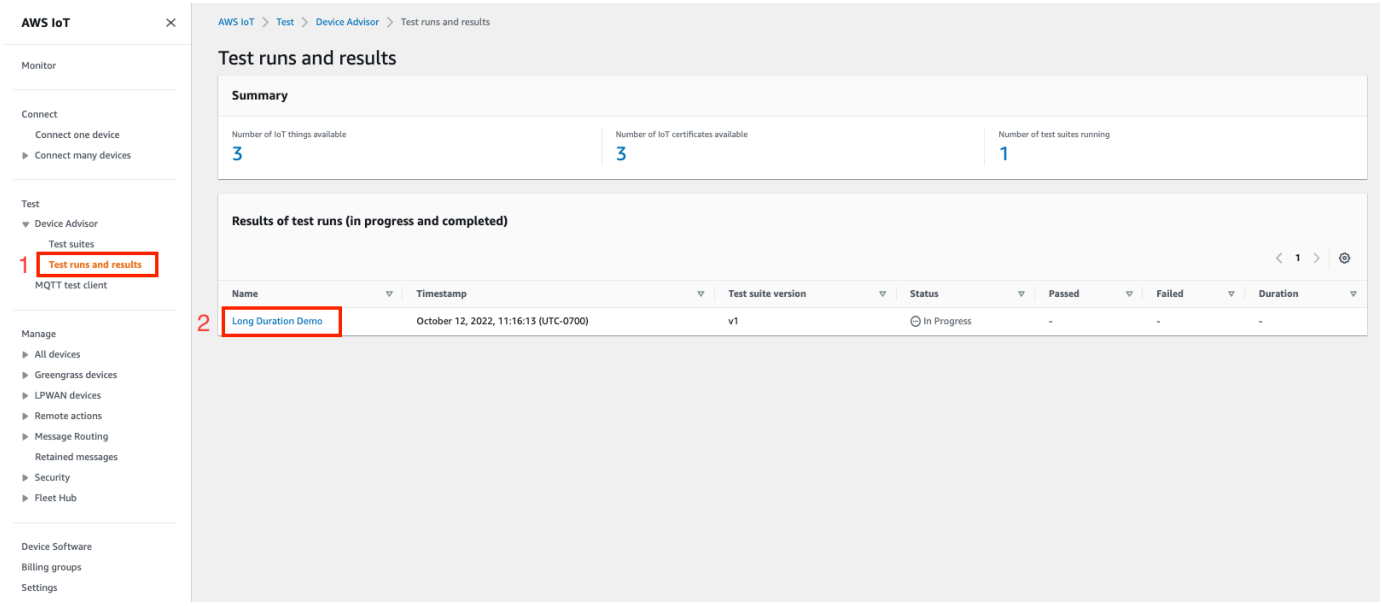
    a. Seleziona Things (Oggetti) o Certificate (Certificato) su cui eseguire il test.

    b. Seleziona Account-level endpoint (Endpoint a livello di account) o Device-level endpoint (Endpoint a livello di dispositivo).

    c. Scegli Run test (Esegui test) per eseguire il test.

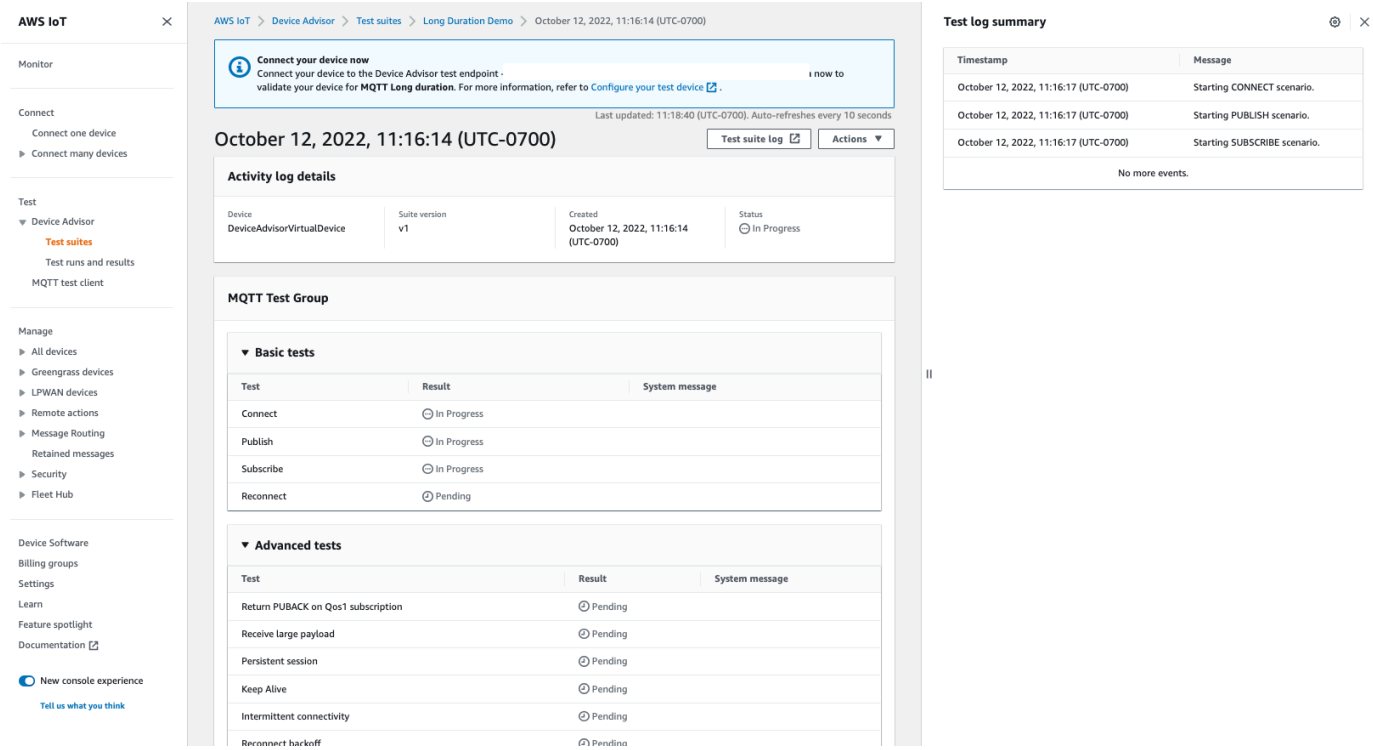


11. Per visualizzare i risultati dell'esecuzione della suite di test, seleziona Test runs and results (Esecuzioni di test e risultati) nel pannello di navigazione a sinistra. Scegli la suite di test che è stata eseguita per visualizzare i dettagli dei risultati.

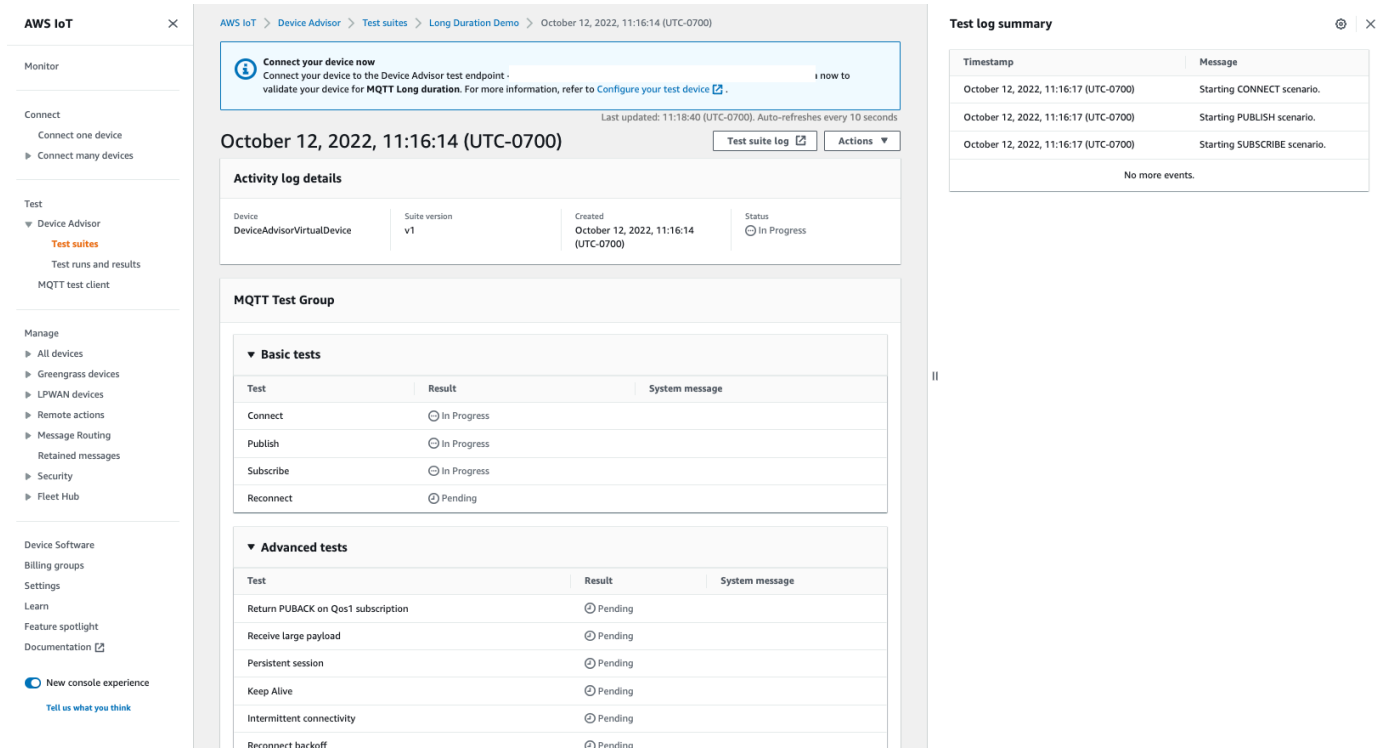


12. Il passaggio precedente visualizza la pagina di riepilogo del test. Tutti i dettagli dell'esecuzione del test vengono visualizzati in questa pagina. Quando la console richiede di avviare la

connessione del dispositivo, connetti il dispositivo all'endpoint fornito. Lo stato di avanzamento dei test è visibile in questa pagina.



13. Il test di lunga durata fornisce un campo Test log summary (Riepilogo del registro di prova) aggiuntivo sul pannello laterale contenente tutti gli eventi importanti che si verificano tra il dispositivo e il broker in tempo quasi reale. Per visualizzare log più dettagliati, fai clic su Test case log (Log del caso di test).



Endpoint VPC di Device Advisor (AWS PrivateLink)

Puoi stabilire una connessione privata tra il tuo VPC e l'endpoint di AWS IoT Core Device Advisor test (piano dati) creando un endpoint VPC di interfaccia. È possibile utilizzare questo endpoint per convalidare AWS IoT i dispositivi per una connettività affidabile e sicura prima di distribuirli in produzione. AWS IoT Core I test predefiniti di Device Advisor consentono di convalidare il software del dispositivo in base alle best practice per l'utilizzo di TLS, MQTT, Device Shadow e Processi AWS IoT.

AWS PrivateLinkalimenta gli endpoint di interfaccia utilizzati con i tuoi dispositivi IoT. Questo servizio consente di accedere privatamente all'endpoint di test AWS IoT Core Device Advisor senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione Direct Connect . Le istanze nel tuo VPC che inviano pacchetti TCP e MQTT non necessitano di indirizzi IP pubblici per comunicare con gli endpoint di test. AWS IoT Core Device Advisor Il traffico tra il tuo VPC e il tuo VPC AWS IoT Core Device Advisor non parte. Cloud AWS Qualsiasi comunicazione TLS e MQTT tra dispositivi IoT e casi di test di Device Advisor rimane all'interno delle risorse in Account AWS.

Ogni endpoint di interfaccia è rappresentato da una o più interfacce di rete elastiche nelle sottoreti.

Per ulteriori informazioni sull'uso degli endpoint VPC dell'interfaccia, consulta Endpoint VPC dell'interfaccia (AWS PrivateLink) nella Guida per l'utente di Amazon VPC.

Considerazioni sugli endpoint AWS IoT Core Device Advisor VPC

Prima di impostare endpoint VPC dell'interfaccia, rivedi le proprietà e le limitazioni dell'endpoint dell'interfaccia nella Guida per l'utente di Amazon VPC. Prima di continuare, valuta quanto segue:

- AWS IoT Core Device Advisor attualmente supporta l'effettuazione di chiamate all'endpoint di test Device Advisor (piano dati) dal tuo VPC. Un broker di messaggi utilizza le comunicazioni presenti nel piano dati per inviare e ricevere dati, mediante l'uso di pacchetti TLS e MQTT. Endpoint VPC per AWS IoT Core Device Advisor connettere il dispositivo agli endpoint di AWS IoT test Device Advisor. Le azioni API del piano di controllo (control-plane) non vengono utilizzate da questo endpoint VPC. Per creare o eseguire una suite di test o un altro piano di controllo APIs, utilizza la console, un AWS SDK o un'interfaccia a riga di AWS comando sulla rete Internet pubblica.
- I seguenti endpoint VPC Regioni AWS supportano: AWS IoT Core Device Advisor
  - Stati Uniti orientali (Virginia settentrionale)
  - Stati Uniti occidentali (Oregon)
  - Asia Pacifico (Tokyo)
  - Europa (Irlanda)
- Device Advisor supporta MQTT con certificati client X.509 e i certificati server RSA.
- Le policy degli endpoint VPC non sono supportate in questo momento.
- Controlla i prerequisiti degli endpoint VPC per istruzioni su come creare risorse che collegano endpoint VPC. È necessario creare un VPC e sottoreti private per utilizzare gli endpoint VPC. AWS IoT Core Device Advisor
- Le tue risorse sono soggette a quote. AWS PrivateLink Per ulteriori informazioni, consulta la pagina relativa alle quote di AWS PrivateLink.
- Gli endpoint VPC supportano solo il traffico. IPv4

Crea un endpoint VPC di interfaccia per AWS IoT Core Device Advisor

Per iniziare a usare gli endpoint VPC, crea un endpoint VPC dell'interfaccia. Quindi, seleziona AWS IoT Core Device Advisor come. Servizio AWS Se utilizzi il AWS CLI, chiama describe-vpc-endpoint-servicesper confermare che AWS IoT Core Device Advisor è presente in una zona di disponibilità del tuo Regione AWS. Verifica che il gruppo di sicurezza collegato all'endpoint consenta la comunicazione tramite protocollo TCP per il traffico MQTT e TLS. Ad esempio, nella regione Stati Uniti orientali (Virginia settentrionale), utilizza il seguente comando:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-
east-1.deviceadvisor.iot
```

Puoi creare un endpoint VPC per AWS IoT Core utilizzare il seguente nome di servizio:

- com.amazonaws.region.deviceadvisor.iot

Per impostazione predefinita, il DNS privato è attivato per l'endpoint. Ciò garantisce che l'uso dell'endpoint di test predefinito rimanga all'interno delle sottoreti private. Per creare un endpoint a livello di account o dispositivo, utilizza la console AWS CLI o un SDK. AWS Ad esempio, se esegui get-endpoint all'interno di una sottorete pubblica o sulla rete Internet pubblica, puoi ottenere l'endpoint e utilizzarlo per connetterti a Device Advisor. Per ulteriori informazioni, consultare Accesso a un servizio tramite un endpoint di interfaccia in Guida per l'utente di Amazon VPC.

Per connettere i client MQTT alle interfacce degli endpoint VPC, AWS PrivateLink il servizio crea record DNS in una zona ospitata privata collegata al VPC. Questi record DNS indirizzano le richieste del dispositivo AWS IoT all'endpoint VPC.

Controllo dell'accesso agli AWS IoT Core Device Advisor endpoint tramite VPC

Puoi limitare l'accesso ai dispositivi AWS IoT Core Device Advisor e consentire l'accesso solo tramite gli endpoint VPC utilizzando le chiavi contestuali delle condizioni VPC. AWS IoT Core supporta le seguenti chiavi di contesto relative al VPC:

- SourceVpc
- SourceVpce
- VPCSourceIp

> **Note**
>
> AWS IoT Core Device Advisor al momento non supporta le policy degli endpoint VPC.

La seguente politica concede l'autorizzazione alla connessione AWS IoT Core Device Advisor utilizzando un ID client che corrisponde al nome dell'oggetto. Inoltre, pubblica in qualsiasi argomento

che abbia come prefisso il nome dell'oggetto. La policy è subordinata al dispositivo che si connette a un endpoint VPC con un particolare ID endpoint VPC. Questa policy nega i tentativi di connessione all'endpoint di test AWS IoT Core Device Advisor pubblico.

```
{
"Version":"2012-10-17",
    "Statement": [
        {
"Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
"StringEquals": {
"aws:SourceVpce": "vpce-1a2b3c4d"
            }
        }

        },
        {
"Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}/*"
            ]
        }
    ]
}
```

Case test di Device Advisor

Device Advisor fornisce test predefiniti in sei categorie.

- TLS
- MQTT
- Shadow
- Esecuzione del processo
- Autorizzazioni e policy
- Test di lunga durata

Casi di test di Device Advisor per qualificarsi per il AWS Device Qualification Program.

Il dispositivo deve superare i seguenti test per essere qualificato in base al Programma AWS per la qualifica dei dispositivi.

> **Note**
>
> Questo è un elenco rivisto dei test di qualifica.

- Collegamento TLS ("TLS Connect")
- Certificato server nome oggetto TLS non corretto ("Nome comune soggetto non corretto (CN)/ Nome alternativo soggetto (SAN)")
- Certificato server TLS non sicuro ("Non firmato da CA riconosciuta")
- Supporto per dispositivi TLS per suite di AWS IoT crittografia («Supporto dispositivi TLS AWS IoT per suite di crittografia consigliate»)
- Frammenti di dimensioni massime ricezione TLS("Frammenti di dimensioni massime ricezione TLS")
- Certificato server scaduto TLS("Certificato server scaduto")
- Certificato server di grandi dimensioni TLS("Certificato server di grandi dimensioni TLS")
- MQTT Connect («Dispositivo invia CONNECT a AWS IoT Core (Happy case)»)
- Iscriviti MQTT ("Puoi iscriverti (Happy Case)")
- Pubblica MQTT ("QoS0 (Happy Case)")
- Tentativi jitter di connessione MQTT("Tentativi di connessione dispositivo con backoff jitter - Nessuna risposta CONNACK")

TLS

Utilizza questi test per determinare se il protocollo TLS (Transport Layer Security Protocol) tra i tuoi dispositivi AWS IoT è sicuro.

> **Note**
>
> Device Advisor ora supporta TLS 1.3.

Happy Path

Connessione TLS

Verifica se il dispositivo sottoposto a test è in grado di completare l'handshake TLS verso. AWS IoT Questo test non convalida l'implementazione MQTT del dispositivo client.

Example Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Per ottenere risultati ottimali, si consiglia un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_tls_connect_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Connect",
         "version":"0.0.0"
      }
   }
]
```

Example Output del test case:

- Pass: il dispositivo in test ha completato l'handshake TLS con. AWS IoT
- Passato con avvisi: il dispositivo in test ha completato l'handshake TLS con AWS IoT, ma sono stati ricevuti messaggi di avviso TLS dal dispositivo o. AWS IoT
- Errore: il dispositivo sottoposto a test non è riuscito a completare l'handshake TLS con a causa di un errore di handshake. AWS IoT

TLS riceve frammenti di dimensioni massime

Questo caso di test verifica se il dispositivo è in grado di ricevere ed elaborare frammenti di dimensioni massime TLS. Il dispositivo di test deve effettuare la sottoscrizione a un argomento preconfigurato con QoS 1 per ricevere un payload di grandi dimensioni. Il payload può essere personalizzato con la configurazione `${payload}`.

Example Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Per ottenere risultati ottimali, si consiglia un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"TLS Receive Maximum Size Fragments",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
         "PAYLOAD_FORMAT":"{"message":"${payload}"}", // A string with a placeholder
 ${payload}, or leave it empty to receive a plain string.
         "TRIGGER_TOPIC": "test_1" // A topic to which a device will subscribe, and
 to which a test case will publish a large payload.
      },
      "test":{
         "id":"TLS_Receive_Maximum_Size_Fragments",
         "version":"0.0.0"
      }
   }
]
```

Suite di cifratura

Supporto per dispositivi TLS per le suite di AWS IoT crittografia consigliate

Verifica che i pacchetti di crittografia nel messaggio TLS Client Hello dal dispositivo sottoposto a test contengano i pacchetti di crittografia AWS IoT consigliati. Fornisce ulteriori informazioni sulle suite di crittografia supportate dal dispositivo.

Example Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_tls_support_aws_iot_cipher_suites_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"TLS_Support_AWS_IoT_Cipher_Suites",
         "version":"0.0.0"
      }
   }
]
```

Example Output dei test case:

- Pass: il dispositivo sottoposto a test contiene almeno una delle suite di crittografia AWS IoT consigliate e non contiene suite di crittografia non supportate.
- Superato con avvisi: i pacchetti di crittografia del dispositivo contengono almeno un pacchetto di crittografia AWS IoT ma:
  1. Non contengono alcun pacchetto di crittografia consigliato
  2. Contiene suite di crittografia che non sono supportate da. AWS IoT

  Si consiglia di verificare che gli eventuali pacchetti di crittografia non supportati siano sicuri.

- Errore: il dispositivo sottoposto a test sulle suite di crittografia non contiene nessuna delle suite di crittografia supportate AWS IoT .

Certificato del server di grandi dimensioni

Certificato del server di grandi dimensioni TLS

Verifica se il dispositivo è in grado di completare l'handshake TLS con AWS IoT quando riceve ed elabora un certificato del server di grandi dimensioni. La dimensione del certificato del server (in byte) utilizzato da questo test è maggiore di 20 rispetto a quella attualmente utilizzata nel test case TLS Connect e IoT Core. Durante questo test case, AWS IoT verifica lo spazio di buffer del dispositivo per TLS. Se lo spazio nel buffer è sufficientemente grande, l'handshake TLS viene completato senza errori. Questo test non convalida l'implementazione MQTT del dispositivo. Il caso di test termina dopo il completamento del processo di handshake TLS.

Example Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Per ottenere risultati ottimali, si consiglia un valore di timeout di 2 minuti. Se questo caso di test non va a buon fine, ma il caso di test Connessione TLS viene superato, si consiglia di incrementare il limite di spazio del buffer del dispositivo per TLS. L'aumento del limite di spazio del buffer garantisce che il dispositivo sia in grado di elaborare un certificato del server di dimensioni maggiori nel caso in cui le dimensioni aumentino.

```
"tests":[
   {
      "name":"my_tls_large_size_server_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"TLS_Large_Size_Server_Cert",
         "version":"0.0.0"
      }
   }
```

```
]
```

Example Output del test case:

- Pass (Superato) - Il dispositivo testato ha completato l'handshake TLS con AWS IoT.
- Passa con avvisi: il dispositivo in test ha completato l'handshake TLS con AWS IoT, ma sono presenti messaggi di avviso TLS dal dispositivo o AWS IoT.
- Errore: il dispositivo sottoposto a test non è riuscito a completare l'handshake TLS a AWS IoT causa di un errore durante il processo di handshake.

Certificato del server non sicuro TLS

Non firmato da CA riconosciuta

Verifica che il dispositivo sottoposto a test chiuda la connessione se viene presentato con un certificato del server che non dispone di una firma valida dalla CA ATS. Un dispositivo deve connettersi solo a un endpoint che presenta un certificato valido.

Example Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_tls_unsecure_server_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Unsecure_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Output dei test case:

- Pass (Superato) — Il dispositivo sottoposto a test ha chiuso la connessione.
- Fallito: il dispositivo in test ha completato l'handshake TLS con. AWS IoT

Certificato del server nome soggetto TLS non corretto/Nome comune soggetto non corretto (CN)/ Nome alternativo soggetto (SAN)

Convalida che il dispositivo in fase di test chiude la connessione se viene presentato con un certificato server per un nome di dominio diverso da quello richiesto.

Example Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_tls_incorrect_subject_name_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",   // in seconds
      },
      "test":{
         "id":"TLS_Incorrect_Subject_Name_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Output dei test case:

- Pass (Superato) — Il dispositivo sottoposto a test ha chiuso la connessione.
- Fallito: il dispositivo in test ha completato l'handshake TLS con. AWS IoT

Certificato server scaduto TLS

Certificato del server scaduto

Convalida che il dispositivo in fase di test chiude la connessione se viene presentato con un certificato server scaduto.

Example Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_tls_expired_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Expired_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Output dei test case:

- Superato: il dispositivo sottoposto al test si rifiuta di completare l'handshake TLS con. AWS IoT Il dispositivo invia un messaggio di avviso TLS prima di chiudere la connessione.
- Superato con avvisi: il dispositivo sottoposto a test rifiuta di completare l'handshake TLS con AWS IoT. Tuttavia, non invia un messaggio di avviso TLS prima di chiudere la connessione.
- Fallito: il dispositivo in test completa l'handshake TLS con. AWS IoT

MQTT

CONNETTI, SCONNETTI e RICONNETTI

«Il dispositivo invia CONNECT a AWS IoT Core (Happy case)»

Convalida che il dispositivo sottoposto a test invia una richiesta CONNECT.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_mqtt_connect_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",   // in seconds
      },
      "test":{
         "id":"MQTT_Connect",
         "version":"0.0.0"
      }
   }
]
```

"Il dispositivo può restituire PUBACK a un argomento arbitrario per QoS1"

Questo test case verificherà se il dispositivo (client) può restituire un messaggio PUBACK se ha ricevuto un messaggio di pubblicazione dal broker dopo essersi iscritto a un argomento con QoS1.

Il contenuto e la dimensione del payload sono configurabili per questo test case. Se la dimensione del payload è configurata, Device Advisor sovrascriverà il valore del contenuto del payload e invierà un payload predefinito al dispositivo con le dimensioni desiderate. La dimensione del payload è un valore compreso tra 0 e 128 KB e non può superare i 128 KB. AWS IoT Core rifiuta

le richieste di pubblicazione e di connessione superiori a 128 KB, come illustrato nella pagine Limiti e quote del protocollo e del broker di messaggi AWS IoT Core.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti. `PAYLOAD_SIZE` può essere configurato su un valore compreso tra 0 e 128 kilobyte. La definizione della dimensione di un payload sostituisce il contenuto del payload poiché Device Advisor invierà al dispositivo un payload predefinito con le dimensioni specificate.

```
"tests":[
{
        "name":"my_mqtt_client_puback_qos1",
        "configuration": {
            // optional:"TRIGGER_TOPIC": "myTopic",
            "EXECUTION_TIMEOUT":"300", // in seconds
            "PAYLOAD_FOR_PUBLISH_VALIDATION":"custom payload",
            "PAYLOAD_SIZE":"100" // in kilobytes
        },
        "test": {
            "id": "MQTT_Client_Puback_QoS1",
            "version": "0.0.0"
        }
    }
]
```

"Riconnessione del dispositivo con backoff del jitter - Nessuna risposta CONNACK"

Convalida che il dispositivo in fase di test utilizza il backoff jitter corretto quando si ricollega con il broker per almeno cinque volte. Il broker registra il timestamp del dispositivo sotto la richiesta CONNECT del test, esegue la convalida dei pacchetti, si interrompe senza inviare un CONNACK al dispositivo in fase di test e attende che il dispositivo sottoposto a test invii nuovamente la richiesta. Il sesto tentativo di connessione è consentito per passare attraverso e CONNACK è autorizzato a fluire di nuovo al dispositivo in prova.

Il processo precedente viene eseguito di nuovo. In totale, questo test case richiede che il dispositivo si connetta almeno 12 volte in totale. I timestamp raccolti vengono utilizzati per

convalidare che il jitter backoff venga utilizzato dal dispositivo in prova. Se il dispositivo sottoposto a test ha un ritardo di backoff strettamente esponenziale, questo test case sarà superato con avvertimenti.

Raccomandiamo l'implementazione del meccanismo Exponential Backoff And Jitter (Backoff esponenziale e jitter) sul dispositivo sottoposto a test per superare questo test case.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 4 minuti.

```
"tests":[
   {
      "name":"my_mqtt_jitter_backoff_retries_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",    // in seconds
      },
      "test":{
         "id":"MQTT_Connect_Jitter_Backoff_Retries",
         "version":"0.0.0"
      }
   }
]
```

"Riconnessione del dispositivo con backoff esponenziale - Nessuna risposta CONNACK"

Verifica che il dispositivo sottoposto a test utilizzi il backoff esponenziale appropriato durante la riconnessione con il broker per almeno cinque volte. Il broker registra il timestamp del dispositivo sotto la richiesta CONNECT del test, esegue la convalida dei pacchetti, si interrompe senza inviare un CONNACK al dispositivo client e attende che il dispositivo in fase di test invii nuovamente la richiesta. I timestamp raccolti vengono utilizzati per convalidare che il backoff esponenziale venga utilizzato dal dispositivo in prova.

Raccomandiamo l'implementazione del meccanismo Exponential Backoff And Jitter (Backoff esponenziale e jitter) sul dispositivo sottoposto a test per superare questo test case.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 4 minuti.

```
"tests":[
   {
      "name":"my_mqtt_exponential_backoff_retries_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"600",  // in seconds
      },
      "test":{
         "id":"MQTT_Connect_Exponential_Backoff_Retries",
         "version":"0.0.0"
      }
   }
]
```

"Riconnessione del dispositivo con jitter backoff - Dopo la disconnessione del server"

Convalida se un dispositivo sottoposto a test utilizza il jitter e il backoff necessari durante la riconnessione dopo che è stato disconnesso dal server. Device Advisor disconnette il dispositivo dal server per almeno cinque volte e osserva il comportamento del dispositivo per la riconnessione MQTT. Device Advisor registra il timestamp della richiesta CONNECT per il dispositivo in fase di test, esegue la convalida dei pacchetti, si interrompe senza inviare un CONNACK al dispositivo client e attende che il dispositivo in fase di test invii nuovamente la richiesta. I timestamp raccolti vengono utilizzati per convalidare che il dispositivo sotttoposto a test utilizzi jitter e backoff durante la riconnessione. Se il dispositivo sottoposto a test ha un backoff strettamente esponenziale o non implementa un meccanismo jitter backoff appropriato, questo test case passerà con avvertimenti. Se il dispositivo sottoposto a test ha implementato un meccanismo di backoff lineare o un meccanismo di backoff costante, il test non andrà a buon fine.

Per superare questo test case, si consiglia di implementare il meccanismo Exponential Backoff And Jitter (Backoff esponenziale e jitter) sul dispositivo sottoposto a test.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 4 minuti.
> Il numero di tentativi di riconnessione da convalidare per il backoff può essere modificato specificando il `RECONNECTION_ATTEMPTS`. Il numero deve essere compreso tra 5 e 10. Il valore predefinito è 5.

```
"tests":[
   {
      "name":"my_mqtt_reconnect_backoff_retries_on_server_disconnect",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "RECONNECTION_ATTEMPTS": 5
      },
      "test":{
         "id":"MQTT_Reconnect_Backoff_Retries_On_Server_Disconnect",
         "version":"0.0.0"
      }
   }
]
```

"Device re-connect with jitter backoff - On unstable connection" (Riconnessione del dispositivo con jitter backoff - Connessione instabile)

Convalida se un dispositivo sottoposto a test utilizza il jitter e il backoff necessari durante la riconnessione su una connessione instabile. Device Advisor disconnette il dispositivo dal server dopo cinque connessioni valide e osserva il comportamento del dispositivo per la riconnessione MQTT. Device Advisor registra il timestamp della richiesta CONNECT per il dispositivo in fase di test, esegue la convalida dei pacchetti, invia un CONNACK, si disconnette, registra il timestamp della disconnessione e attende che il dispositivo in fase di test invii nuovamente la richiesta. I timestamp raccolti vengono utilizzati per convalidare che il dispositivo sottoposto a test utilizzi jitter e backoff durante connessioni corrette ma instabili. Se il dispositivo sottoposto a test ha un backoff strettamente esponenziale o non implementa un meccanismo jitter backoff appropriato, questo test case passerà con avvertimenti. Se il dispositivo sottoposto a test ha implementato un meccanismo di backoff lineare o un meccanismo di backoff costante, il test non andrà a buon fine.

Per superare questo test case, si consiglia di implementare il meccanismo Exponential Backoff And Jitter (Backoff esponenziale e jitter) sul dispositivo sottoposto a test.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 4 minuti.
> Il numero di tentativi di riconnessione da convalidare per il backoff può essere modificato specificando il `RECONNECTION_ATTEMPTS`. Il numero deve essere compreso tra 5 e 10. Il valore predefinito è 5.

```
"tests":[
   {
      "name":"my_mqtt_reconnect_backoff_retries_on_unstable_connection",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "RECONNECTION_ATTEMPTS": 5
      },
      "test":{
         "id":"MQTT_Reconnect_Backoff_Retries_On_Unstable_Connection",
         "version":"0.0.0"
      }
   }
]
```

Pubblica

"QoS0 (Happy Case)"

Verifica che il dispositivo in test pubblichi un messaggio con QoS0 o QoS1. È inoltre possibile convalidare l'argomento del messaggio e il payload specificando il valore dell'argomento e il payload nelle impostazioni di test.

> **Note**
>
> EXECUTION_TIMEOUT dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_mqtt_publish_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_FOR_PUBLISH_VALIDATION": "my_TOPIC_FOR_PUBLISH_VALIDATION",
         "PAYLOAD_FOR_PUBLISH_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish",
         "version":"0.0.0"
      }
   }
]
```

"QoS1 riprova pubblicazione - Nessun PUBACK"

Convalida che il dispositivo in fase di test ripubblica un messaggio inviato con QOS1, se il broker non invia PUBACK. È inoltre possibile convalidare l'argomento del messaggio specificando questo argomento nelle impostazioni di test. Il dispositivo client non deve disconnettersi prima di ripubblicare il messaggio. Questo test conferma inoltre che il messaggio ripubblicato abbia lo stesso identificatore di pacchetto dell'originale. Durante l'esecuzione del test, se il dispositivo perde la connessione e si riconnette, il test case si ripristina senza errori e il dispositivo deve eseguire nuovamente i passaggi del test case.

Definizione del test case API:

> **Note**
>
> EXECUTION_TIMEOUT dispone di un valore predefinito di 5 minuti. È consigliato per almeno 4 minuti.

```
"tests":[
   {
      "name":"my_mqtt_publish_retry_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_FOR_PUBLISH_VALIDATION": "my_TOPIC_FOR_PUBLISH_VALIDATION",
         "PAYLOAD_FOR_PUBLISH_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish_Retry_No_Puback",
         "version":"0.0.0"
      }
   }
]
```

"Pubblica messaggi mantenuti"

Convalida che il dispositivo in fase di test pubblica un messaggio `retainFlag` impostato su true. È inoltre possibile convalidare l'argomento e il payload del messaggio impostando il valore dell'argomento e il payload nelle impostazioni di test. Se il `retainFlag` inviato all'interno del pacchetto PUBLISH non è impostato su true, il test case avrà esito negativo.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti. Per eseguire questo test case, aggiungere l'azione `iot:RetainPublish` nel ruolo del dispositivo.

```
"tests":[
   {
      "name":"my_mqtt_publish_retained_messages_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds

 "TOPIC_FOR_PUBLISH_RETAINED_VALIDATION": "my_TOPIC_FOR_PUBLISH_RETAINED_VALIDATION",
```

```
 "PAYLOAD_FOR_PUBLISH_RETAINED_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_RETAINED_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish_Retained_Messages",
         "version":"0.0.0"
      }
   }
]
```

"Pubblica con proprietà utente"

Convalida che il dispositivo in fase di test pubblica un messaggio con la proprietà utente corretta. È possibile convalidare la proprietà utente impostando la coppia nome-valore nelle impostazioni del test. Se la proprietà utente non viene fornita o non corrisponde, il test case ha esito negativo.

Definizione del test case API:

> **Note**
>
> Questo è un unico test case. MQTT5 `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_mqtt_user_property_test",
      "test":{
        "USER_PROPERTIES": [
            {"name": "name1", "value":"value1"},
            {"name": "name2", "value":"value2"}
        ],
        "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"MQTT_Publish_User_Property",
         "version":"0.0.0"
      }
   }
]
```

Subscribe

"Puoi iscriverti (Happy Case)"

Convalida che il dispositivo sottoposto a test si sottoscriva gli argomenti MQTT. È inoltre possibile convalidare l'argomento a cui il dispositivo sottoposto a test sottoscriva specificando questo argomento nelle impostazioni di test.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_mqtt_subscribe_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION":
["my_TOPIC_FOR_PUBLISH_VALIDATION_a","my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
      },
      "test":{
         "id":"MQTT_Subscribe",
         "version":"0.0.0"
      }
   }
]
```

"Riprova sottoscrizione - Nessun SUBACK"

Convalida che il dispositivo in fase di test tenta una sottoscrizione non riuscita agli argomenti MQTT. Il server attende e non invia un SUBACK. Se il dispositivo client non riprova la sottoscrizione, il test ha esito negativo. Il dispositivo client deve riprovare la sottoscrizione non riuscita con lo stesso pacchetto Id. È inoltre possibile convalidare l'argomento a cui il dispositivo sottoposto a test sottoscriva specificando questo argomento nelle impostazioni di test. Durante l'esecuzione del test, se il dispositivo perde la connessione e si riconnette, il test case si ripristina senza errori e il dispositivo deve eseguire nuovamente i passaggi del test case.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 4 minuti.

```
"tests":[
   {
      "name":"my_mqtt_subscribe_retry_test",
      "configuration":{
         "EXECUTION_TIMEOUT":"300",  // in seconds
         // optional:
         "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION":
["myTOPIC_FOR_PUBLISH_VALIDATION_a","my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
      },
      "test":{
         "id":"MQTT_Subscribe_Retry_No_Suback",
         "version":"0.0.0"
      }
   }
]
```

Keep-Alive

«Matt No Acc» PingResp

Questo test case convalida se il dispositivo sottoposto a test si disconnette quando non riceve una risposta ping. Nell'ambito di questo test case, Device Advisor blocca le risposte inviate AWS IoT Core per le richieste di pubblicazione, sottoscrizione e ping. Convalida anche se il dispositivo sottoposto a test disconnette la connessione MQTT.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un timeout superiore a 1,5 volte il valore `keepAliveTime`.

Il massimo non `keepAliveTime` deve essere superiore a 230 secondi per questo test.

```
"tests":[
    {
       "name":"Mqtt No Ack PingResp",
       "configuration":
          //optional:
          "EXECUTION_TIMEOUT":"306",   // in seconds
       },
       "test":{
          "id":"MQTT_No_Ack_PingResp",
          "version":"0.0.0"
       }
    }
]
```

Sessione persistente

"Sessione persistente (happy case)"

Questo test case convalida il comportamento del dispositivo quando viene disconnesso da una sessione persistente. Il test case verifica se il dispositivo è in grado di riconnettersi, riprendere le sottoscrizioni agli argomenti di attivazione senza sottoscriversi nuovamente in modo esplicito, ricevere i messaggi archiviati negli argomenti e funzionare come previsto durante una sessione persistente. Quando questo test case viene superato, indica che il dispositivo client è in grado di mantenere una sessione persistente con il AWS IoT Core broker nel modo previsto. Per ulteriori informazioni sulle sessioni AWS IoT persistenti, vedere Utilizzo delle sessioni persistenti MQTT.

In questo test case, è previsto che il dispositivo client esegua una richiesta CONNECT a AWS IoT Core con un flag di sessione pulita impostato su false, quindi effettui la sottoscrizione a un argomento di attivazione. Dopo un abbonamento riuscito, il dispositivo verrà disconnesso da AWS IoT Core Device Advisor. Mentre il dispositivo si trova nello stato disconnesso, in tale argomento verrà memorizzato un payload del messaggio QoS 1. Device Advisor consentirà quindi al dispositivo client di connettersi nuovamente con l'endpoint di test. A questo punto, poiché esiste una sessione persistente, il dispositivo client riprende le sottoscrizioni degli argomenti senza inviare pacchetti SUBSCRIBE aggiuntivi e ricevere il messaggio QoS 1 dal broker. Dopo la riconnessione, se il dispositivo client si iscrive nuovamente al relativo argomento di

attivazione inviando un pacchetto SUBSCRIBE aggiuntivo, and/or se il client non riesce a ricevere il messaggio memorizzato dall'argomento di attivazione, il test case avrà esito negativo.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di almeno 4 minuti. Alla prima connessione, il dispositivo client deve sottoscrivere esplicitamente un `TRIGGER_TOPIC` non sottoscritto in precedenza. Per superare il test case, il dispositivo client deve sottoscrivere correttamente `TRIGGER_TOPIC` con un QoS 1. Dopo la riconnessione, il dispositivo client comprende che esiste una sessione persistente attiva e pertanto accetta il messaggio memorizzato inviato dall'argomento di attivazione e restituisce PUBACK per il messaggio specifico.

```
"tests":[
   {
      "name":"my_mqtt_persistent_session_happy_case",
      "configuration":{
         //required:
         "TRIGGER_TOPIC": "myTrigger/topic",
         // optional:
         // if Payload not provided, a string will be stored in the trigger topic to
 be sent back to the client device
         "PAYLOAD": "The message which should be received from AWS IoT Broker after
 re-connecting to a persistent session from the specified trigger topic.",

         "EXECUTION_TIMEOUT":"300" // in seconds
      },
      "test":{
         "id":"MQTT_Persistent_Session_Happy_Case",
         "version":"0.0.0"
      }
   }
]
```

"Sessione persistente - Scadenza della sessione"

Questo test case aiuta a convalidare il comportamento del dispositivo quando un dispositivo disconnesso si riconnette a una sessione persistente scaduta. Dopo la scadenza della sessione,

è previsto che il dispositivo sottoscriva nuovamente gli argomenti precedentemente sottoscritti inviando esplicitamente un nuovo pacchetto SUBSCRIBE.

Durante la prima connessione, ci aspettiamo che il dispositivo di test si CONNETTA al broker AWS IoT, poiché il suo `CleanSession` flag è impostato su false per avviare una sessione persistente. Il dispositivo deve quindi sottoscrivere un argomento di attivazione. Quindi il dispositivo viene disconnesso da AWS IoT Core Device Advisor, dopo una sottoscrizione riuscita e l'avvio di una sessione persistente. Dopo la disconnessione, AWS IoT Core Device Advisor consente al dispositivo di test di riconnettersi all'endpoint di test. A questo punto, quando il dispositivo di test invia un altro pacchetto CONNECT, AWS IoT Core Device Advisor restituisce un pacchetto CONNACK che indica che la sessione persistente è scaduta. Il dispositivo di test deve interpretare correttamente questo pacchetto e si prevede che sottoscriva nuovamente lo stesso argomento di attivazione quando la sessione persistente viene terminata. Se il dispositivo di test non sottoscrive di nuovo l'argomento, il test case non riesce. Per superare il test, il dispositivo deve comprendere che la sessione persistente è terminata e inviare un nuovo pacchetto SUBSCRIBE per lo stesso argomento di attivazione nella seconda connessione.

Se il test case di un dispositivo di test riesce, significa che il dispositivo è in grado di gestire la riconnessione alla scadenza della sessione persistente nel modo previsto.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di almeno 4 minuti. Il dispositivo client deve sottoscrivere esplicitamente un `TRIGGER_TOPIC` non sottoscritto in precedenza. Per superare il test case, il dispositivo di test deve inviare un pacchetto CONNECT con il flag `CleanSession` impostato su false e sottoscrivere correttamente un argomento di attivazione con un QoS 1. Dopo una connessione riuscita, AWS IoT Core Device Advisor disconnette il dispositivo. Dopo la disconnessione, AWS IoT Core Device Advisor consente al dispositivo di riconnettersi e si prevede che il dispositivo si iscriva nuovamente alla stessa, `TRIGGER_TOPIC` poiché AWS IoT Core Device Advisor avrebbe interrotto la sessione persistente.

```
"tests":[
   {
      "name":"my_expired_persistent_session_test",
      "configuration":{
```

```
         //required:
         "TRIGGER_TOPIC": "myTrigger/topic",
         // optional:
         "EXECUTION_TIMEOUT":"300" // in seconds
      },
      "test":{
         "id":"MQTT_Expired_Persistent_Session",
         "version":"0.0.0"
      }
   }
]
```

Shadow

Utilizza questi test per verificare che i dispositivi in prova utilizzino correttamente il servizio AWS IoT Device Shadow. Per ulteriori informazioni, consulta AWS IoT Servizio Device Shadow. Se questi test case sono configurati nella suite di test, è necessario fornire una cosa all'avvio dell'esecuzione della suite.

Al momento l'over MQTT non WebSocket è supportato.

Pubblica

"Il dispositivo pubblica lo stato dopo la connessione (Happy case)"

Verifica se un dispositivo può pubblicare il suo stato dopo la connessione a AWS IoT Core

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_shadow_publish_reported_state",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300", // in seconds
```

```
            "SHADOW_NAME": "SHADOW_NAME",
            "REPORTED_STATE": {
               "STATE_ATTRIBUTE": "STATE_VALUE"
            }
         },
         "test":{
            "id":"Shadow_Publish_Reported_State",
            "version":"0.0.0"
         }
      }
]
```

Il REPORTED_STATE può essere fornito per una convalida aggiuntiva sullo stato di shadow esatto del dispositivo dopo la connessione. Per impostazione predefinita, questo test case convalida lo stato di pubblicazione del dispositivo.

Se *SHADOW_NAME* non viene fornito, il test case cerca i messaggi pubblicati nei prefissi dell'argomento del tipo di shadow senza nome (classica) per impostazione predefinita. Fornisci un nome shadow se il dispositivo utilizza il tipo di shadow denominato. Per ulteriori informazioni., consulta Utilizzo delle copie shadow nei dispositivi.

Aggiorna

"Il dispositivo aggiorna lo stato segnalato sullo stato desiderato (Happy case)"

Convalida se il dispositivo legge tutti i messaggi di aggiornamento ricevuti e sincronizza lo stato del dispositivo in modo che corrisponda alle proprietà dello stato desiderate. Il dispositivo dovrebbe pubblicare l'ultimo stato segnalato dopo la sincronizzazione. Se il dispositivo ha già una shadow esistente prima di eseguire il test, assicurati che lo stato desiderato configurato per il test case e lo stato segnalato esistente non corrispondano già. È possibile identificare i messaggi di aggiornamento Shadow inviati da Device Advisor esaminando il ClientTokencampo nel documento Shadow così come saràDeviceAdvisorShadowTestCaseSetup.

Definizione del test case API:

> **Note**
>
> EXECUTION_TIMEOUT dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 2 minuti.

```
"tests":[
   {
      "name":"my_shadow_update_reported_state",
      "configuration": {
         "DESIRED_STATE": {
            "STATE_ATTRIBUTE": "STATE_VALUE"
         },
         // optional:
         "EXECUTION_TIMEOUT":"300", // in seconds
         "SHADOW_NAME": "SHADOW_NAME"
      },
      "test":{
         "id":"Shadow_Update_Reported_State",
         "version":"0.0.0"
      }
   }
]
```

`DESIRED_STATE` deve avere almeno un attributo e un valore associato.

Se `SHADOW_NAME` non viene fornito, il test case cerca i messaggi pubblicati nei prefissi dell'argomento del tipo di shadow senza nome (classico) per impostazione predefinita. Fornisci un nome shadow se il dispositivo utilizza il tipo di shadow denominato. Per ulteriori informazioni, consulta Utilizzo delle copie shadow nei dispositivi.

Esecuzione del processo

"Il dispositivo può completare l'esecuzione di un processo"

Questo test case consente di verificare se il dispositivo è in grado di ricevere aggiornamenti tramite AWS IoT Jobs e di pubblicare lo stato degli aggiornamenti eseguiti con successo. Per ulteriori informazioni su AWS IoT Jobs, consulta Jobs.

Per eseguire correttamente questo test case, ci sono due AWS argomenti riservati a cui devi assegnare il ruolo del dispositivo. Per effettuare la sottoscrizione ai messaggi relativi all'attività del processo, utilizza gli argomenti notify e notify-next. Il ruolo del dispositivo deve consentire l'azione PUBLISH per i seguenti argomenti:

- $aws/things/thingName/jobs/jobId/get
- $aws/things/thingName/jobs/jobId/update

Si consiglia di concedere le azioni SUBSCRIBE e RECEIVE per i seguenti argomenti:

- $aws/things/ ThingName/jobs/get/accepted
- $aws/things/thingName/jobs/jobId/get/rejected
- $aws/things/thingName/jobs/jobId/update/accepted
- $aws/things/thingName/jobs/jobId/update/rejected

Si consiglia di concedere l'azione SUBSCRIBE per il seguente argomento:

- $aws/things/thingName/jobs/notify-next

Per ulteriori informazioni su questi argomenti riservati, consulta gli argomenti riservati per Jobs AWS IoT.

Al momento l'over WebSocket MQTT non è supportato.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 5 minuti. Consigliamo un valore di timeout di 3 minuti. A seconda del documento AWS IoT Job o della fonte forniti, modifica il valore di timeout (ad esempio, se l'esecuzione di un lavoro richiederà molto tempo, definisci un valore di timeout più lungo per il test case). Per eseguire il test, è necessario un documento AWS IoT Job valido o un ID lavoro già esistente. Un documento AWS IoT Job può essere fornito come documento JSON o collegamento S3. Se viene fornito un documento di lavoro, fornire un ID di lavoro è facoltativo. Se viene fornito un Job ID, Device Advisor utilizzerà tale ID durante la creazione del AWS IoT Job per conto dell'utente. Se il documento di lavoro non viene fornito, puoi fornire un ID esistente che si trova nella stessa regione in cui stai eseguendo il test case. In questo caso, Device Advisor utilizzerà quel AWS IoT Job durante l'esecuzione del test case.

```
"tests": [
   {
      "name":"my_job_execution",
      "configuration": {
         // optional:
         // Test case will create a job task by using either JOB_DOCUMENT or
 JOB_DOCUMENT_SOURCE.
```

```
        // If you manage the job task on your own, leave it empty and provide the
 JOB_JOBID (self-managed job task).
        // JOB_DOCUMENT is a JSON formatted string
        "JOB_DOCUMENT": "{
           \"operation\":\"reboot\",
           \"files\" : {
              \"fileName\" : \"install.py\",
              \"url\" : \"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/
bucket-name/key}\"
           }
        }",
        // JOB_DOCUMENT_SOURCE is an S3 link to the job document. It will be used
 only if JOB_DOCUMENT is not provided.
        "JOB_DOCUMENT_SOURCE": "https://s3.amazonaws.com/bucket-name/key",
        // JOB_JOBID is mandatory, only if neither document nor document source is
 provided. (Test case needs to know the self-managed job task id).
        "JOB_JOBID": "String",
        // JOB_PRESIGN_ROLE_ARN is used for the presign Url, which will replace the
 placeholder in the JOB_DOCUMENT field
        "JOB_PRESIGN_ROLE_ARN": "String",
        // Presigned Url expiration time. It must be between 60 and 3600 seconds,
 with the default value being 3600.
        "JOB_PRESIGN_EXPIRES_IN_SEC": "Long"
        "EXECUTION_TIMEOUT": "300", // in seconds
     },
     "test": {
        "id": "Job_Execution",
        "version": "0.0.0"
     }
   }
]
```

Per ulteriori informazioni sulla creazione e sull'utilizzo di documenti del processo, consulta documento del processo.

Autorizzazioni e policy

Utilizza questi test per determinare se le policy associate ai certificati dei dispositivi seguono le best practice standard.

Al momento l'over MQTT non WebSocket è supportato.

"Le policy allegate al certificato del dispositivo non contengono caratteri jolly"

Verifica se le policy di autorizzazione associate a un dispositivo seguono le procedure consigliate e non concedono al dispositivo più autorizzazioni del necessario.

Definizione del test case API:

> **Note**
>
> `EXECUTION_TIMEOUT` dispone di un valore predefinito di 1 minuto. Si consiglia di impostare un timeout di almeno 30 secondi.

```
"tests":[
   {
        "name":"my_security_device_policies",
        "configuration": {
            // optional:
            "EXECUTION_TIMEOUT":"60"    // in seconds
        },
        "test": {
            "id": "Security_Device_Policies",
            "version": "0.0.0"
        }
    }
]
```

Test di lunga durata

I test di lunga durata sono una nuova suite di test che monitorano il comportamento di un dispositivo quando funziona per periodi di tempo più lunghi. Rispetto all'esecuzione di singoli test che si concentrano sui comportamenti specifici di un dispositivo, il test di lunga durata esamina il comportamento del dispositivo in una varietà di scenari del mondo reale nel corso del ciclo di vita del dispositivo. Device Advisor orchestra i test nell'ordine più efficiente possibile. Il test genera risultati e log, incluso un log di riepilogo con metriche utili sulle prestazioni del dispositivo durante il test.

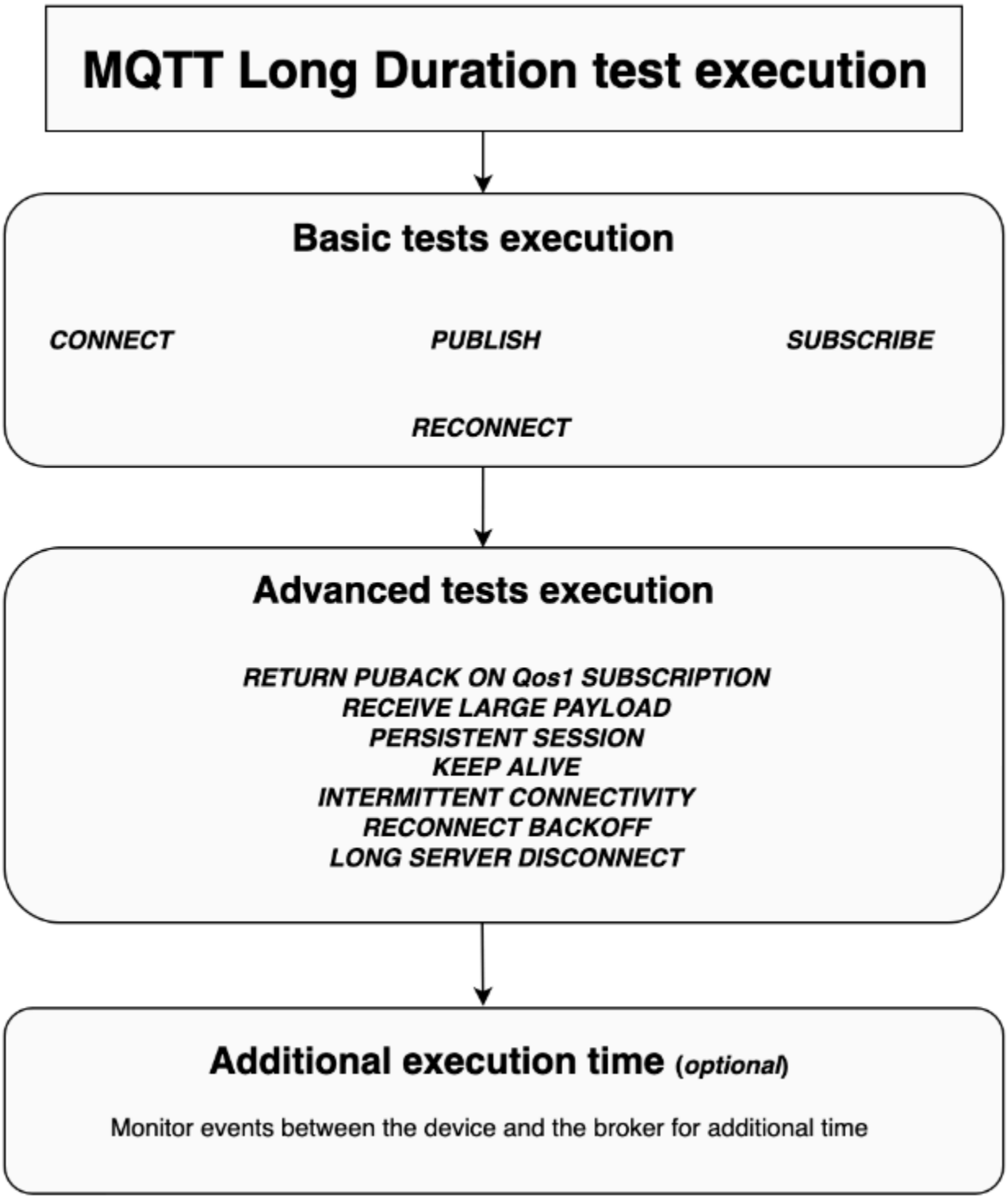
Caso di test di lunga durata MQTT

Nel test case di lunga durata MQTT, il comportamento del dispositivo viene inizialmente osservato in scenari di casi di successo come MQTT Connect, Subscribe, Publish e Reconnect. Quindi, il

dispositivo viene osservato in diversi scenari di errore complessi come MQTT Reconnect Backoff, Long Server Disconnect e Intermittent Connectivity.

Flusso di esecuzione del test case di lunga durata MQTT

L'esecuzione di un test case di lunga durata MQTT comprendere tre fasi:


MQTT Long Duration test execution
Basic tests execution
CONNECT
PUBLISH
SUBSCRIBE
RECONNECT
Advanced tests execution
RETURN PUBACK ON Qos1 SUBSCRIPTION
RECEIVE LARGE PAYLOAD
PERSISTENT SESSION
KEEP ALIVE
INTERMITTENT CONNECTIVITY
RECONNECT BACKOFF
LONG SERVER DISCONNECT
Additional execution time (optional)
Monitor events between the device and the broker for additional time

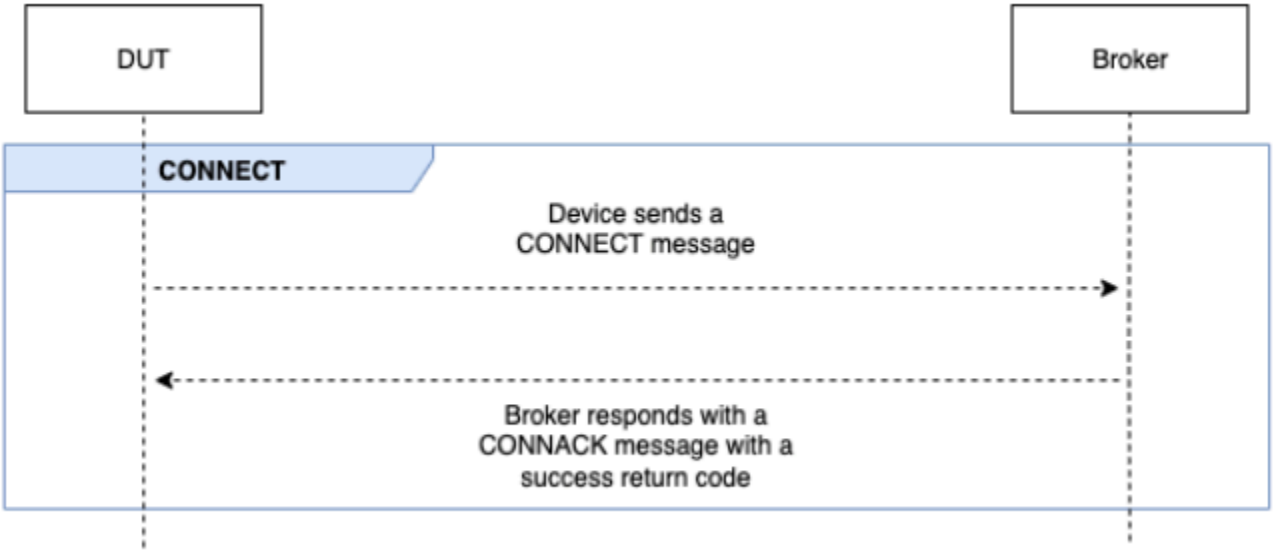
Esecuzione di test di base

In questa fase, il test case esegue semplici test in parallelo. Il test verifica se il dispositivo dispone delle operazioni selezionate nella configurazione.

Il set di test di base può includere quanto specificato di seguito, in base alle operazioni selezionate.

CONNECT

Questo scenario convalida se il dispositivo è in grado di stabilire una connessione con il broker.

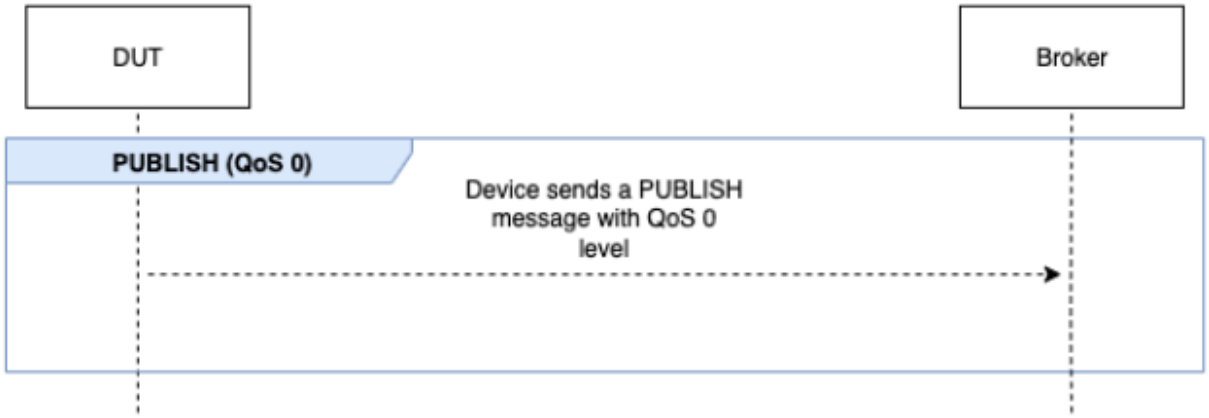


PUBLISH

Questo scenario convalida se il dispositivo pubblica tramite il broker.

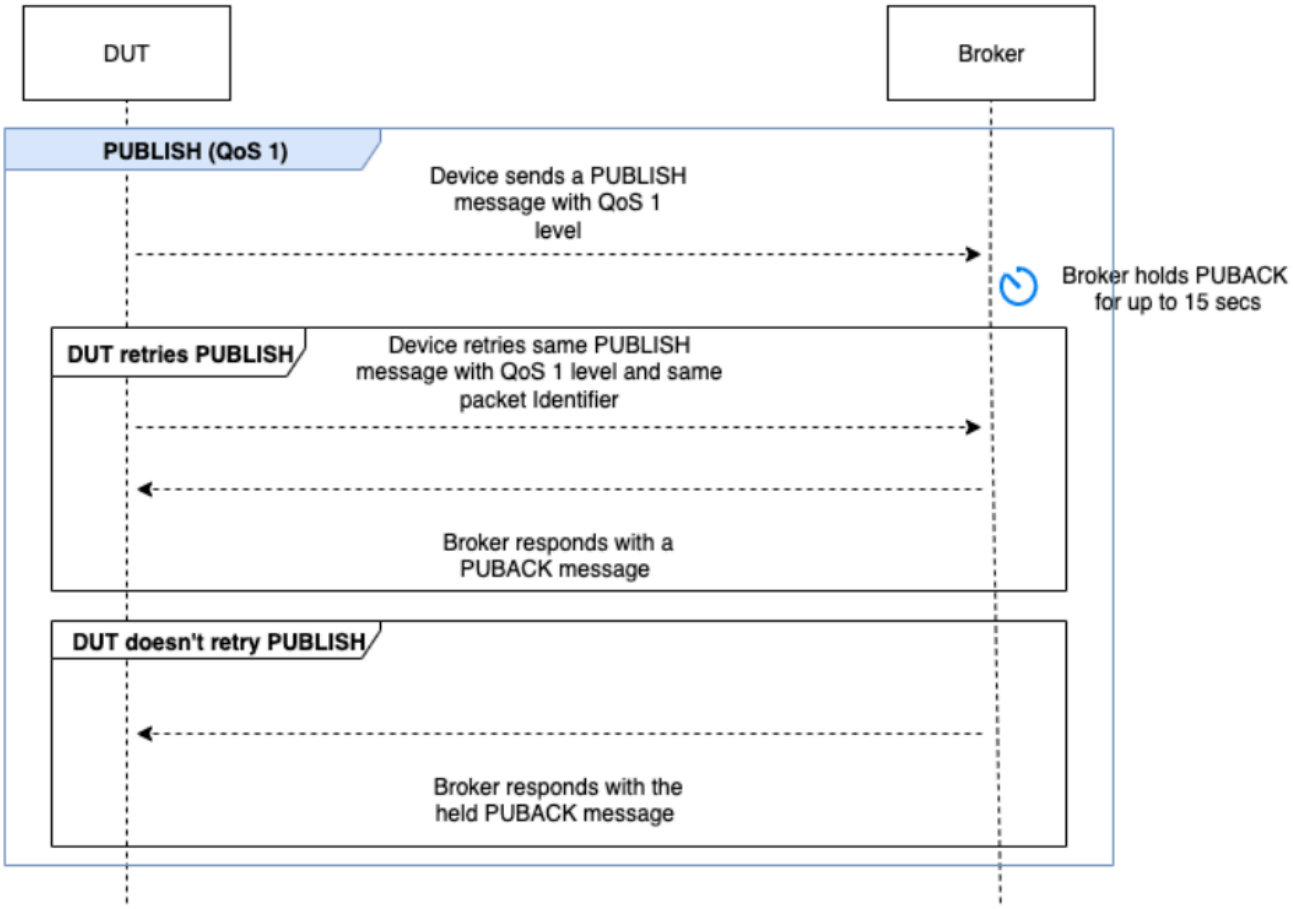
QoS 0

Questo test case convalida se il dispositivo invia un messaggio `PUBLISH` al broker durante una pubblicazione con QoS 0. Il test non attende che il messaggio `PUBACK` venga ricevuto dal dispositivo.



QoS 1

In questo test case, si prevede che il dispositivo invii due messaggi `PUBLISH` al broker con QoS 1. Dopo il primo messaggio `PUBLISH`, il broker attende per un massimo di 15 secondi prima di rispondere. Il dispositivo deve riprovare a inviare il messaggio `PUBLISH` originale con lo stesso identificatore di pacchetto entro 15 secondi. In tal caso, il broker risponde con un messaggio `PUBACK` e il test viene convalidato. Se il dispositivo non riprova a inviare `PUBLISH`, il `PUBACK` originale viene inviato al dispositivo e il test viene contrassegnato come Pass with warnings (Superato con avvertenze), insieme a un messaggio di sistema. Durante l'esecuzione del test, se il dispositivo perde la connessione e si riconnette, lo scenario di test viene ripristinato senza errori e il dispositivo deve eseguire nuovamente i passaggi dello scenario di test.

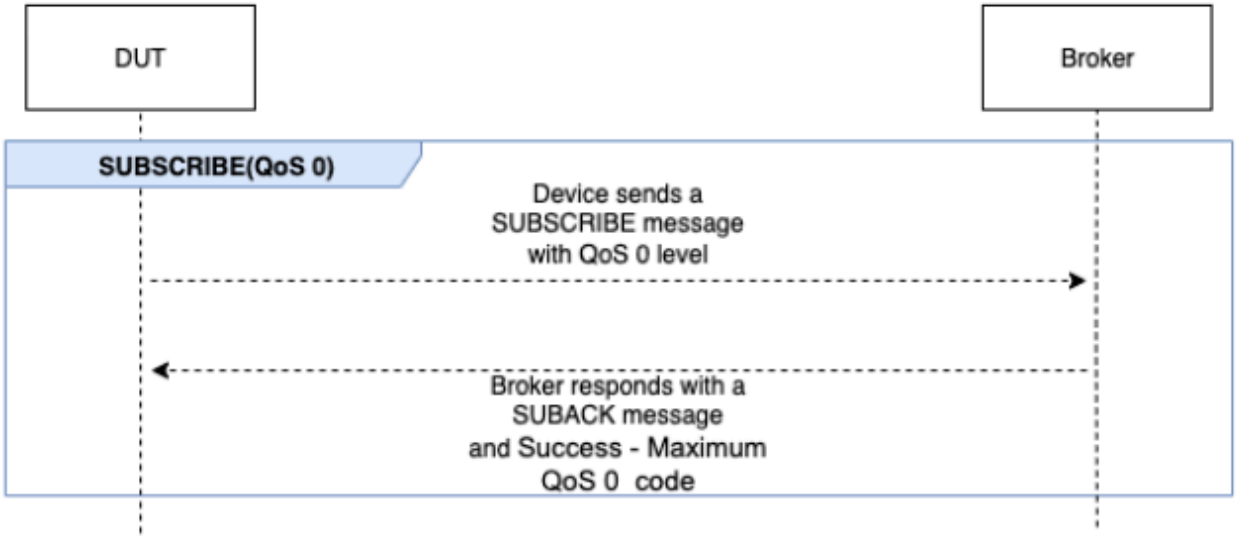


SUBSCRIBE

Questo scenario convalida se il dispositivo esegue la sottoscrizione tramite il broker.

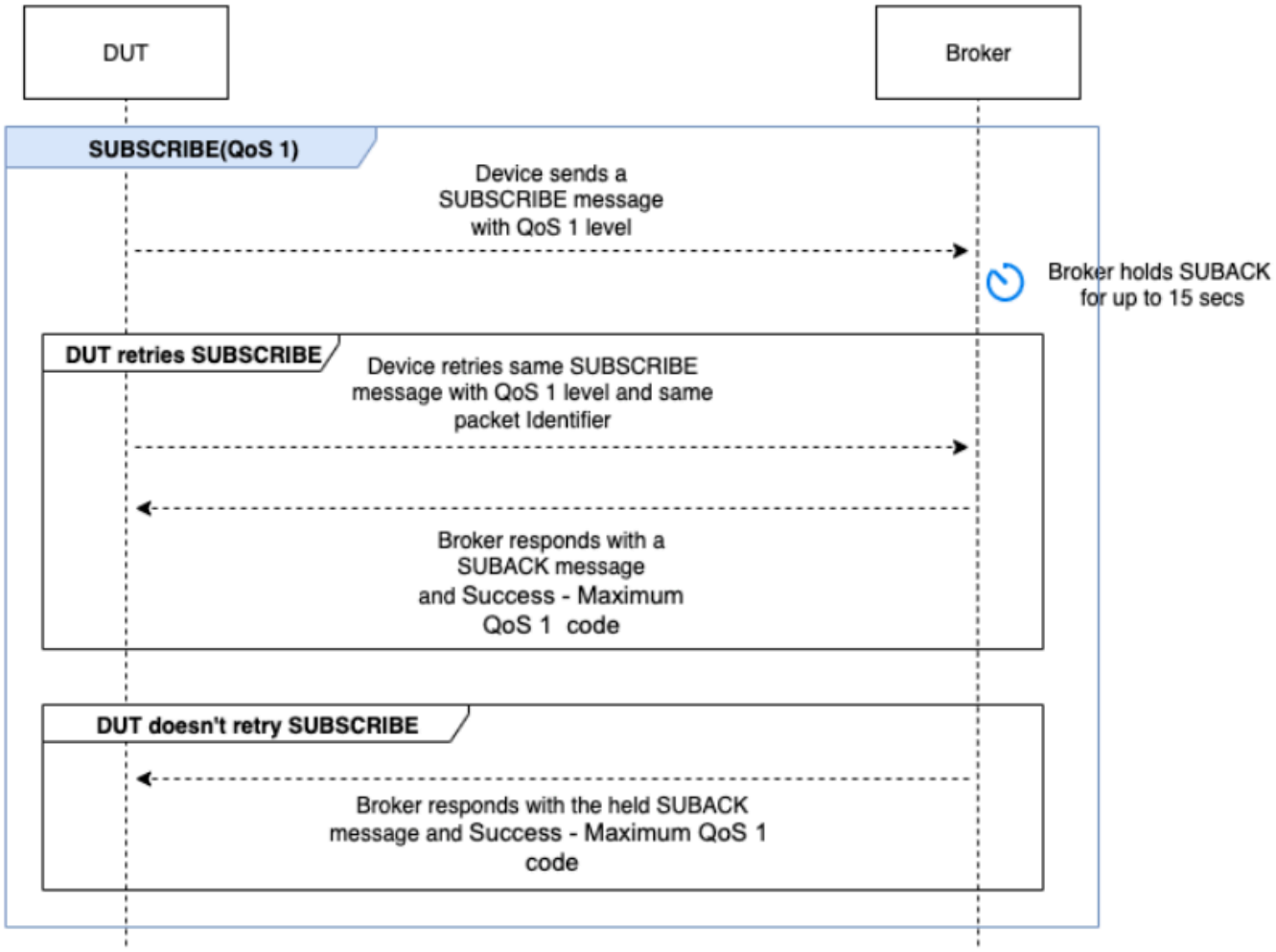
QoS 0

Questo test case convalida se il dispositivo invia un messaggio `SUBSCRIBE` al broker durante una sottoscrizione con QoS 0. Il test non attende che il dispositivo riceva un messaggio SUBACK.



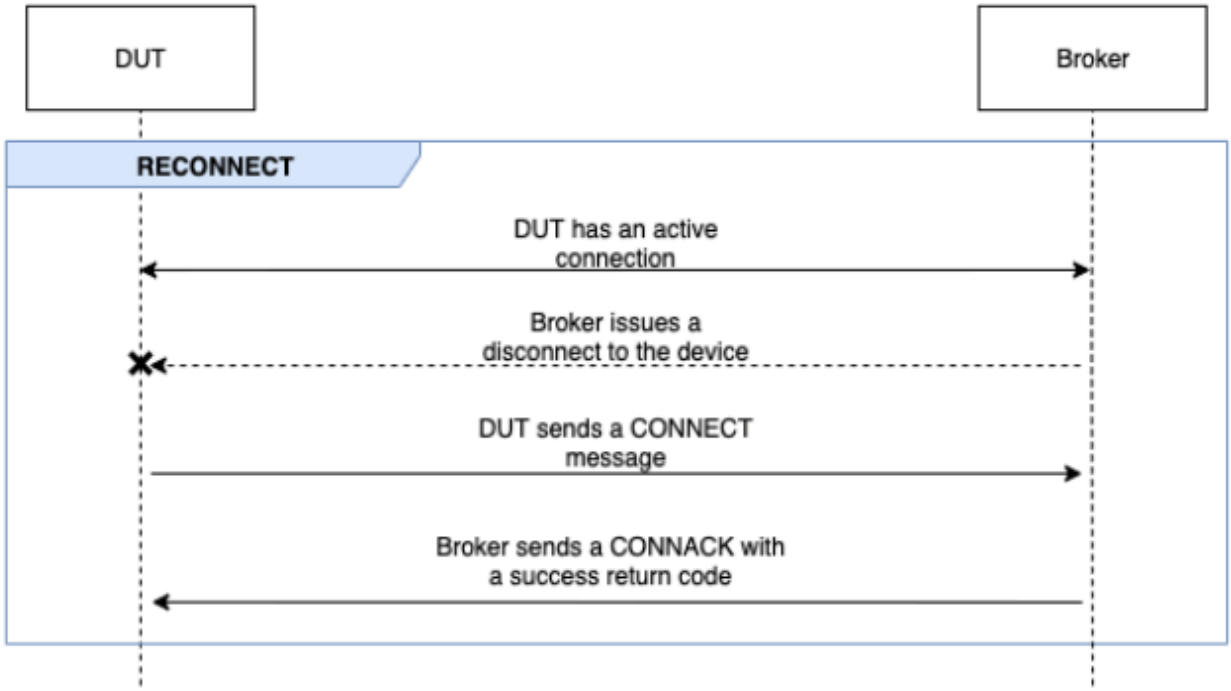
QoS 1

In questo test case, si prevede che il dispositivo invii due messaggi `SUBSCRIBE` al broker con QoS 1. Dopo il primo messaggio `SUBSCRIBE`, il broker attende per un massimo di 15 secondi prima di rispondere. Il dispositivo deve riprovare a inviare il messaggio `SUBSCRIBE` originale con lo stesso identificatore di pacchetto entro 15 secondi. In tal caso, il broker risponde con un messaggio `SUBACK` e il test viene convalidato. Se il dispositivo non riprova a inviare `SUBSCRIBE`, il `SUBACK` originale viene inviato al dispositivo e il test viene contrassegnato come Pass with warnings (Superato con avvertenze), insieme a un messaggio di sistema. Durante l'esecuzione del test, se il dispositivo perde la connessione e si riconnette, lo scenario di test viene ripristinato senza errori e il dispositivo deve eseguire nuovamente i passaggi dello scenario di test.



RECONNECT

Questo scenario convalida se il dispositivo si riconnette al broker dopo che il dispositivo è stato disconnesso da una connessione. Device Advisor non disconnette il dispositivo se in precedenza si è connesso più di una volta durante la suite di test. Invece, contrassegnerà il test come Pass (Superato).



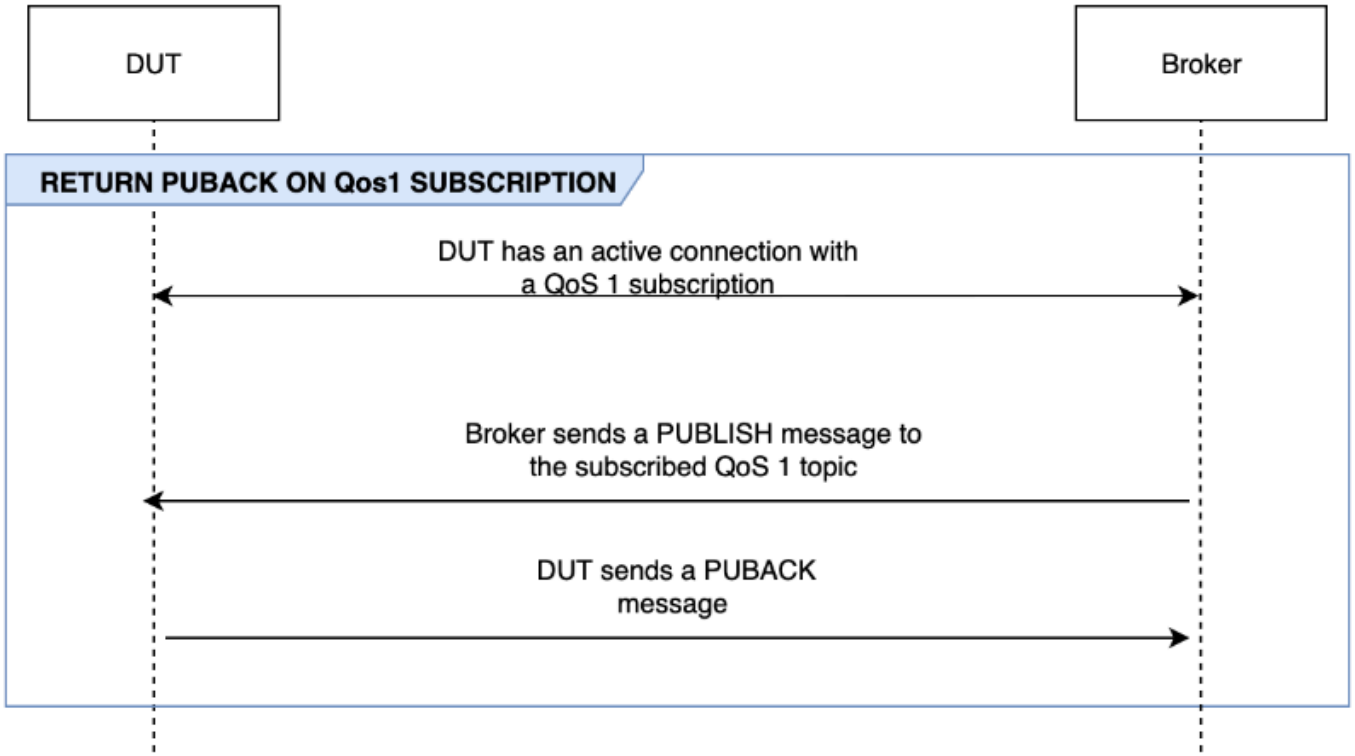
Esecuzione di test avanzati

In questa fase, il test case esegue test più complessi in serie per verificare se il dispositivo segue le best practice. Questi test avanzati sono disponibili per la selezione e possono essere disattivati se non necessari. Ogni test avanzato dispone del proprio valore di timeout in base a ciò che lo scenario richiede.

RETURN PUBACK ON QoS 1 SUBSCRIPTION

> **Note**
>
> Selezionare questo scenario solo se il dispositivo è in grado di eseguire sottoscrizioni QoS 1.

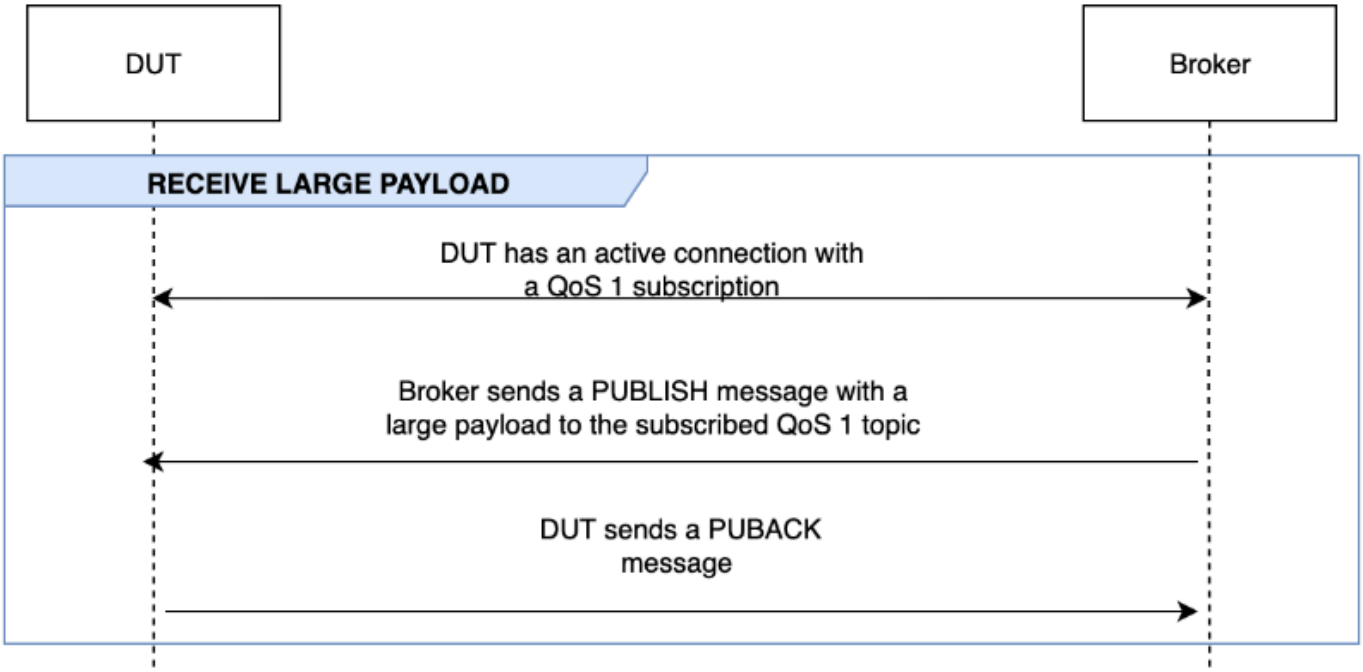
Questo scenario convalida se il dispositivo restituisce un messaggio `PUBACK` dopo che esegue la sottoscrizione a un argomento e riceve un messaggio `PUBLISH` dal broker.



RECEIVE LARGE PAYLOAD

> **Note**
>
> Selezionare questo scenario se il dispositivo è in grado di eseguire sottoscrizioni QoS 1.

Questo scenario convalida se il dispositivo risponde con un messaggio `PUBACK` dopo aver ricevuto un messaggio `PUBLISH` dal broker per un argomento QoS 1 con un payload di grandi dimensioni. Il formato di payload previsto può essere configurato utilizzando l'opzione `LONG_PAYLOAD_FORMAT`.



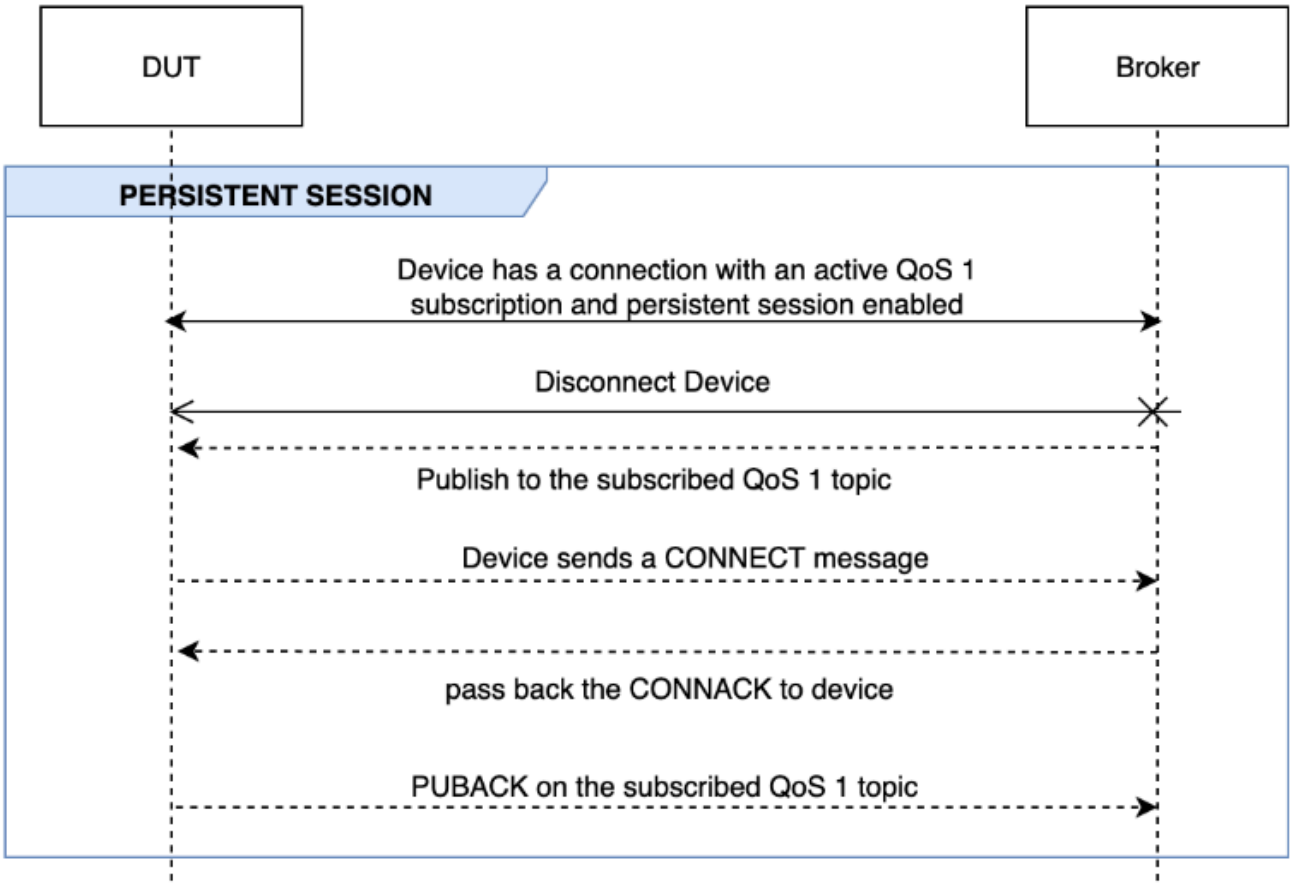
PERSISTENT SESSION

> **Note**
>
> Selezionare questo scenario se il dispositivo è in grado di eseguire sottoscrizioni QoS 1 e mantenere una sessione persistente.

Questo scenario convalida il comportamento del dispositivo nel mantenimento di sessioni persistenti. Il test convalida quando vengono soddisfatte le seguenti condizioni:

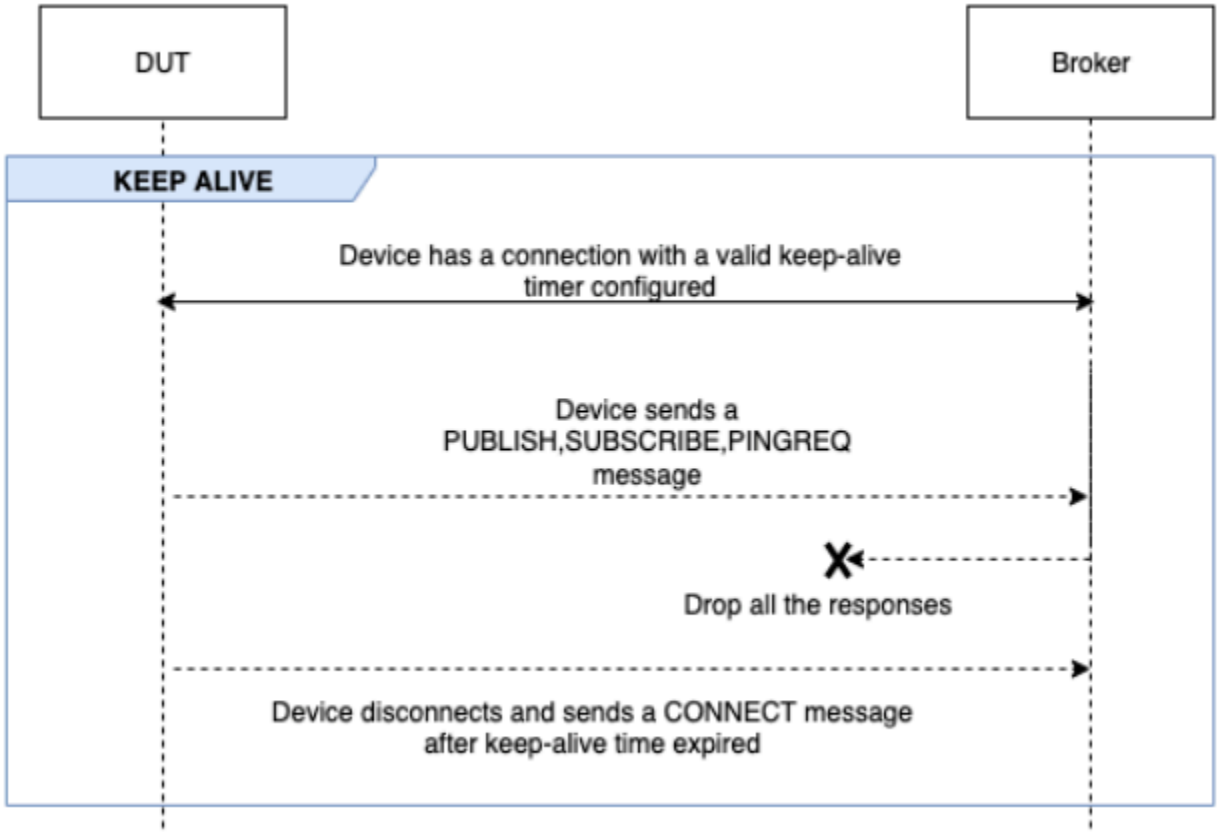
- Il dispositivo si connette al broker con una sottoscrizione QoS 1 attivo e sessioni persistenti abilitate.
- Il dispositivo si disconnette dal broker durante la sessione.
- Il dispositivo si riconnette al broker e riprende le sottoscrizioni agli argomenti trigger senza effettuare nuovamente la sottoscrizione a questi argomenti in maniera esplicita.
- Il dispositivo riceve i messaggi archiviati dal broker per gli argomenti sottoscritti e funziona come previsto.

Per ulteriori informazioni sulle sessioni AWS IoT persistenti, vedere Utilizzo delle sessioni persistenti MQTT.



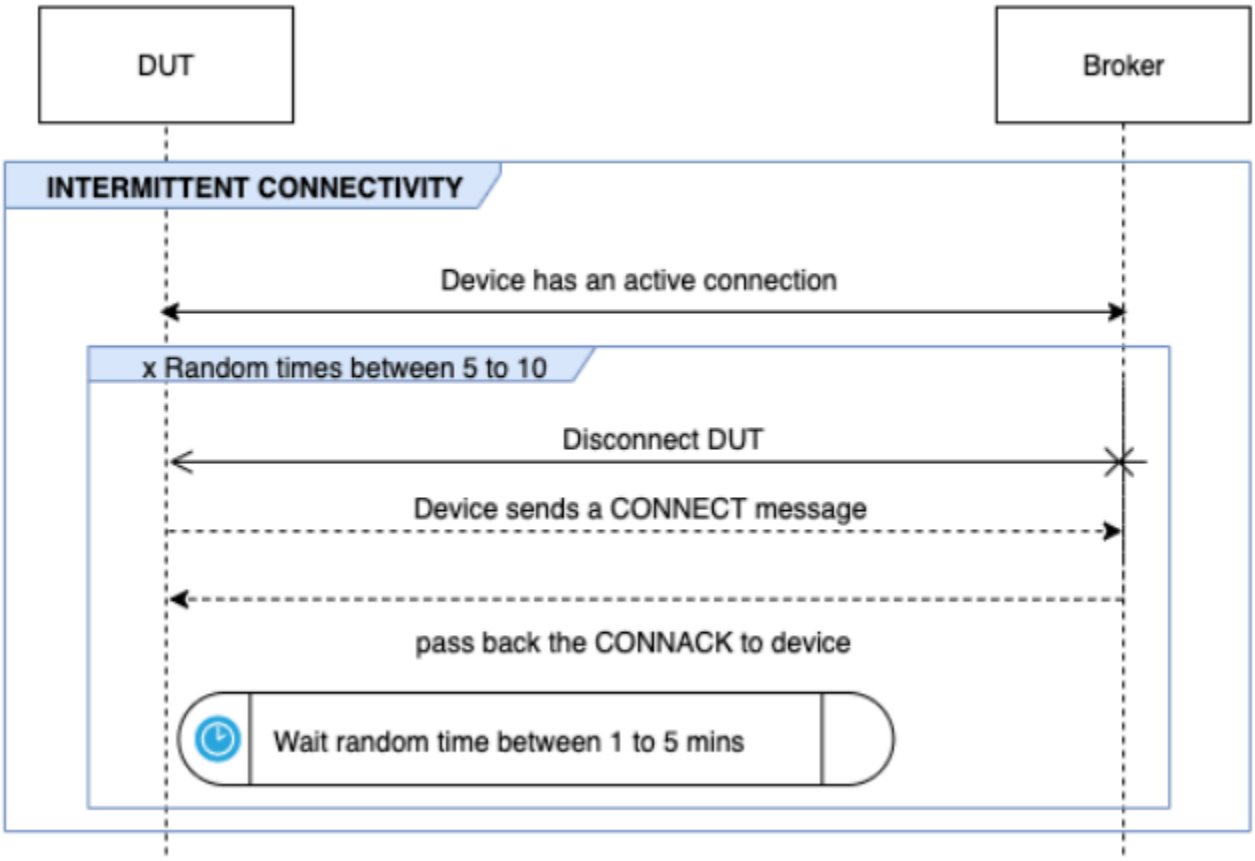
KEEP ALIVE

Questo scenario cpmvalida se il dispositivo si disconnette dopo che non riceve una risposta ping dal broker. È necessario che nella connessione sia configurato timer keep-alive valido. Come parte di questo test, il broker blocca tutte le risposte inviate per i messaggi `PUBLISH`, `SUBSCRIBE` e `PINGREQ`. Convalida anche se il dispositivo sottoposto a test disconnette la connessione MQTT.



INTERMITTENT CONNECTIVITY

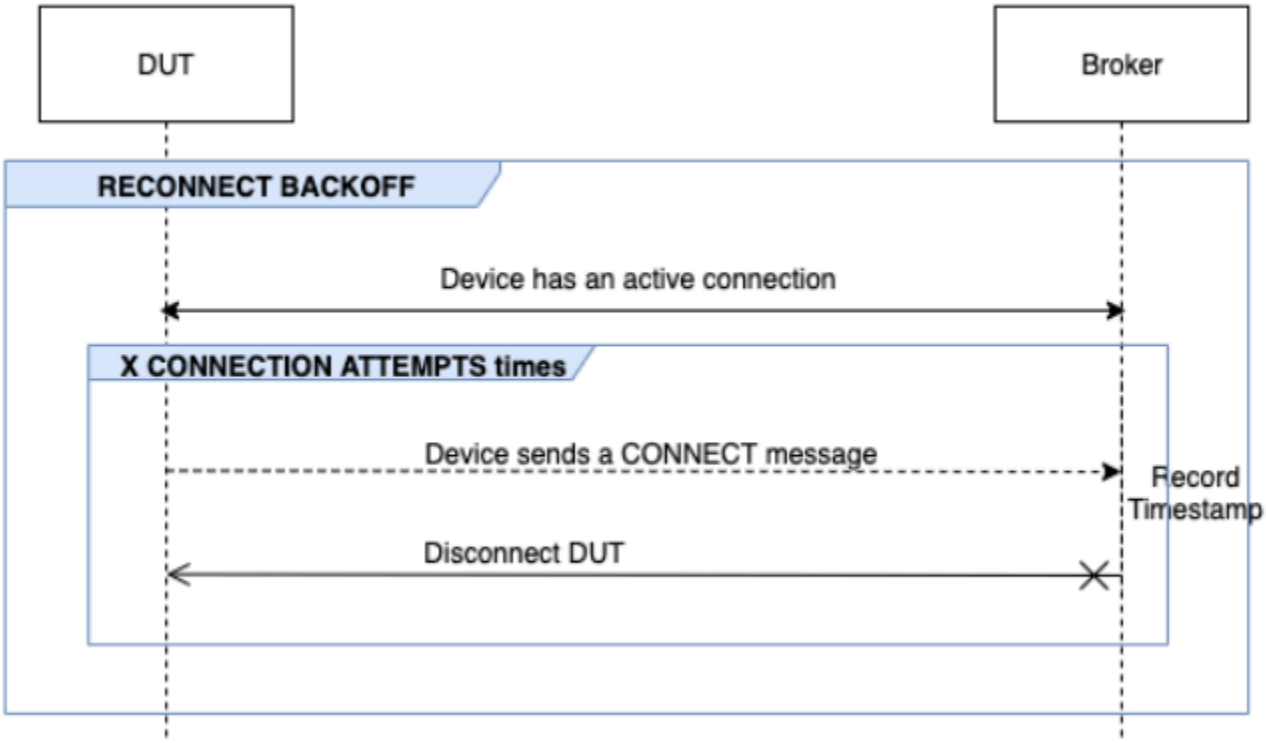
Questo scenario convalida se il dispositivo può riconnettersi al broker dopo che scollega il dispositivo a intervalli casuali per un periodo di tempo casuale.



RECONNECT BACKOFF

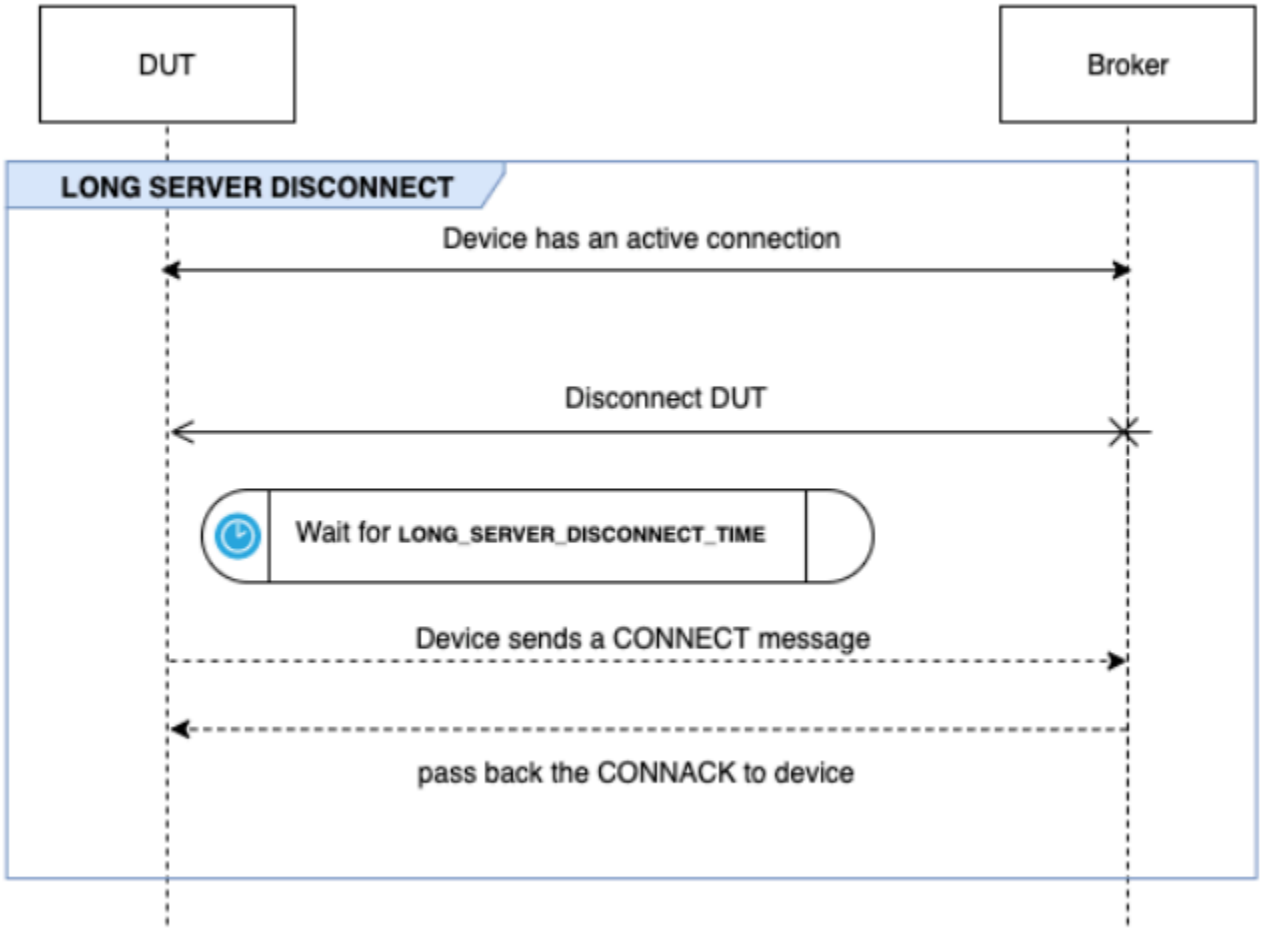
Questo scenario convalida se il dispositivo dispone di un meccanismo di backoff implementato quando il broker si disconnette più volte. Device Advisor segnala il tipo di backoff come esponenziale, jitter, lineare o costante. Il numero di tentativi di backoff è configurabile utilizzando l'opzione `BACKOFF_CONNECTION_ATTEMPTS`. Il valore predefinito è 5. Il valore è configurabile tra 5 e 10.

Per superare questo test, si consiglia di implementare il meccanismo Exponential Backoff And Jitter (Backoff esponenziale e jitter) sul dispositivo in fase di test.



LONG SERVER DISCONNECT

Questo scenario convalida se il dispositivo può riconnettersi dopo che il broker scollega il dispositivo per un lungo periodo di tempo (fino a 120 minuti). L'ora di disconnessione del server può essere configurata utilizzando l'opzione `LONG_SERVER_DISCONNECT_TIME`. Il valore predefinito è di 120 minuti. Questo valore è configurabile da 30 a 120 minuti.



Tempo di esecuzione aggiuntivo

Il tempo di esecuzione aggiuntivo è il tempo che il test attende dopo il completamento di tutti i test precedenti e prima di terminare il test case. I clienti utilizzano questo periodo di tempo aggiuntivo per monitorare e registrare tutte le comunicazioni tra il dispositivo e il broker. Il tempo di esecuzione aggiuntivo può essere configurato utilizzando l'opzione `ADDITIONAL_EXECUTION_TIME`. Per impostazione predefinita, questa opzione è impostata su 0 minuti e può essere compresa tra 0 e 120 minuti.

Opzioni di configurazione del test di lunga durata MQTT

Tutte le opzioni di configurazione fornite per il test di lunga durata MQTT sono opzionali. Sono disponibili le seguenti opzioni:

OPERAZIONI

L'elenco delle operazioni eseguite dal dispositivo, ad esempio `CONNECT`, `PUBLISH` e `SUBSCRIBE`. Il test case esegue scenari in base alle operazioni specificate. Le operazioni non specificate sono considerate valide.

```
{
"OPERATIONS": ["PUBLISH", "SUBSCRIBE"]
//by default the test assumes device can CONNECT
}
```

SCENARI

In base alle operazioni selezionate, il test case esegue scenari per convalidare il comportamento del dispositivo. Esistono due tipi di scenari:

- Gli scenari di base sono semplici test che verificano se il dispositivo è in grado di eseguire le operazioni selezionate in precedenza come parte della configurazione. Questi sono preselezionati in base alle operazioni specificate nella configurazione. Non è richiesto alcun ulteriore input nella configurazione.
- Gli scenari avanzati sono scenari più complessi che vengono eseguiti rispetto al dispositivo per convalidare se il dispositivo segue le best practice quando soddisfa le condizioni del mondo reale. Questi sono opzionali e possono essere superati come una serie di scenari per l'input di configurazione della suite di test.

```
{
    "SCENARIOS": [      // list of advanced scenarios
                "PUBACK_QOS_1",
                "RECEIVE_LARGE_PAYLOAD",
                "PERSISTENT_SESSION",
                "KEEP_ALIVE",
                "INTERMITTENT_CONNECTIVITY",
                "RECONNECT_BACK_OFF",
                "LONG_SERVER_DISCONNECT"
    ]
}
```

BASIC_TESTS_EXECUTION_TIME_OUT:

Il periodo di tempo massimo che il test case attende per il completamento di tutti i test di base. Il valore predefinito è di 60 minuti. Questo valore è configurabile da 30 a 120 minuti.

LONG_SERVER_DISCONNECT_TIME:

Il tempo richiesto dal test case per scollegare e ricollegare il dispositivo durante il test Disconnessione server di lunga durata. Il valore predefinito è di 60 minuti. Questo valore è configurabile da 30 a 120 minuti.

ADDITIONAL_EXECUTION_TIME:

La configurazione di questa opzione fornisce una finestra temporale dopo il completamento di tutti i test, per monitorare gli eventi tra il dispositivo e il broker. Il valore predefinito è di 0 minuti. Questo valore è configurabile da 0 a 120 minuti.

BACKOFF_CONNECTION_ATTEMPTS:

Questa opzione configura il numero di volte in cui il dispositivo viene disconnesso dal test case. Viene utilizzato dal test Riconnessione backoff. Il valore predefinito è 5. Questo valore è configurabile da 5 a 10.

LONG_PAYLOAD_FORMAT:

Il formato del payload del messaggio atteso dal dispositivo quando il test case viene pubblicato in un argomento QoS 1 sottoscritto dal dispositivo.

Definizione del test case API:

```
{
"tests":[
   {
      "name":"my_mqtt_long_duration_test",
      "configuration": {
         // optional
         "OPERATIONS": ["PUBLISH", "SUBSCRIBE"],
         "SCENARIOS": [
            "LONG_SERVER_DISCONNECT",
            "RECONNECT_BACK_OFF",
            "KEEP_ALIVE",
            "RECEIVE_LARGE_PAYLOAD",
            "INTERMITTENT_CONNECTIVITY",
            "PERSISTENT_SESSION",
         ],
         "BASIC_TESTS_EXECUTION_TIMEOUT": 60, // in minutes (60 minutes by default)
         "LONG_SERVER_DISCONNECT_TIME": 60,   // in minutes (120 minutes by default)
         "ADDITIONAL_EXECUTION_TIME": 60,     // in minutes (0 minutes by default)
```

```
         "BACKOFF_CONNECTION_ATTEMPTS": "5",
         "LONG_PAYLOAD_FORMAT":"{"message":"${payload}"}"
      },
      "test":{
         "id":"MQTT_Long_Duration",
         "version":"0.0.0"
      }
   }
 ]
}
```

Log di riepilogo del test case di lunga durata MQTT

Il test case di lunga durata MQTT viene eseguito più a lungo rispetto ai normali test case. Durante l'esecuzione, viene fornito un log di riepilogo separato, contenente eventi importanti come connessioni dispositivo, pubblicazione e sottoscrizione. I dettagli includono l'oggetto del test, gli elementi esclusi dal test e gli eventuali errori. Alla fine del log, il test include un riepilogo di tutti gli eventi che si sono verificati durante l'esecuzione del test case. Questo include:

- Timer Keep Alive configurato sul dispositivo.
- Flag sessione persistente configurato sul dispositivo.
- Il numero di connessioni del dispositivo durante l'esecuzione del test.
- Il tipo di riconnessione backoff del dispositivo, se convalidato per il test di riconnessione backoff.
- Gli argomenti su cui il dispositivo ha pubblicato durante l'esecuzione del test case.
- Gli argomenti sottoscritti dal dispositivo, durante l'esecuzione del test case.

AWS IoT Core Ubicazione del dispositivo

Prima di utilizzare la funzione di localizzazione del AWS IoT Core dispositivo, consulta i Termini e condizioni di questa funzione. Tieni presente che AWS potrebbe trasmettere i parametri della richiesta di ricerca di geolocalizzazione, come i dati sulla posizione utilizzati per eseguire le ricerche, e altre informazioni al fornitore di dati terzo prescelto, che potrebbe essere diverso da Regione AWS quello che stai utilizzando attualmente. Il provider terzo e il solutore da utilizzare vengono scelti in base al payload di input ricevuto. Per ulteriori informazioni, consultare Termini del servizio AWS.

Usa AWS IoT Core Device Location per testare la posizione dei tuoi dispositivi IoT utilizzando solutori di terze parti. I risolutori sono algoritmi forniti da fornitori di terze parti che risolvono i dati di misurazione e stimano la posizione del dispositivo. Identificando la posizione dei dispositivi, è possibile monitorarli ed eseguirne il debug in loco per risolvere eventuali problemi.

I dati di misurazione raccolti da varie fonti vengono risolti e le informazioni di geolocalizzazione vengono riportate come un payload GeoJSON. Il formato GeoJSON viene utilizzato per codificare strutture di dati geografici. Il payload contiene le coordinate di latitudine e longitudine della posizione del dispositivo, che si basano sul sistema di coordinate del World Geodetic System (). WGS84

Argomenti

- Tipi di misurazione e risolutori
- Come funziona AWS IoT Core Device Location
- Come usare Device Location AWS IoT Core
- Risoluzione della posizione di dispositivi IoT
- Risoluzione della posizione del dispositivo utilizzando gli argomenti MQTT di AWS IoT Core Device Location
- Risolutori di posizione e payload del dispositivo

Tipi di misurazione e risolutori

AWS IoT Core Device Location collabora con fornitori terzi per risolvere i dati di misurazione e fornire una posizione stimata del dispositivo. Nella tabella seguente vengono visualizzati i tipi di misurazione e i risolutori di posizione di terze parti, nonché le informazioni sui dispositivi supportati.

Per informazioni sui dispositivi LoRa WAN e sulla configurazione della posizione dei dispositivi per essi, vedere Configurazione della posizione delle LoRa risorse WAN.

> **Note**
>
> I dispositivi IoT generici e i dispositivi Sidewalk possono utilizzare gli argomenti MQTT sulla posizione dei dispositivi per ottenere informazioni sulla posizione. Per i tipi di misurazione degli indirizzi Wi-Fi, cellulare e IP, se i dispositivi pubblicano i dati di misurazione negli argomenti riservati nel formato GeoJSON definito AWS IoT Core , Device Location può risolvere la posizione del dispositivo. Per il tipo di misurazione GNSS, il dispositivo deve disporre del LR11xx chip per scansionare i dati di misurazione per ottenere le informazioni sulla posizione risolte utilizzando il risolutore GNSS. Per informazioni su come ottenere informazioni sulla posizione dei dispositivi LoRa WAN, vedere Configurazione della posizione per le risorse LoRa WAN nella documentazione Wireless.AWS IoT

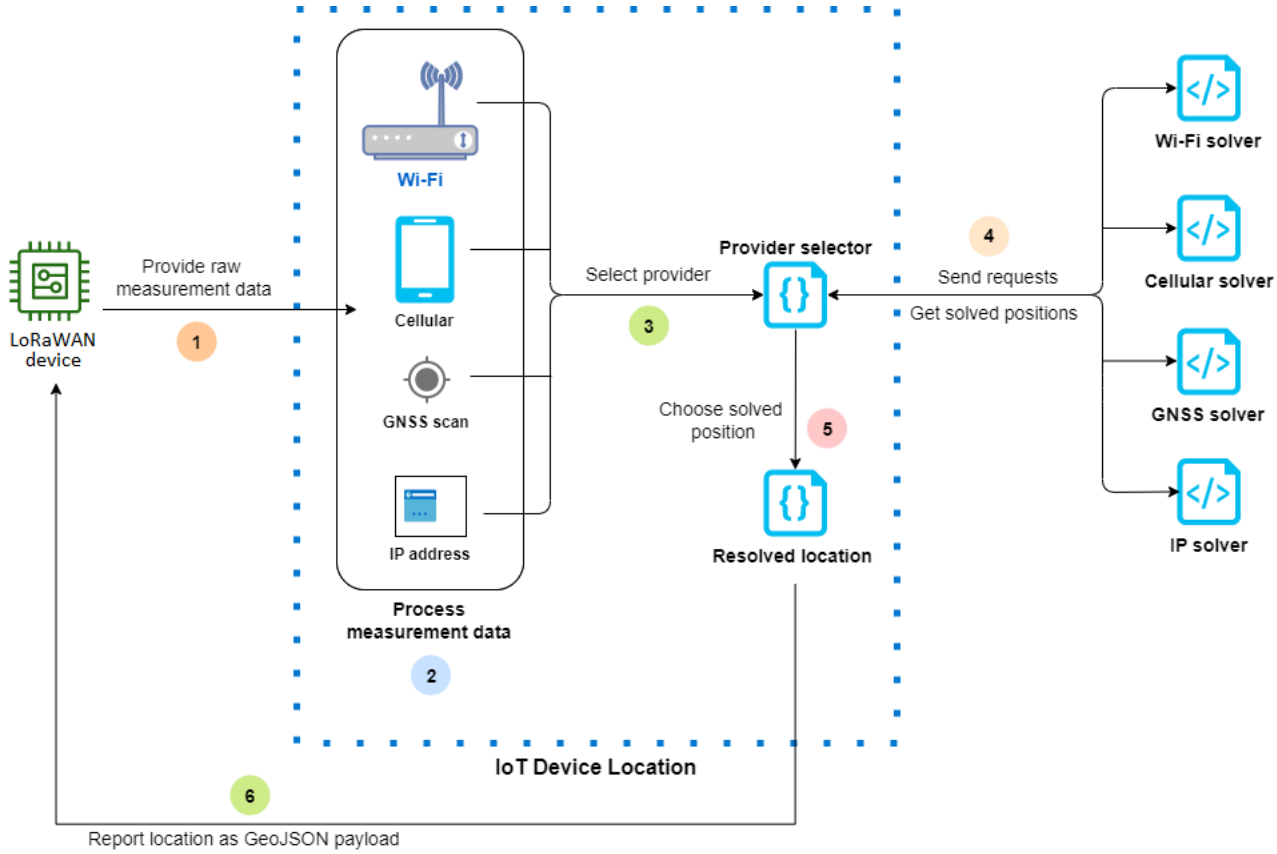
Tipi di misurazione e risolutori

Tipo di misurazione	Risolutori di terze parti	Dispositivi supportati
Punti di accesso Wi-Fi	Risolutore basato su Wi-Fi	Dispositivi IoT generici, LoRa WAN e dispositivi Amazon Sidewalk
Torri radio della rete cellulare: dati GSM, LTE, CDMA, SCDMA, WCMDA e TD-SCDMA	Risolutore basato su rete cellulare	Dispositivi IoT generici, LoRa WAN e dispositivi Amazon Sidewalk
IP address (Indirizzo IP)	Risolutore di ricerca inversa IP	Qualsiasi dispositivo IoT che si connette tramite TCP/IP
Dati di scansione GNSS (messaggi NAV)	Risolutore GNSS	Dispositivi IoT generici, LoRa WAN e dispositivi Amazon Sidewalk
Bluetooth Low Energy (BLE)	Risolutore basato su BLE	Dispositivi Amazon Sidewalk

Per ulteriori informazioni sui risolutori di posizione ed esempi che mostrano il payload del dispositivo per i vari tipi di misurazione, consulta Risolutori di posizione e payload del dispositivo.

Come funziona AWS IoT Core Device Location

Il diagramma seguente mostra come AWS IoT Core Device Location raccoglie i dati di misurazione e risolve le informazioni sulla posizione dei dispositivi.



I passaggi seguenti mostrano come funziona AWS IoT Core Device Location.

1. Ricezione dei dati di misurazione

   I dati di misurazione non elaborati correlati alla posizione del dispositivo vengono innanzitutto inviati dal dispositivo. I dati di misurazione sono specificati come un payload JSON.

2. Elaborazione dei dati di misurazione

   I dati di misurazione vengono elaborati e AWS IoT Core Device Location sceglie i dati di misurazione da utilizzare, che possono essere informazioni Wi-Fi, cellulare, scansione GNSS o informazioni sull'indirizzo IP.

3. Scelta del risolutore

   Il risolutore di terze parti viene scelto in base ai dati di misurazione. Ad esempio, se i dati di misurazione contengono informazioni su Wi-Fi e indirizzo IP, viene scelto il risolutore Wi-Fi e il risolutore di ricerca inversa IP.

4. Ottenere la posizione risolta

   Una richiesta API viene inviata ai fornitori di solver che richiedono di risolvere la posizione. AWS IoT Core Device Location ottiene quindi le informazioni di geolocalizzazione stimate dai solutori.

5. Scelta della posizione risolta

   Le informazioni sulla posizione risolte e la relativa precisione vengono confrontate e AWS IoT Core Device Location sceglie i risultati di geolocalizzazione con la massima precisione.

6. Generazione delle informazioni sulla posizione

   Le informazioni di geolocalizzazione vengono inviate all'utente come un payload GeoJSON. Il payload contiene le coordinate geografiche, le informazioni WGS84 sulla precisione, i livelli di confidenza e il timestamp in cui è stata ottenuta la posizione risolta.

Come usare Device Location AWS IoT Core

I passaggi seguenti mostrano come utilizzare AWS IoT Core Device Location.

1. Fornire dati di misurazione

   Specificare i dati di misurazione grezzi relativi alla posizione del dispositivo come un payload JSON. Per recuperare i dati di misurazione del payload, accedi ai registri del dispositivo o utilizza CloudWatch Logs e copia le informazioni sui dati del payload. Il payload JSON deve contenere uno o più tipi di dati di misurazione. Per esempi che mostrano il formato di payload per vari risolutori, consulta Risolutori di posizione e payload del dispositivo.

2. Risolvere le informazioni sulla posizione

   Utilizzando la pagina Device Location nella AWS IoT console o il funzionamento dell'GetPositionEstimateAPI, trasmetti i dati di misurazione del payload e risolvi la posizione del dispositivo. AWS IoT Core Device Location sceglie quindi il solutore con la massima precisione e riporta la posizione del dispositivo. Per ulteriori informazioni, consulta Risoluzione della posizione di dispositivi IoT.

3. Copiare le informazioni sulla posizione

   Verifica le informazioni di geolocalizzazione che sono state risolte da AWS IoT Core Device Location e riportate come payload GeoJSON. Puoi copiare il payload per utilizzarlo con le tue applicazioni e altri file. Servizio AWS Ad esempio, puoi inviare i dati della tua posizione geografica ad Amazon Location Service utilizzando l'azione della Location (Ubicazione) AWS IoT regola.

I seguenti argomenti mostrano come utilizzare AWS IoT Core Device Location ed esempi di payload di localizzazione del dispositivo.

- Risoluzione della posizione di dispositivi IoT
- Risolutori di posizione e payload del dispositivo

Risoluzione della posizione di dispositivi IoT

Utilizzate AWS IoT Core Device Location per decodificare i dati di misurazione dai dispositivi e risolvere la posizione del dispositivo utilizzando solutori di terze parti. La posizione risolta viene generata come un payload GeoJSON con le coordinate geografiche e le informazioni sulla precisione. Puoi determinare la posizione del dispositivo dalla AWS IoT console, dall'API AWS IoT wireless o. AWS CLI

Argomenti

- Risoluzione della posizione del dispositivo (console)
- Risoluzione della posizione del dispositivo (API)
- Risoluzione dei problemi relativi alla risoluzione della posizione

Risoluzione della posizione del dispositivo (console)

Per risolvere la posizione del dispositivo (console)

1. Vai alla pagina di localizzazione del dispositivo nella AWS IoT console.
2. Ottieni i dati di misurazione del carico utile dai registri del dispositivo o dai CloudWatch registri e inseriscili nella sezione Risolvi la posizione tramite payload.

Il codice riportato di seguito mostra un payload JSON di esempio. Il payload contiene dati di misurazione della rete cellulare e Wi-Fi. Se il payload contiene altri tipi di dati di misurazione, verrà utilizzato il risolutore con la precisione maggiore. Per ulteriori informazioni ed esempi di payload, consulta the section called “Risolutori di posizione e payload del dispositivo”.

> **Note**
>
> Il payload JSON deve contenere almeno un tipo di dati di misurazione.

```
{
    "Timestamp": 1664313161,
    "Ip":{
        "IpAddress": "54.240.198.35"
    },
    "WiFiAccessPoints": [{
        "MacAddress": "A0:EC:F9:1E:32:C1",
        "Rss": -77
    }],
    "CellTowers": {
        "Gsm": [{
            "Mcc": 262,
            "Mnc": 1,
            "Lac": 5126,
            "GeranCid": 16504,
            "GsmLocalId": {
                "Bsic": 6,
                "Bcch": 82
            },
            "GsmTimingAdvance": 1,
            "RxLevel": -110,
            "GsmNmr": [{
                "Bsic": 7,
                "Bcch": 85,
                "RxLevel": -100,
                "GlobalIdentity": {
                    "Lac": 1,
                    "GeranCid": 1
                }
            }]
        }],
```

```
        "Wcdma": [{
            "Mcc": 262,
            "Mnc": 7,
            "Lac": 65535,
            "UtranCid": 14674663,
            "WcdmaNmr": [{
                    "Uarfcndl": 10786,
                    "UtranCid": 14674663,
                    "Psc": 149
                },
                {
                    "Uarfcndl": 10762,
                    "UtranCid": 14674663,
                    "Psc": 211
                }
            ]
        }],
        "Lte": [{
            "Mcc": 262,
            "Mnc": 2,
            "EutranCid": 2898945,
            "Rsrp": -50,
            "Rsrq": -5,
            "LteNmr": [{
                    "Earfcn": 6300,
                    "Pci": 237,
                    "Rsrp": -60,
                    "Rsrq": -6,
                    "EutranCid": 2898945
                },
                {
                    "Earfcn": 6300,
                    "Pci": 442,
                    "Rsrp": -70,
                    "Rsrq": -7,
                    "EutranCid": 2898945
                }
            ]
        }]
    }
}
```

3. Per risolvere le informazioni sulla posizione, scegli Resolve (Risolvi).

Le informazioni sulla posizione sono di tipo blob e vengono restituite come un payload che utilizza il formato GeoJSON, ideale per codificare strutture di dati geografici. Il payload contiene:

- Le coordinate geografiche, che includono le informazioni WGS84 sulla latitudine e la longitudine. Potrebbe anche includere un'informazione di altitudine.
- Il tipo di informazioni sulla posizione restituite, ad esempio Point (Punto). Un tipo di posizione del punto rappresenta la posizione come WGS84 latitudine e longitudine, codificata come punto GeoJSON.
- Le informazioni di precisione orizzontale e verticale, che indicano la differenza, espressa in metri, tra le informazioni sulla posizione stimate dai risolutori e la posizione effettiva del dispositivo.
- Il livello di affidabilità, che indica l'incertezza nella risposta della stima sulla posizione. Il valore predefinito è 0,68, che indica una probabilità del 68% che la posizione effettiva del dispositivo sia compresa nel raggio di incertezza della posizione stimata.
- La città, lo stato, il Paese e il codice postale in cui si trova il dispositivo. Queste informazioni verranno riportate solo quando viene utilizzato il risolutore di ricerca inversa IP.
- Le informazioni sul timestamp, che corrisponde alla data e ora in cui la posizione è stata risolta. Viene utilizzato il formato timestamp Unix.

Il codice riportato di seguito mostra un payload GeoJSON di esempio restituito risolvendo la posizione.

> **Note**
>
> Se AWS IoT Core Device Location segnala errori durante il tentativo di risolvere la posizione, puoi risolvere gli errori e risolvere la posizione. Per ulteriori informazioni, consulta Risoluzione dei problemi relativi alla risoluzione della posizione.

```
{
    "coordinates": [
        13.376076698303223,
        52.51823043823242
    ],
    "type": "Point",
    "properties": {
```

```
        "verticalAccuracy": 45,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 303,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T12:23:58.189Z"
    }
}
```

4. Vai alla sezione Posizione delle risorse e verifica le informazioni di geolocalizzazione riportate da Device Location. AWS IoT Core È possibile copiare il payload per utilizzarlo con altre applicazioni e applicazioni. Servizio AWS Ad esempio, è possibile utilizzare la Location (Ubicazione) per inviare i dati relativi alla posizione geografica al servizio di posizione Amazon.

Risoluzione della posizione del dispositivo (API)

Per risolvere la posizione del dispositivo utilizzando l'API AWS IoT wireless, utilizza l'operazione GetPositionEstimateAPI o il comando get-position-estimateCLI. Specificare i dati di misurazione del payload come input ed eseguire l'operazione API per risolvere la posizione del dispositivo.

> **Note**
>
> L'operazione API `GetPositionEstimate` non memorizza alcuna informazione sul dispositivo o sullo stato e non può essere utilizzata per recuperare dati storici sulla posizione. Esegue un'operazione una tantum che risolve i dati di misurazione e produce la posizione stimata. Per recuperare le informazioni sulla posizione, è necessario specificare le informazioni sul payload ogni volta che si esegue questa operazione API.

Il seguente comando mostra un esempio di come risolvere la posizione mediante questa operazione API.

> **Note**
>
> Durante l'esecuzione del comando dell'interfaccia a riga di comando `get-position-estimate`, è necessario specificare il file JSON di output come il primo input. Questo file

JSON archivierà le informazioni sulla posizione stimata ottenute come risposta dall'interfaccia a riga di comando in formato GeoJSON. Ad esempio, il comando seguente memorizza le informazioni sulla posizione nel *locationout.json* file.

```
aws iotwireless get-position-estimate locationout.json \
    --ip IpAddress=""54.240.198.35"" \
    --wi-fi-access-points \
        MacAddress="A0:EC:F9:1E:32:C1",Rss=-75 \
        MacAddress="A0:EC:F9:15:72:5E",Rss=-67
```

Questo esempio include sia i punti di accesso Wi-Fi che l'indirizzo IP come tipi di misurazione. AWS IoT Core Device Location sceglie tra il solver Wi-Fi e il solutore di ricerca inversa IP e seleziona il solutore con la maggiore precisione.

La posizione risolta viene restituita come un payload che utilizza il formato GeoJSON, ideale per codificare strutture di dati geografici. Viene quindi memorizzato nel file. *locationout.json*
Il payload contiene le coordinate di WGS84 latitudine e longitudine, le informazioni sul livello di precisione e confidenza, il tipo di dati sulla posizione e il timestamp in cui la posizione è stata risolta.

```
{
    "coordinates": [
        13.37704086303711,
        52.51865005493164
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 707,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 389,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T14:03:57.391Z"
    }
}
```

Risoluzione dei problemi relativi alla risoluzione della posizione

Quando tenti di risolvere la posizione, potresti visualizzare uno dei seguenti codici di errore. AWS IoT Core La posizione del dispositivo potrebbe generare un errore durante l'utilizzo dell'operazione `GetPositionEstimate` API oppure fare riferimento al numero di riga corrispondente all'errore nella AWS IoT console.

- Errore 400

  Questo errore indica che il formato del payload del dispositivo JSON non può essere convalidato da AWS IoT Core Device Location. Le possibile cause dell'errore sono:

  - I dati di misurazione JSON sono formattati in modo errato.
  - Il payload contiene solo le informazioni sul timestamp.
  - I parametri dei dati di misurazione, come l'indirizzo IP, non sono validi.

  Per risolvere questo errore, verificare se il JSON sia formattato correttamente e contenga dati di uno o più tipi di misurazione come input. Se l'indirizzo IP non è valido, per informazioni su come fornire un indirizzo IP valido per risolvere l'errore, consulta Risolutore di ricerca inversa IP.

- Errore 403

  Questo errore indica che non disponi delle autorizzazioni per eseguire l'operazione API o per utilizzare la AWS IoT console per recuperare la posizione del dispositivo. Per risolvere questo errore, verificare di disporre delle autorizzazioni richieste per eseguire questa azione. Questo errore può verificarsi se la Console di gestione AWS sessione o il token di AWS CLI sessione sono scaduti. Per risolvere questo errore, aggiorna il token di sessione per utilizzare o disconnettiti da Console di gestione AWS e quindi accedi utilizzando le tue credenziali. AWS CLI

- Errore 404

  Questo errore indica che nessuna informazione sulla posizione è stata trovata o risolta da AWS IoT Core Device Location. L'errore potrebbe verificarsi a causa di problemi quali dati insufficienti nell'immissione dei dati di misurazione. Esempio:

  - L'indirizzo MAC o le informazioni sul ripetitore di rete cellulare non sono sufficienti.
  - L'indirizzo IP non è disponibile per cercare e recuperare la posizione.
  - Il payload GNSS non è sufficiente.

  In questi casi, per risolvere l'errore, verificare se i dati di misurazione contengono informazioni sufficienti per risolvere la posizione del dispositivo.

- Errore 500

  Questo errore indica che si è verificata un'eccezione interna del server quando AWS IoT Core Device Location ha tentato di risolvere la posizione. Per tentare di correggere questo errore, aggiornare la sessione e riprovare a inviare i dati di misurazione da risolvere.

Risoluzione della posizione del dispositivo utilizzando gli argomenti MQTT di AWS IoT Core Device Location

È possibile utilizzare argomenti MQTT riservati per ottenere le informazioni più recenti sulla posizione dei dispositivi con la funzione AWS IoT Core Device Location.

Formato degli argomenti relativi alla posizione del dispositivo

Gli argomenti riservati per AWS IoT Core Device Location utilizzano il seguente prefisso:

`$aws/device_location/{customer_device_id}/`

Per creare un argomento completo, sostituisci prima *`customer_device_id`* con l'ID univoco che usi per identificare il tuo dispositivo. Ti consigliamo di specificare`WirelessDeviceId`, ad esempio per i dispositivi LoRa WAN e Sidewalk, e*`thingName`*, se il dispositivo è registrato come oggetto AWS IoT . Poi aggiungi l'argomento insieme allo stub dell'argomento, ad esempio `get_position_estimate` o `get_position_estimate/accepted`, come mostrato nella sezione seguente.

> **Note**
>
> *`{customer_device_id}`* può contenere solo lettere, numeri e trattini. Quando ti iscrivi agli argomenti relativi alla posizione del dispositivo, puoi utilizzare solo il segno più (+) come jolly. Ad esempio, puoi usare il jolly + per consentire a *`{customer_device_id}`* di ottenere le informazioni sulla posizione dei tuoi dispositivi. Quando ti iscrivi all'argomento `$aws/device_location/+/get_position_estimate/accepted`, verrà pubblicato un messaggio con le informazioni sulla posizione dei dispositivi che corrispondono a qualsiasi ID dispositivo se il problema è stato risolto con successo.

Di seguito sono riportati gli argomenti riservati utilizzati per interagire con AWS IoT Core Device Location.

Argomenti relativi alla posizione del dispositivo

Topic	Operazioni consentite	Description
$aws/device_location/ / get_position_estimate *customer_device_id*	Pubblica	Un dispositivo pubblica su questo argomento per ottenere i dati di misurazione grezzi scansionati da risolvere tramite Device Location. AWS IoT Core
$aws/device_location/ / get_position_estimate/ accepted *customer_ device_id*	Subscribe	AWS IoT Core Device Location pubblica le informazioni sulla posizione in questo argomento quando risolve correttamente la posizione del dispositivo.
customer_device_id $aws/device_location/ / get_position_estimate/reje cted	Subscribe	AWS IoT Core Device Location pubblica le informazioni sull'errore in questo argomento quando non riesce a risolvere la posizione del dispositivo.

Policy per gli argomenti MQTT relativi alla posizione dei dispositivi

Per ricevere messaggi dagli argomenti relativi alla localizzazione del dispositivo, il dispositivo deve utilizzare una politica che gli consenta di connettersi al gateway del AWS IoT dispositivo e sottoscrivere gli argomenti MQTT.

Di seguito viene mostrato un esempio della policy necessaria per la ricezione di messaggi per i vari argomenti.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
device_location/device-123/get_position_estimate/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
device_location/device-123/get_position_estimate/rejected"
            ]
        }
    ]
}
```

Argomenti relativi alla posizione del dispositivo e payload

Di seguito vengono illustrati gli argomenti relativi alla localizzazione del AWS IoT Core dispositivo, il formato del relativo payload dei messaggi e un esempio di policy per ogni argomento.

Argomenti

- /get_position_estimate
- /get_position_estimate/accepted
- /get_position_estimate/rejected

/get_position_estimate

Pubblicate un messaggio su questo argomento per far sì che i dati di misurazione grezzi del dispositivo vengano risolti da AWS IoT Core Device Location.

```
$aws/device_location/customer_device_id/get_position_estimate
```

AWS IoT Core Device Location risponde pubblicando su uno /get_position_estimate/accepted o/get_position_estimate/rejected.

> **Note**
>
> Il messaggio pubblicato su questo argomento deve essere un payload JSON valido. Se il messaggio di input non è in formato JSON valido, non riceverai alcuna risposta. Per ulteriori informazioni, consulta Payload del messaggio.

Payload del messaggio

Il formato del payload del messaggio segue una struttura simile a quella del corpo della richiesta di operazione dell'API AWS IoT Wireless,. `GetPositionEstimate` Contiene:

- Una stringa opzionale `Timestamp`, che corrisponde alla data e all'ora in cui la posizione è stata risolta. La stringa `Timestamp` può avere una lunghezza minima di 1 e una lunghezza massima di 10.
- Una stringa opzionale `MessageId`, che può essere utilizzata per mappare la richiesta alla risposta. Se specifichi questa stringa, il messaggio pubblicato negli argomenti `get_position_estimate/accepted` o`get_position_estimate/rejected` conterrà questo `MessageId`.
  La stringa `MessageID` può avere una lunghezza minima di 1 e una lunghezza massima di 256.
- I dati di misurazione del dispositivo che contengono uno o più dei seguenti tipi di misurazione:
  - `WiFiAccessPoint`
  - `CellTowers`
  - `IpAddress`
  - `Gnss`

Quello che segue è un playload del messaggio di esempio.

```
{
    "Timestamp": "1664313161",
    "MessageId": "ABCD1",
    "WiFiAccessPoints":  [
        {
            "MacAddress": "A0:EC:F9:1E:32:C1",
            "Rss": -66
        }
    ],
    "Ip":{
        "IpAddress": "54.192.168.0"
    },
    "Gnss":{
      "Payload":"8295A614A2029517F4F77C0A7823B161A6FC57E25183D96535E3689783F6CA48",
      "CaptureTime":1354393948
    }
}
```

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate"
            ]
        }
    ]
}
```

/get_position_estimate/accepted

AWS IoT Core Device Location pubblica una risposta a questo argomento quando restituisce le informazioni sulla posizione risolte per il dispositivo. Le informazioni sulla posizione vengono restituite in formato GeoJSON.

```
$aws/device_location/customer_device_id/get_position_estimate/accepted
```

Di seguito viene illustrato il payload del messaggio e un esempio di policy.

Payload del messaggio

Di seguito è riportato un esempio di payload del messaggi in formato GeoJSON. Se hai specificato a `MessageId` nei dati di misurazione grezzi e AWS IoT Core Device Location ha risolto correttamente le informazioni sulla posizione, il payload del messaggio restituisce le stesse `MessageId` informazioni.

```
{
    "coordinates": [
        13.37704086303711,
        52.51865005493164
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 707,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 389,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T14:03:57.391Z",
        "messageId": "ABCD1"
    }
}
```

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
device_location/device-123/get_position_estimate/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate/accepted"
            ]
        }
    ]
}
```

/get_position_estimate/rejected

AWS IoT Core Device Location pubblica una risposta di errore a questo argomento quando non riesce a risolvere la posizione del dispositivo.

```
$aws/device_location/customer_device_id/get_position_estimate/rejected
```

Di seguito viene illustrato il payload del messaggio e una policy di esempio. Per informazioni sugli errori, vedi Risoluzione dei problemi relativi alla risoluzione della posizione.

Payload del messaggio

Di seguito è riportato un esempio del payload del messaggio che fornisce il codice e il messaggio di errore, che indicano perché AWS IoT Core Device Location non è riuscito a risolvere le informazioni

sulla posizione. Se hai specificato a `MessageId` quando hai fornito i dati di misurazione non elaborati e AWS IoT Core Device Location non è riuscito a risolvere le informazioni sulla posizione, le stesse `MessageId` informazioni verranno restituite nel payload del messaggio.

```
{
    "errorCode": 500,
    "errorMessage":"Internal server error",
    "messageId": "ABCD1"
}
```

Policy di esempio

Di seguito è illustrato un esempio della policy necessaria:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/device_location/
device-123/get_position_estimate/rejected"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/device-123/
get_position_estimate/rejected"
      ]
    }
  ]
}
```

Risolutori di posizione e payload del dispositivo

I location solver sono algoritmi che possono essere utilizzati per determinare la posizione dei dispositivi IoT. AWS IoT Core Device Location supporta i seguenti risolutori di posizione. Sono disponibili esempi del formato di payload JSON per questi tipi di misurazione, dei dispositivi supportati dal risolutore e della modalità di risoluzione della posizione.

Per risolvere la posizione del dispositivo, specificare uno o più di questi tipi di dati di misurazione. Una singola posizione risolta verrà restituita per tutti i dati di misurazione combinati.

Argomenti

- Risolutore basato su Wi-Fi
- Risolutore basato su rete cellulare
- Risolutore di ricerca inversa IP
- Risolutore GNSS
- Solver BLE per dispositivi compatibili con Amazon Sidewalk

Risolutore basato su Wi-Fi

Utilizzare il risolutore basato su Wi-Fi per risolvere la posizione utilizzando le informazioni di scansione di punti di accesso Wi-Fi. Il solver supporta la tecnologia WLAN e può essere utilizzato per calcolare la posizione del dispositivo per dispositivi IoT generici che si connettono tramite TCP/IP, dispositivi LoRa wireless WAN e dispositivi abilitati per Amazon Sidewalk.

I dispositivi LoRa WAN devono disporre del chipset LoRa Edge, in grado di decodificare le informazioni di scansione Wi-Fi in ingresso. LoRa Edge è una piattaforma a bassissimo consumo che integra un LoRa ricetrasmettitore a lungo raggio, uno scanner GNSS a più costellazioni e uno scanner MAC Wi-Fi passivo destinato alle applicazioni di geolocalizzazione. Quando viene ricevuto un messaggio uplink dal dispositivo, i dati di scansione Wi-Fi vengono inviati a Device Location e la posizione viene stimata in base ai AWS IoT Core risultati della scansione Wi-Fi. Le informazioni decodificate vengono quindi passate al risolutore basato su Wi-Fi per recuperare le informazioni sulla posizione.

Per ulteriori informazioni sulla risoluzione dei dati di localizzazione per i dispositivi compatibili con Amazon Sidewalk, consulta la guida per AWS IoT Core per Amazon Sidewalk sviluppatori.

Esempio di payload del risolutore basato su Wi-Fi

Il codice seguente mostra un esempio di payload JSON del dispositivo contenente i dati di misurazione. Quando AWS IoT Core Device Location riceve questi dati come input, invia una richiesta HTTP al solver provider per risolvere le informazioni sulla posizione. Per recuperare le informazioni, specificare i valori per Indirizzo MAC e l'RSS (intensità del segnale ricevuto). A tale scopo, fornite il payload JSON utilizzando questo formato o utilizzate il parametro WiFiAccessPointsobject dell'operazione API. GetPositionEstimate

```
{
    "Timestamp": 1664313161,    // optional
    "WiFiAccessPoints": [
        {
            "MacAddress": "A0:EC:F9:1E:32:C1",  // required
            "Rss": -75                          // required
        }
    ]
}
```

Risolutore basato su rete cellulare

È possibile utilizzare il risolutore basato su rete cellulare per risolvere la posizione utilizzando i dati di misurazione ottenuti dalle torri radio di reti cellulare. Il risolutore supporta le tecnologie riportate di seguito. Anche se si includono i dati di misurazione di una o tutte queste tecnologie, viene ottenuta una singola informazione sulla posizione risolta.

- GSM
- CDMA
- WCDMA
- TD-SCDMA
- LTE

Esempi di payload di risolutore basato su rete cellulare

Il codice seguente mostra esempi di payload JSON del dispositivo contenente i dati di misurazione della rete cellulare. Quando AWS IoT Core Device Location riceve questi dati come input, invia una richiesta HTTP al solver provider per risolvere le informazioni sulla posizione. Per recuperare le informazioni, è possibile fornire il payload JSON utilizzando questo formato nella console o

specificare i valori per il CellTowersparametro dell'operazione API. GetPositionEstimate È possibile fornire i dati di misurazione specificando i valori dei parametri utilizzando una o tutte queste tecnologie di rete cellulare.

LTE (Long-term evolution)

Quando si utilizzano questi dati di misurazione, è necessario specificare informazioni quali la rete e il codice paese della rete mobile, nonché parametri aggiuntivi opzionali, incluse le informazioni sull'ID locale. Il codice seguente mostra un esempio del formato di payload. Per ulteriori informazioni su questi parametri, consulta Oggetto LTE.

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Lte": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "EutranCid": int,             // required. Make sure that you use int for
 EutranCid.
          "Tac": int,                   // optional
          "LteLocalId": {               // optional
              "Pci": int,               // required
              "Earfcn": int,            // required
          },
          "LteTimingAdvance": int,      // optional
          "Rsrp": int,                  // optional
          "Rsrq": float,                // optional
          "NrCapable": boolean,         // optional
          "LteNmr": [                   // optional
                {
                    "Pci": int,         // required
                    "Earfcn": int,      // required
                    "EutranCid": int,   // required
                    "Rsrp": int,        // optional
                    "Rsrq": float       // optional
                }
            ]
         }
      ]
   }
}
```

GSM (Global System for Mobile Communications)

Quando si utilizzano questi dati di misurazione, è necessario specificare informazioni quali la rete e il codice paese della rete mobile, le informazioni sulla stazione base, nonché parametri aggiuntivi opzionali. Il codice seguente mostra un esempio del formato di payload. Per ulteriori informazioni su questi parametri, consulta Oggetto GSM.

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Gsm": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "Lac": int,                   // required
          "GeranCid": int,              // required
          "GsmLocalId": {               // optional
              "Bsic": int,              // required
              "Bcch": int,              // required
          },
          "GsmTimingAdvance": int,      // optional
          "RxLevel": int,               // optional
          "GsmNmr": [                   // optional
            {
                "Bsic": int,            // required
                "Bcch": int,            // required
                "RxLevel": int,         // optional
                "GlobalIdentity": {
                    "Lac": int,         // required
                    "GeranCid": int     // required
                }
             }
          ]
       }
    ]
}
```

CDMA (Code-division multiple access)

Quando si utilizzano questi dati di misurazione, è necessario specificare informazioni quali la potenza del segnale e le informazioni di identificazione, le informazioni sulla stazione base, nonché parametri

aggiuntivi opzionali. Il codice seguente mostra un esempio del formato di payload. Per ulteriori informazioni su questi parametri, consulta Oggetto CDMA.

```
{
    "Timestamp": 1664313161,                // optional
    "CellTowers": {
        "Cdma": [
        {
            "SystemId": int,                // required
            "NetworkId": int,               // required
            "BaseStationId": int,           // required
            "RegistrationZone": int,        // optional
            "CdmaLocalId": {                // optional
              "PnOffset": int,              // required
              "CdmaChannel": int,           // required
            },
            "PilotPower": int,              // optional
            "BaseLat": float,               // optional
            "BaseLng": float,               // optional
            "CdmaNmr": [                    // optional
                {
                    "PnOffset": int,        // required
                    "CdmaChannel": int,     // required
                    "PilotPower": int,      // optional
                    "BaseStationId": int    // optional
                }
             ]
          }
       ]
    }
}
```

WCDMA (Wideband code-division multiple access)

Quando si utilizzano questi dati di misurazione, è necessario specificare informazioni quali la rete e il codice paese, la potenza del segnale e le informazioni di identificazione, le informazioni sulla stazione base, nonché parametri aggiuntivi opzionali. Il codice seguente mostra un esempio del formato di payload. Per ulteriori informazioni su questi parametri, consulta Oggetto CDMA.

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
```

```
        "Wcdma": [
        {
          "Mcc": int,                    // required
          "Mnc": int,                    // required
          "UtranCid": int,               // required
          "Lac": int,                    // optional
          "WcdmaLocalId": {              // optional
              "Uarfcndl": int,           // required
              "Psc": int,                // required
          },
          "Rscp": int,                   // optional
          "Pathloss": int,               // optional
          "WcdmaNmr": [                  // optional
                {
                  "Uarfcndl": int,       // required
                  "Psc": int,            // required
                  "UtranCid": int,       // required
                  "Rscp": int,           // optional
                  "Pathloss": int,       // optional
                }
             ]
          }
       ]
    }
}
```

TD-SCDMA (Time division synchronous code-division multiple access)

Quando si utilizzano questi dati di misurazione, è necessario specificare informazioni quali la rete e il codice paese, la potenza del segnale e le informazioni di identificazione, le informazioni sulla stazione base, nonché parametri aggiuntivi opzionali. Il codice seguente mostra un esempio del formato di payload. Per ulteriori informazioni su questi parametri, consulta Oggetto CDMA.

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Tdscdma": [
        {
          "Mcc": int,                    // required
          "Mnc": int,                    // required
          "UtranCid": int,               // required
          "Lac": int,                    // optional
          "TdscdmaLocalId": {            // optional
```

```
            "Uarfcn": int,            // required
            "CellParams": int,        // required
         },
         "TdscdmaTimingAdvance": int,  // optional
         "Rscp": int,                  // optional
         "Pathloss": int,              // optional
         "TdscdmaNmr": [               // optional
               {
                 "Uarfcn": int,        // required
                 "CellParams": int,    // required
                 "UtranCid": int,      // optional
                 "Rscp": int,          // optional
                 "Pathloss": int,      // optional
               }
            ]
         }
      ]
   }
}
```

Risolutore di ricerca inversa IP

È possibile utilizzare il risolutore di ricerca inversa IP per risolvere la posizione utilizzando l'indirizzo IP come input. Il solver può ottenere le informazioni sulla posizione dai dispositivi di cui è stato fornito. AWS IoT Specificate le informazioni sull'indirizzo IP utilizzando un formato che sia il modello IPv6 standard IPv4 o il modello compresso IPv6 esadecimale. Si ottiene quindi la stima della posizione risolta, incluse informazioni aggiuntive quali la città e il paese in cui si trova il dispositivo.

> **Note**
>
> Utilizzando la ricerca inversa IP, l'utente accetta di non utilizzarla allo scopo di identificare o localizzare un indirizzo residenziale o stradale specifico.

Esempio di payload del risolutore di ricerca inversa IP

Il codice seguente mostra un esempio di payload JSON del dispositivo contenente i dati di misurazione. Quando AWS IoT Core Device Location riceve le informazioni sull'indirizzo IP contenute nei dati di misurazione, cerca tali informazioni nel database del solver provider, che viene quindi utilizzato per risolvere le informazioni sulla posizione. Per recuperare le informazioni, fornite il

payload JSON utilizzando questo formato o specificate i valori per il parametro Ip dell'operazione API. GetPositionEstimate

> **Note**
>
> Quando si utilizza questo risolutore, oltre alle coordinate vengono riportati anche la città, lo stato, il Paese e il codice postale in cui si trova il dispositivo. Per vedere un esempio, consulta Risoluzione della posizione del dispositivo (console).

```
{
    "Timestamp": 1664313161,
    "Ip":{
        "IpAddress":"54.240.198.35"
      }
}
```

Risolutore GNSS

Utilizzare il risolutore GNSS (Global Navigation Satellite System) per recuperare la posizione del dispositivo utilizzando le informazioni contenute nei messaggi del risultato di scansione GNSS o nei messaggi NAV. Facoltativamente, è possibile fornire informazioni di assistenza GNSS aggiuntive, che riducono il numero di variabili che il risolutore deve utilizzare per cercare i segnali. Fornendo queste informazioni di assistenza, che includono la posizione, l'altitudine, il tempo di acquisizione e le informazioni sulla precisione, il risolutore può identificare facilmente i satelliti in vista e calcolare la posizione del dispositivo.

Questo solver può essere utilizzato con dispositivi LoRa WAN e dispositivi abilitati per Amazon Sidewalk, nonché con altri dispositivi che sono stati dotati di dispositivi AWS IoT di cui è stato effettuato il provisioning. AWS IoT Per dispositivi IoT generici, se i dispositivi supportano la stima della posizione mediante GNSS, quando le informazioni di scansione GNSS vengono ricevute dal dispositivo, i ricetrasmettitori risolvono le informazioni sulla posizione. Per i dispositivi LoRa WAN, i dispositivi devono disporre del chipset Edge. LoRa Quando viene ricevuto un messaggio di uplink dal dispositivo, i dati di scansione GNSS vengono inviati alla LoRa rete WAN o AWS IoT ad AWS IoT Amazon Sidewalk e la posizione viene stimata in base ai risultati della scansione dei ricetrasmettitori.

Esempio di payload del risolutore GNSS

Il codice seguente mostra un esempio di payload JSON del dispositivo contenente i dati di misurazione. Quando AWS IoT Core Device Location riceve le informazioni di scansione GNSS contenenti il payload nei dati di misurazione, utilizza i ricetrasmettitori e tutte le informazioni di assistenza aggiuntive incluse per cercare segnali e risolvere le informazioni sulla posizione. Per recuperare le informazioni, fornite il payload JSON utilizzando questo formato o specificate i valori per il parametro Gnss dell'operazione API. GetPositionEstimate

> **Note**
>
> Prima che AWS IoT Core Device Location possa risolvere la posizione del dispositivo, è necessario rimuovere il byte di destinazione dal payload.

```
{
    "Timestamp": 1664313161,                // optional
    "Gnss": {
        "AssistAltitude": number,           // optional
        "AssistPosition": [ number ],       // optional
        "CaptureTime": number,              // optional
        "CaptureTimeAccuracy": number,      // optional
        "Payload": "string",                // required
        "Use2DSolver": boolean              // optional
   }
}
```

Solver BLE per dispositivi compatibili con Amazon Sidewalk

> **Note**
>
> Per la posizione basata su Bluetooth Low Energy, AWS IoT restituisce le coordinate della posizione in base alla posizione approssimativa dei Sidewalk Gateway vicini collegati ad Amazon Sidewalk e con la funzione Community Finding abilitata. I dati di localizzazione del gateway sono AWS Contenuti e vengono forniti all'utente esclusivamente allo scopo di aiutarlo a localizzare i dispositivi collegati ad Amazon Sidewalk e l'utente deve utilizzare i dati solo a tale scopo. Devi utilizzare e accedere ai dati sulla posizione solo tramite l'interfaccia e la funzionalità che generalmente ti mettiamo a disposizione e non devi tentare

di reidentificare, decodificare o ri-mappare i dati sulla posizione del Gateway che ti abbiamo fornito.

I dispositivi terminali Amazon Sidewalk che si connettono AWS IoT Core per Amazon Sidewalk possono risolvere i dati sulla posizione utilizzando messaggi uplink BLE, Wi-Fi o GNSS pubblicati dal dispositivo finale Amazon Sidewalk. Per ulteriori informazioni, consulta la guida per gli sviluppatori.AWS IoT Core per Amazon Sidewalk

Messaggi di eventi

Questa sezione contiene informazioni sui messaggi pubblicati in occasione AWS IoT dell'aggiornamento o della modifica di elementi o lavori. Per informazioni sul AWS IoT Events servizio che consente di creare rilevatori per monitorare i dispositivi per rilevare eventuali guasti o modifiche di funzionamento e per attivare azioni quando si verificano, vedere. AWS IoT Events

Come vengono generati i messaggi di evento

AWS IoT pubblica messaggi relativi agli eventi quando si verificano determinati eventi. Ad esempio, vengono generati eventi dal registro quando vengono aggiunti, aggiornati o eliminati oggetti. Ogni evento comporta l'invio di un singolo messaggio di evento. I messaggi di evento vengono pubblicati tramite MQTT con un payload JSON. Il contenuto del payload dipende dal tipo di evento.

> **Note**
>
> I messaggi di evento vengono sicuramente pubblicati una volta. È anche possibile che vengano pubblicati più di una volta. L'ordinamento dei messaggi di evento non è garantito.

Policy per la ricezione di messaggi di evento

Per ricevere messaggi relativi agli eventi, il dispositivo deve utilizzare una politica appropriata che gli consenta di connettersi al gateway del AWS IoT dispositivo e sottoscrivere gli argomenti relativi agli eventi MQTT. Devi anche sottoscrivere i filtri di argomenti appropriati.

Di seguito viene mostrato un esempio della policy necessaria per la ricezione di eventi del ciclo di vita:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe",
                "iot:Receive"
```

```
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:/$aws/events/*"
            ]
        }
    ]
}
```

Abilita eventi per AWS IoT

Prima che gli abbonati agli argomenti riservati possano ricevere messaggi, è necessario abilitare i messaggi di evento da Console di gestione AWS o utilizzando l'API o la CLI. Per informazioni sui messaggi di evento gestiti dalle diverse opzioni, consulta la Tabella delle impostazioni di configurazione degli AWS IoT eventi.

- Per abilitare i messaggi relativi agli eventi, vai alla scheda Impostazioni della AWS IoT console e quindi, nella sezione Messaggi basati sugli eventi, scegli Gestisci eventi. Puoi specificare gli eventi da gestire.
- Per controllare quali tipi di eventi vengono pubblicati utilizzando l'API o la CLI, chiama l'UpdateEventConfigurationsAPI o utilizza il comando CLIupdate-event-configurations. Esempio:

```
aws iot update-event-configurations --event-configurations "{\"THING\":{\"Enabled\":
 true}}"
```

> **Note**
>
> Tutte le virgolette doppie (") sono precedute dal carattere di escape barra rovesciata (\).

È possibile ottenere la configurazione corrente dell'evento chiamando l'DescribeEventConfigurationsAPI o utilizzando il comando describe-event-configurations CLI. Ad esempio:.

```
aws iot describe-event-configurations
```

Tabella delle impostazioni di configurazione degli eventi AWS IoT

Categoria dell'evento (AWS IoT Console: Impostazioni: messaggi basati su eventi)	Chiave-valore **`eventConfigurations`** (AWS CLI/API)	Argomento del messaggio dell'evento
(Può essere configurato solo utilizzando la AWS CLI/API)	`CA_CERTIFICATE`	`$aws/events/certificates/registered/` *`caCertificateId`*
(Può essere configurato solo utilizzando la AWS CLI/API)	`CERTIFICATE`	`$aws/events/presence/connected/` *`clientId`*
(Può essere configurato solo utilizzando la AWS CLI/API)	`CERTIFICATE`	`$aws/events/presence/disconnected/` *`clientId`*
(Può essere configurato solo utilizzando la AWS CLI/API)	`CERTIFICATE`	`$aws/events/subscriptions/subscribed/` *`clientId`*
(Può essere configurato solo utilizzando la AWS CLI/API)	`CERTIFICATE`	`$aws/events/subscriptions/unsubscribed/` *`clientId`*
Processo completato, annullato	`JOB`	`$aws/events/job/`*`jobID`*`/canceled`
Processo completato, annullato	`JOB`	`$aws/events/job/`*`jobID`*`/cancellation_in_progress`
Processo completato, annullato	`JOB`	`$aws/events/job/`*`jobID`*`/completed`
Processo completato, annullato	`JOB`	`$aws/events/job/`*`jobID`*`/deleted`

Categoria dell'evento (AWS IoT Console: Impostazioni: messaggi basati su eventi)	Chiave-valore **`eventConfigurations`** (AWS CLI/API)	Argomento del messaggio dell'evento
Processo completato, annullato	`JOB`	`$aws/events/job/`*`jobID`*`/deletion_in_progress`
Esecuzione del processo: riuscita, non riuscita, rifiutata, annullata, rimossa	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/canceled`
Esecuzione del processo: riuscita, non riuscita, rifiutata, annullata, rimossa	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/deleted`
Esecuzione del processo: riuscita, non riuscita, rifiutata, annullata, rimossa	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/failed`
Esecuzione del processo: riuscita, non riuscita, rifiutata, annullata, rimossa	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/rejected`
Esecuzione del processo: riuscita, non riuscita, rifiutata, annullata, rimossa	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/removed`
Esecuzione del processo: riuscita, non riuscita, rifiutata, annullata, rimossa	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/succeeded`
Esecuzione del processo: riuscita, non riuscita, rifiutata, annullata, rimossa	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/timed_out`

Categoria dell'evento (AWS IoT Console: Impostazioni: messaggi basati su eventi)	Chiave-valore **`eventConfigurations`** (AWS CLI/API)	Argomento del messaggio dell'evento
Oggetto: creato, aggiornato, eliminato	`THING`	`$aws/events/thing/` *`thingName`* `/created`
Oggetto: creato, aggiornato, eliminato	`THING`	`$aws/events/thing/` *`thingName`* `/updated`
Oggetto: creato, aggiornato, eliminato	`THING`	`$aws/events/thing/` *`thingName`* `/deleted`
Gruppo di cose: aggiunto, rimosso	`THING_GROUP`	`$aws/events/thingGroup/` *`thingGroupName`* `/created`
Gruppo di cose: aggiunto, rimosso	`THING_GROUP`	`$aws/events/thingGroup/` *`thingGroupName`* `/updated`
Gruppo di cose: aggiunto, rimosso	`THING_GROUP`	`$aws/events/thingGroup/` *`thingGroupName`* `/deleted`
Gerarchia del gruppo di cose: aggiunta, rimossa	`THING_GROUP_HIERARCHY`	`$aws/events/thingGroupHierarchy/thingGroup/` *`parentThingGroupName`* `/childThingGroup/` *`childThingGroupName`* `/added`

Categoria dell'evento (AWS IoT Console: Impostazioni: messaggi basati su eventi)	Chiave-valore **eventConfigurations** (AWS CLI/API)	Argomento del messaggio dell'evento
Gerarchia del gruppo di cose: aggiunta, rimossa	THING_GROUP_HIERARCHY	$aws/events/thingGroupHierarchy/thingGroup/ *parentThingGroupName* /childThingGroup/ *childThingGroupName* /removed
Appartenenza al gruppo di cose: aggiunta, rimossa	THING_GROUP_MEMBERSHIP	$aws/events/thingGroupMembership/thingGroup/ *thingGroupName* /thing/*thingName* /added
Appartenenza al gruppo di cose: aggiunta, rimossa	THING_GROUP_MEMBERSHIP	$aws/events/thingGroupMembership/thingGroup/ *thingGroupName* /thing/*thingName* /removed
Tipo di oggetto: creato, aggiornato, eliminato	THING_TYPE	$aws/events/thingType/ *thingTypeName* /created
Tipo di oggetto: creato, aggiornato, eliminato	THING_TYPE	$aws/events/thingType/ *thingTypeName* /updated
Tipo di oggetto: creato, aggiornato, eliminato	THING_TYPE	$aws/events/thingType/ *thingTypeName* /deleted

Categoria dell'evento (AWS IoT Console: Impostazioni: messaggi basati su eventi)	Chiave-valore **`eventConfigurations`** (AWS CLI/API)	Argomento del messaggio dell'evento

Associazione del tipo di oggetto: aggiunta, rimossa	`THING_TYPE_ASSOCIATION`	`$aws/events/thingTypeAssociation/thing/` *`thingName`* `/thingType/` *`thingTypeName`* `/added`
		
		`$aws/events/thingTypeAssociation/thing/` *`thingName`* `/thingType/` *`thingTypeName`* `/removed`

Eventi del registro

Il registro pubblica messaggi di evento quando vengono creati, aggiornati o eliminati oggetti, tipi di oggetto e gruppi di oggetti. Tuttavia, questi eventi non sono disponibili per impostazione predefinita. Per ulteriori informazioni su come configurare tali eventi, consulta Abilita eventi per AWS IoT.

Il registro può fornire i seguenti tipi di evento:

- Eventi oggetto
- Eventi di tipo di oggetto
- Eventi del gruppo di oggetti

Eventi oggetto

Cosa Created/Updated/Deleted

Il registro pubblica i messaggi di evento seguenti quando vengono creati, aggiornati o eliminati oggetti:

- `$aws/events/thing/`*`thingName`*`/created`

- `$aws/events/thing/thingName/updated`
- `$aws/events/thing/thingName/deleted`

I messaggi contengono il payload di esempio seguente:

```
{
    "eventType" : "THING_EVENT",
    "eventId" : "f5ae9b94-8b8e-4d8e-8c8f-b3266dd89853",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName" : "MyThing",
    "versionNumber" : 1,
    "thingTypeName" : null,
    "attributes": {
                "attribute3": "value3",
                "attribute1": "value1",
                "attribute2": "value2"
    }
}
```

I payload contengono gli attributi seguenti:

eventType

Imposta su "THING_EVENT".

eventId

ID evento univoco (stringa).

timestamp

Timestamp UNIX del momento in cui si è verificato l'evento.

operation

Operazione che ha attivato l'evento. I valori validi sono:

- created
- AGGIORNATO
- ELIMINATO

accountId

La tua Account AWS carta d'identità.

thingId

ID dell'oggetto creato, aggiornato o eliminato.

thingName

Nome dell'oggetto creato, aggiornato o eliminato.

versionNumber

Versione dell'oggetto creato, aggiornato o eliminato. Questo valore è impostato su 1 quando viene creato un oggetto. Il valore viene incrementato di 1 ogni volta che l'oggetto viene aggiornato.

thingTypeName

Tipo di oggetto associato all'oggetto, se ne esiste uno. In caso contrario, `null`.

attributes

Raccolta di coppie nome/valore associate all'oggetto.

Eventi di tipo di oggetto

Eventi correlati al tipo di oggetto:

- Tipo di cosa Created/Updated/Deprecated/Undeprecated/Deleted
- Tipo di oggetto associato o dissociato rispetto a un oggetto

Tipo di cosa Created/Updated/Deprecated/Undeprecated/Deleted

Il registro pubblica i seguenti messaggi di evento quando i tipi di oggetto vengono creati, aggiornati, obsoleti, non obsoleti o eliminati:

- `$aws/events/thingType/`*`thingTypeName`*`/created`
- `$aws/events/thingType/`*`thingTypeName`*`/updated`
- `$aws/events/thingType/`*`thingTypeName`*`/deleted`

Il messaggio contiene il seguente payload di esempio:

```
{
    "eventType" : "THING_TYPE_EVENT",
    "eventId" : "8827376c-4b05-49a3-9b3b-733729df7ed5",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingTypeId" : "c530ae83-32aa-4592-94d3-da29879d1aac",
    "thingTypeName" : "MyThingType",
    "isDeprecated" : false|true,
    "deprecationDate" : null,
    "searchableAttributes" : [ "attribute1", "attribute2", "attribute3" ],
    "propagatingAttributes": [
        {
     "userPropertyKey": "key",
     "thingAttribute": "model"
        },
        {
     "userPropertyKey": "key",
     "connectionAttribute": "iot:ClientId"
        }
    ],
    "description" : "My thing type"
}
```

I payload contengono gli attributi seguenti:

eventType

Imposta su "THING_TYPE_EVENT".

eventId

ID evento univoco (stringa).

timestamp

Timestamp UNIX del momento in cui si è verificato l'evento.

operation

Operazione che ha attivato l'evento. I valori validi sono:

- created
- AGGIORNATO
- ELIMINATO

accountId

Account AWS Il tuo ID.

thingTypeId

L'ID del tipo di oggetto che viene creato, aggiornato, obsoleto o eliminato.

thingTypeName

Il nome del tipo di oggetto che viene creato, aggiornato, obsoleto o eliminato.

isDeprecated

`true` se il tipo di oggetto è obsoleto. In caso contrario, `false`.

deprecationDate

Timestamp UNIX del momento in cui il tipo di oggetto è stato dichiarato obsoleto.

searchableAttributes

Raccolta di coppie nome/valore associate al tipo di oggetto che può essere usato per la ricerca.

Attributi di propagazione

Un elenco di attributi di propagazione. Un attributo di propagazione può contenere un attributo thing, un attributo connection e una chiave di proprietà utente. Per ulteriori informazioni, vedere Aggiungere attributi di propagazione per l'arricchimento dei messaggi.

description

Descrizione del tipo di oggetto.

Tipo di oggetto associato o dissociato rispetto a un oggetto

Il registro pubblica i messaggi di evento seguenti quando un tipo di oggetto viene associato o dissociato rispetto a un oggetto.

- `$aws/events/thingTypeAssociation/thing/`*`thingName`*`/thingType/`*`typeName`*`/added`
- `$aws/events/thingTypeAssociation/thing/`*`thingName`*`/thingType/`*`typeName`*`/removed`

Di seguito è riportato un esempio di `added` payload. I messaggi di payload per `removed` sono simili.

```
{
    "eventId" : "87f8e095-531c-47b3-aab5-5171364d138d",
    "eventType" : "THING_TYPE_ASSOCIATION_EVENT",
    "operation" : "ADDED",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName": "myThing",
    "thingTypeName" : "MyThingType",
    "timestamp" : 1234567890123,
}
```

I payload contengono gli attributi seguenti:

eventId

ID evento univoco (stringa).

eventType

Imposta su "THING_TYPE_ASSOCIATION_EVENT".

operation

Operazione che ha attivato l'evento. I valori validi sono:

- AGGIUNTO
- REMOVED

thingId

L'ID dell'oggetto la cui associazione a un determinato tipo è stata modificata.

thingName

Il nome dell'oggetto la cui associazione a un determinato tipo è stata modificata.

thingTypeName

Il tipo di oggetto associato, o non più associato, con l'oggetto.

timestamp

Timestamp UNIX del momento in cui si è verificato l'evento.

Eventi del gruppo di oggetti

Eventi correlati al gruppo di oggetti:

- Thing Group Created/Updated/Deleted
- Oggetto aggiunto o rimosso in un gruppo di oggetti
- Gruppo di oggetti aggiunto o rimosso in un gruppo di oggetti

Thing Group Created/Updated/Deleted

Il registro pubblica i messaggi di evento seguenti quando viene creato, aggiornato o eliminato un gruppo di oggetti.

- `$aws/events/thingGroup/`*`groupName`*`/created`
- `$aws/events/thingGroup/`*`groupName`*`/updated`
- `$aws/events/thingGroup/`*`groupName`*`/deleted`

Di seguito è riportato un esempio di `updated` payload. I messaggi di payload per `created` e `deleted` sono simili.

```
{
  "eventType": "THING_GROUP_EVENT",
  "eventId": "8b9ea8626aeaa1e42100f3f32b975899",
  "timestamp": 1603995417409,
  "operation": "UPDATED",
  "accountId": "571EXAMPLE833",
  "thingGroupId": "8757eec8-bb37-4cca-a6fa-403b003d139f",
  "thingGroupName": "Tg_level5",
  "versionNumber": 3,
  "parentGroupName": "Tg_level4",
  "parentGroupId": "5fce366a-7875-4c0e-870b-79d8d1dce119",
  "description": "New description for Tg_level5",
  "rootToParentThingGroups": [
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/TgTopLevel",
      "groupId": "36aa0482-f80d-4e13-9bff-1c0a75c055f6"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level1",
      "groupId": "bc1643e1-5a85-4eac-b45a-92509cbe2a77"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level2",
      "groupId": "0476f3d2-9beb-48bb-ae2c-ea8bd6458158"
```

```
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level3",
      "groupId": "1d9d4ffe-a6b0-48d6-9de6-2e54d1eae78f"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level4",
      "groupId": "5fce366a-7875-4c0e-870b-79d8d1dce119"
    }
  ],
  "attributes": {
    "attribute1": "value1",
    "attribute3": "value3",
    "attribute2": "value2"
  },
  "dynamicGroupMappingId": null
}
```

I payload contengono gli attributi seguenti:

eventType

Imposta su "THING_GROUP_EVENT".

eventId

ID evento univoco (stringa).

timestamp

Timestamp UNIX del momento in cui si è verificato l'evento.

operation

Operazione che ha attivato l'evento. I valori validi sono:

- created
- AGGIORNATO
- ELIMINATO

accountId

Il tuo Account AWS ID.

thingGroupId

ID del gruppo di oggetti creato, aggiornato o eliminato.

thingGroupName

Nome del gruppo di oggetti creato, aggiornato o eliminato.

versionNumber

Versione del gruppo di oggetti. Questo valore è impostato su 1 quando viene creato un gruppo di oggetti. Il valore viene incrementato di 1 ogni volta che il gruppo di oggetti viene aggiornato.

parentGroupName

Nome del gruppo di oggetti padre, se esistente.

parentGroupId

ID del gruppo di oggetti padre, se esistente.

description

Descrizione del gruppo di oggetti.

rootToParentThingGroups

Matrice di informazioni sul gruppo di oggetti padre. È presente un elemento per ogni gruppo di oggetti padre, iniziando dal gruppo di oggetti root e continuando fino a raggiungere il gruppo di oggetti padre. Ogni voce contiene `groupArn` e `groupId` del gruppo di oggetti.

attributes

Raccolta di coppie nome/valore associate al gruppo di oggetti.

Oggetto aggiunto o rimosso in un gruppo di oggetti

Il registro pubblica i messaggi di evento seguenti quando un oggetto viene aggiunto o rimosso in un gruppo di oggetti.

- `$aws/events/thingGroupMembership/thingGroup/`*`thingGroupName`*`/thing/`*`thingName`*`/added`
- `$aws/events/thingGroupMembership/thingGroup/`*`thingGroupName`*`/thing/`*`thingName`*`/removed`

I messaggi contengono il payload di esempio seguente:

```
{
```

```
    "eventType" : "THING_GROUP_MEMBERSHIP_EVENT",
    "eventId" : "d684bd5f-6f6e-48e1-950c-766ac7f02fd1",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|REMOVED",
    "accountId" : "123456789012",
    "groupArn" : "arn:aws:iot:ap-northeast-2:123456789012:thinggroup/
MyChildThingGroup",
    "groupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "thingArn" : "arn:aws:iot:ap-northeast-2:123456789012:thing/MyThing",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "membershipId" : "8505ebf8-4d32-4286-80e9-c23a4a16bbd8"
}
```

I payload contengono gli attributi seguenti:

eventType

Imposta su "THING_GROUP_MEMBERSHIP_EVENT".

eventId

ID evento.

timestamp

Timestamp UNIX del momento in cui si è verificato l'evento.

operation

`ADDED` quando un oggetto viene aggiunto a un gruppo di oggetti. `REMOVED` quando un oggetto viene rimosso da un gruppo di oggetti.

accountId

Il tuo Account AWS documento d'identità.

groupArn

ARN del gruppo di oggetti.

groupId

ID del gruppo.

thingArn

ARN dell'oggetto aggiunto o rimosso nel gruppo di oggetti.

thingId

ID dell'oggetto aggiunto o rimosso nel gruppo di oggetti.

membershipId

ID che rappresenta la relazione tra l'oggetto e il gruppo di oggetti. Questo valore viene generato quando aggiungi un oggetto a un gruppo di oggetti.

Gruppo di oggetti aggiunto o rimosso in un gruppo di oggetti

Il registro pubblica i messaggi di evento seguenti quando un gruppo di oggetti viene aggiunto o rimosso in un altro gruppo di oggetti.

- `$aws/events/thingGroupHierarchy/thingGroup/`*`parentThingGroupName`*`/childThingGroup/`*`childThingGroupName`*`/added`
- `$aws/events/thingGroupHierarchy/thingGroup/`*`parentThingGroupName`*`/childThingGroup/`*`childThingGroupName`*`/removed`

Il messaggio contiene il seguente payload di esempio:

```
{
    "eventType" : "THING_GROUP_HIERARCHY_EVENT",
    "eventId" : "264192c7-b573-46ef-ab7b-489fcd47da41",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|REMOVED",
    "accountId" : "123456789012",
    "thingGroupId" : "8f82a106-6b1d-4331-8984-a84db5f6f8cb",
    "thingGroupName" : "MyRootThingGroup",
    "childGroupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "childGroupName" : "MyChildThingGroup"
}
```

I payload contengono gli attributi seguenti:

eventType

Imposta su "THING_GROUP_HIERARCHY_EVENT".

eventId

ID evento.

timestamp

Timestamp UNIX del momento in cui si è verificato l'evento.

operation

`ADDED` quando un oggetto viene aggiunto a un gruppo di oggetti. `REMOVED` quando un oggetto viene rimosso da un gruppo di oggetti.

accountId

Il tuo Account AWS documento d'identità.

thingGroupId

ID del gruppo di oggetti padre.

thingGroupName

Nome del gruppo di oggetti padre.

childGroupId

ID del gruppo di oggetti figlio.

childGroupName

Nome del gruppo di oggetti figlio.

Eventi del servizio Jobs

Il servizio AWS IoT Jobs pubblica su argomenti riservati sul protocollo MQTT quando i lavori sono in sospeso, completati o annullati e quando un dispositivo segnala un esito positivo o negativo durante l'esecuzione di un lavoro. I dispositivi o le applicazioni di gestione e monitoraggio permettono di tenere traccia dello stato dei processi sottoscrivendo questi argomenti.

Come abilitare eventi processi

I messaggi di risposta del servizio AWS IoT Jobs non passano attraverso il broker di messaggi e non possono essere sottoscritti da altri client o regole. Per effettuare la sottoscrizione ai messaggi relativi all'attività del processo, utilizza gli argomenti `notify` e `notify-next`. Per ulteriori informazioni sugli argomenti dei processi, consulta Argomenti di processo.

Per ricevere notifiche sugli aggiornamenti dei lavori, abilita questi eventi di lavoro utilizzando o utilizzando l'API o la CLI. Console di gestione AWS Per ulteriori informazioni, consulta Abilita eventi per AWS IoT.

Come funzionano gli eventi di processo

Poiché l'eliminazione e l'annullamento di un processo potrebbero richiedere alcuni minuti, vengono inviati due messaggi per indicare l'inizio e la fine di una richiesta. Ad esempio, quando si avvia una richiesta di annullamento, viene inviato un messaggio all'argomento `$aws/events/job/jobID/cancellation_in_progress`. Quando la richiesta di annullamento è completa, viene inviato un messaggio all'argomento `$aws/events/job/jobID/canceled`.

Lo stesso processo si verifica per una richiesta di eliminazione di un processo. Le applicazioni di gestione e monitoraggio permettono di tenere traccia dello stato dei processi sottoscrivendo questi argomenti. Per ulteriori informazioni sulla pubblicazione e sulla sottoscrizione di argomenti MQTT, consulta the section called “Protocolli di dispositivo di comunicazione”.

Tipi di eventi di processo

Gli esempi seguenti mostrano i diversi tipi di eventi relativi ai job:

Job Completed/Canceled/Deleted

Il servizio AWS IoT Jobs pubblica un messaggio su un argomento MQTT quando un lavoro viene completato, annullato, eliminato o quando sono in corso annullamenti o eliminazioni:

- `$aws/events/job/jobID/completed`
- `$aws/events/job/jobID/canceled`
- `$aws/events/job/jobID/deleted`
- `$aws/events/job/jobID/cancellation_in_progress`
- `$aws/events/job/jobID/deletion_in_progress`

Il messaggio `completed` contiene il seguente payload di esempio:

```
{
  "eventType": "JOB",
  "eventId": "7364ffd1-8b65-4824-85d5-6c14686c97c6",
  "timestamp": 1234567890,
  "operation": "completed",
  "jobId": "27450507-bf6f-4012-92af-bb8a1c8c4484",
  "status": "COMPLETED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/a39f6f91-70cf-4bd2-a381-9c66df1a80d0",
```

```
    "arn:aws:iot:us-east-1:123456789012:thinggroup/2fc4c0a4-6e45-4525-
a238-0fe8d3dd21bb"
  ],
  "description": "My Job Description",
  "completedAt": 1234567890123,
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123,
  "jobProcessDetails": {
    "numberOfCanceledThings": 0,
    "numberOfRejectedThings": 0,
    "numberOfFailedThings": 0,
    "numberOfRemovedThings": 0,
    "numberOfSucceededThings": 3
  }
}
```

Il messaggio `canceled` contiene il seguente payload di esempio.

```
{
  "eventType": "JOB",
  "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
  "timestamp": 1234567890,
  "operation": "canceled",
  "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
  "status": "CANCELED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
    "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
  ],
  "description": "My job description",
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123
}
```

Il messaggio `deleted` contiene il seguente payload di esempio.

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
```

```
      "operation": "deleted",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "DELETED",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

Il messaggio `cancellation_in_progress` contiene il seguente payload di esempio:

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
      "operation": "cancellation_in_progress",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "CANCELLATION_IN_PROGRESS",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

Il messaggio `deletion_in_progress` contiene il seguente payload di esempio:

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
```

```
      "timestamp": 1234567890,
      "operation": "deletion_in_progress",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "DELETION_IN_PROGRESS",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

Stato terminale dell'esecuzione del processo

Il servizio AWS IoT Jobs pubblica un messaggio quando un dispositivo aggiorna l'esecuzione di un lavoro allo stato del terminale:

- `$aws/events/jobExecution/`*`jobID`*`/succeeded`
- `$aws/events/jobExecution/`*`jobID`*`/failed`
- `$aws/events/jobExecution/`*`jobID`*`/rejected`
- `$aws/events/jobExecution/`*`jobID`*`/canceled`
- `$aws/events/jobExecution/`*`jobID`*`/timed_out`
- `$aws/events/jobExecution/`*`jobID`*`/removed`
- `$aws/events/jobExecution/`*`jobID`*`/deleted`

Il messaggio contiene il seguente payload di esempio:

```
{
  "eventType": "JOB_EXECUTION",
  "eventId": "cca89fa5-8a7f-4ced-8c20-5e653afb3572",
  "timestamp": 1234567890,
  "operation": "succeeded|failed|rejected|canceled|removed|timed_out",
  "jobId": "154b39e5-60b0-48a4-9b73-f6f8dd032d27",
  "thingArn": "arn:aws:iot:us-east-1:123456789012:myThing/6d639fbc-8f85-4a90-924d-
a2867f8366a7",
  "status": "SUCCEEDED|FAILED|REJECTED|CANCELED|REMOVED|TIMED_OUT",
```

```
  "statusDetails": {
    "key": "value"
  }
}
```

Eventi di esecuzione dei comandi

AWS IoT pubblica messaggi di eventi su argomenti MQTT quando l'esecuzione dei comandi cambia lo stato. È possibile utilizzare questi eventi per monitorare l'avanzamento dell'esecuzione dei comandi e creare applicazioni che rispondono ai cambiamenti di stato.

Argomenti degli eventi di esecuzione dei comandi

Gli eventi di esecuzione dei comandi sono pubblicati nel seguente argomento MQTT:

```
$aws/events/commandExecution/commandId/status
```

Dove:

- *commandId*è l'identificatore del comando.
- *status*è lo stato dell'esecuzione del comando. I valori validi sono: `CREATED`, `IN_PROGRESS`, `SUCCEEDED`, `FAILED`, `REJECTED`, `TIMED_OUT`.

Per ricevere notifiche per tutti i comandi e tutti gli stati, sottoscrivi il seguente argomento utilizzando caratteri jolly:

```
$aws/events/commandExecution/+/#
```

Messaggio relativo all'evento di esecuzione del comando

Quando lo stato di esecuzione di un comando cambia, AWS IoT pubblica un messaggio di evento nell'argomento MQTT corrispondente. Il messaggio contiene il seguente payload di esempio:

```
{
    "executionId": "2bd65c51-4cfd-49e4-9310-d5cbfdbc8554",
    "status": "FAILED",
    "statusReason": {
        "reasonCode": "DEVICE_TOO_BUSY",
        "reasonDescription": ""
```

```
    },
    "eventType": "COMMAND_EXECUTION",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/0b9d9ddf-
e873-43a9-8e2c-9fe004a90086",
    "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/5006c3fc-
de96-4def-8427-7eee36c6f2bd",
    "timestamp": 1717708862107
}
```

Il payload contiene gli attributi seguenti:

ID di esecuzione

Un identificatore univoco per l'esecuzione del comando (stringa).

status

Lo stato dell'esecuzione del comando. I valori validi sono: `CREATED`, `IN_PROGRESS`, `SUCCEEDED`, `FAILED`, `REJECTED`, `TIMED_OUT`.

StatusReason

Un oggetto contenente informazioni aggiuntive sullo stato, se disponibili. Contiene `reasonCode` e `reasonDescription` campi.

eventType

Impostato su «COMMAND_EXECUTION».

COMANDOARN

L'Amazon Resource Name (ARN) del comando.

TargetARN

L'ARN del dispositivo di destinazione (cosa o client) per l'esecuzione del comando.

timestamp

Timestamp UNIX del momento in cui si è verificato l'evento.

Eventi del ciclo di vita

AWS IoT può pubblicare eventi del ciclo di vita sugli argomenti MQTT. Questi eventi sono disponibili per impostazione predefinita e non possono essere disabilitati.

> **Note**
>
> È possibile che i messaggi del ciclo di vita non vengano inviati in ordine. Potresti anche ricevere messaggi duplicati.
> `thingName`sarà incluso solo se il client si connette utilizzando la funzione oggetto esclusiva.

In questo argomento:

- Eventi di connessione/disconnessione
- Evento di tentativo di connessione fallito
- Eventi di sottoscrizione/annullamento della sottoscrizione

Eventi di connessione/disconnessione

> **Note**
>
> Con l'indicizzazione del parco veicoli di AWS IoT Device Management, puoi cercare elementi, eseguire query aggregate e creare gruppi dinamici basati sugli eventi degli oggetti. Connect/Disconnect Per ulteriori informazioni, consulta Indicizzazione del parco istanze.

AWS IoT pubblica un messaggio sui seguenti argomenti MQTT quando un client si connette o si disconnette:

- `$aws/events/presence/connected/`*`clientId`* - Client connesso al broker di messaggi.
- `$aws/events/presence/disconnected/`*`clientId`* - Client disconnesso dal broker di messaggi.

Di seguito è riportato un elenco di elementi JSON contenuti nei connection/disconnection messaggi pubblicati sull'argomento. `$aws/events/presence/connected/`*`clientId`*

clientId

ID del client che si connette o si disconnette.

> **Note**
>
> I client IDs che contengono # o + non ricevono eventi del ciclo di vita.

thingName

Il nome del tuo dispositivo IoT. `thingName`sarà incluso solo se il client si connette utilizzando la funzione oggetto esclusiva.

clientInitiatedDisconnect

True se il client ha avviato la disconnessione. In caso contrario, false. Presente solo nei messaggi di disconnessione.

disconnectReason

Il motivo per cui il client viene disconnesso. Presente solo nei messaggi di disconnessione. La tabella seguente contiene valori validi e indica se il broker invierà Messaggi Last Will and Testament (LWT) quando si verifica la disconnessione.

Motivo di disconnessione	Description	Il broker invierà i messaggi LWT
`AUTH_ERROR`	Il client non è riuscito a eseguire l'autenticazione oppure l'autorizzazione non è riuscita.	Sì[1]
`CLIENT_INITIATED_DISCONNECT`	Il client indica che si disconnette. Il client può farlo inviando un pacchetto di `DISCONNECT` controllo MQTT o un `Close frame` se il client utilizza una WebSocket connessione.	No
`CLIENT_ERROR`	Il client ha rilevato un errore che causa la disconnessione. Ad esempio, il client viene disconnesso per l'invio di più di 1 pacchetto `CONNECT` MQTT sulla stessa connessione o se tenta di pubblicare con un payload che supera il limite del payload.	Sì
`CONNECTION_LOST`	La connessione client-server viene interrotta. Ciò può verificarsi durante un periodo di alta latenza	Sì

Motivo di disconnessione	Description	Il broker invierà i messaggi LWT
	di rete o quando la connessione Internet viene persa.	
DUPLICATE_CLIENTID	Il client utilizza un ID client già in uso. In questo caso, il client già connesso verrà disconnesso con questo motivo di disconnessione.	Sì
FORBIDDEN_ACCESS	Il client non è autorizzato alla connessione. Ad esempio, un client con un indirizzo IP negato non riesce a connettersi.	Sì[1]
MQTT_KEEP_ALIVE_TIMEOUT	Se non viene rilevata alcuna comunicazione client-server per 1,5x del tempo keep-alive del client, il client viene disconnesso.	Sì
SERVER_ERROR	Disconnesso a causa di problemi imprevisti del server.	Sì
SERVER_INITIATED_DISCONNECT	Il server disconnette intenzionalmente un client per motivi operativi.	Sì
API_INITIATED_DISCONNECT	Il client è stato disconnesso utilizzando l'API. `DeleteConnection`	Sì[2]
THROTTLED	Il client viene disconnesso per il superamento di un limite di throttling.	Sì
WEBSOCKET_TTL_EXPIRATION	Il client è disconnesso perché a è WebSocket stato connesso più a lungo del suo time-to-live valore.	Sì
CUSTOMAUTH_TTL_EXPIRATION	Il client è disconnesso perché è stato connesso più a lungo del time-to-live valore del relativo autorizzatore personalizzato.	Sì

[1] Se il dispositivo ha una connessione attiva prima di ricevere questo errore.

[2] Per evitare messaggi di Last Will and Testament (LWT), imposta in modo `preventWillMessage=true` da sovrascrivere il comportamento di invio LWT predefinito dell'`DeleteConnection`API.

eventType

Tipo di evento. I valori validi sono `connected` e `disconnected`.

ipAddress

L'indirizzo IP del client di connessione. Può essere in o formato. IPv4 IPv6 Presente solo nei messaggi di connessione.

principalIdentifier

Credenziale usata per l'autenticazione. Per i certificati di autenticazione reciproca TLS, si tratta dell'ID certificato. Per altre connessioni, si tratta delle credenziali IAM.

sessionIdentifier

Un identificatore univoco globale AWS IoT che esiste per tutta la durata della sessione.

timestamp

Un'approssimazione del momento in cui si è verificato l'evento.

versionNumber

Numero di versione per l'evento del ciclo di vita. Consiste nell'aumentare in maniera monotona un valore intero lungo per ogni connessione dell'ID client. Il numero di versione può essere utilizzato da un sottoscrittore per dedurre l'ordine degli eventi del ciclo di vita.

> **Note**
>
> I messaggi di connessione e disconnessione per una connessione client hanno lo stesso numero di versione.
> Il numero di versione potrebbe ignorare valori e non è garantito che aumenti di 1 in modo costante per ogni evento.
> Se un client non è connesso per circa un'ora, il numero di versione viene reimpostato su 0. Per le sessioni persistenti, il numero di versione viene reimpostato su 0 dopo che un client è stato disconnesso più a lungo del periodo configurato time-to-live (TTL) per la sessione persistente.

Un messaggio di connessione ha la seguente struttura.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1573002230757,
    "eventType": "connected",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
    "ipAddress": "192.0.2.0",
    "versionNumber": 0
}
```

Un messaggio di disconnessione ha la seguente struttura.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1573002340451,
    "eventType": "disconnected",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
    "clientInitiatedDisconnect": true,
    "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
    "versionNumber": 0
}
```

Gestione delle disconnessioni client

Secondo le best practice, occorre sempre avere uno stato di attesa implementato per gli eventi del ciclo di vita, inclusi i messaggi Last Will and Testament (LWT). Quando si riceve un messaggio di disconnessione, il codice deve attendere un periodo di tempo e verificare che un dispositivo è ancora offline prima di effettuare operazioni. A questo scopo, è possibile utilizzare Code di ritardo Amazon SQS. Quando un client riceve un LWT o un evento del ciclo di vita, è possibile accodare un messaggio, ad esempio per 5 secondi. Quando il messaggio diventa disponibile e viene elaborato (da Lambda o da un altro servizio), è possibile controllare innanzitutto se il dispositivo è ancora realmente offline prima di effettuare ulteriori operazioni.

Evento di tentativo di connessione fallito

AWS IoT pubblica un messaggio sul seguente argomento MQTT quando un client non è autorizzato a connettersi o quando è configurato un testamento e un testamento e il client non è autorizzato a pubblicare su quell'argomento di ultimo testamento.

```
$aws/events/presence/connect_failed/clientId
```

Di seguito è riportato un elenco di elementi JSON contenuti nei messaggi di autorizzazione alla connessione pubblicati sull'argomento. `$aws/events/presence/connect_failed/clientId`

clientId

L'ID client del client che ha tentato e non è riuscito a connettersi.

> **Note**
>
> I client IDs che contengono # o + non ricevono eventi del ciclo di vita.

thingName

Il nome del tuo dispositivo IoT. `thingName`sarà incluso solo se il client si connette utilizzando la funzione oggetto esclusiva.

timestamp

Un'approssimazione del momento in cui si è verificato l'evento.

eventType

Tipo di evento. Il valore valido è`connect_failed`.

connectFailureReason

Il motivo per cui la connessione non riesce. Il valore valido è`AUTHORIZATION_FAILED`.

principalIdentifier

Credenziale usata per l'autenticazione. Per i certificati di autenticazione reciproca TLS, si tratta dell'ID certificato. Per altre connessioni, si tratta delle credenziali IAM.

sessionIdentifier

Un identificatore univoco globale AWS IoT che esiste per tutta la durata della sessione.

ipAddress

L'indirizzo IP del client di connessione. Può essere in IPv4 o IPv6 formato. Presente solo nei messaggi di connessione.

Un messaggio di errore di connessione ha la seguente struttura.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1460065214626,
    "eventType": "connect_failed",
    "connectFailureReason": "AUTHORIZATION_FAILED",
    "principalIdentifier": "12345678901234567890123456789012",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "ipAddress" : "192.0.2.0"
}
```

Eventi di sottoscrizione/annullamento della sottoscrizione

AWS IoT pubblica un messaggio sul seguente argomento MQTT quando un client sottoscrive o annulla l'iscrizione a un argomento MQTT:

```
$aws/events/subscriptions/subscribed/clientId
```

or

```
$aws/events/subscriptions/unsubscribed/clientId
```

Dove `clientId` è l'ID client MQTT che si connette al broker di messaggi AWS IoT .

Il messaggio pubblicato in questo argomento ha la struttura seguente:

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
```

```
    "timestamp": 1460065214626,
    "eventType": "subscribed" | "unsubscribed",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
    "topics" : ["foo/bar","device/data","dog/cat"]
}
```

Di seguito è riportato un elenco di elementi JSON contenuti nei messaggi di cui è stata eseguita o annullata la sottoscrizione pubblicati negli argomenti `$aws/events/subscriptions/subscribed/`*`clientId`* e `$aws/events/subscriptions/unsubscribed/`*`clientId`*.

clientId

ID del client che esegue la sottoscrizione o l'annullamento della sottoscrizione.

> **Note**
>
> I client IDs che contengono # o + non ricevono eventi del ciclo di vita.

thingName

Il nome del tuo dispositivo IoT. `thingName`sarà incluso solo se il client si connette utilizzando la funzione oggetto esclusiva.

eventType

Tipo di evento. I valori validi sono `subscribed` e `unsubscribed`.

principalIdentifier

Credenziale usata per l'autenticazione. Per i certificati di autenticazione reciproca TLS, si tratta dell'ID certificato. Per altre connessioni, si tratta delle credenziali IAM.

sessionIdentifier

Un identificatore univoco globale AWS IoT che esiste per tutta la durata della sessione.

timestamp

Un'approssimazione del momento in cui si è verificato l'evento.

topics

Matrice degli argomenti MQTT sottoscritti dal client.

> **Note**
>
> È possibile che i messaggi del ciclo di vita non vengano inviati in ordine. Potresti anche ricevere messaggi duplicati.

Risoluzione dei problemi AWS IoT

Aiutaci a migliorare questo argomento

Facci sapere che cosa contribuirebbe a migliorarlo

Le informazioni seguenti possono risultare utili per risolvere i problemi comuni di AWS IoT.

Processi

- AWS IoT Core guida alla risoluzione dei problemi
- AWS IoT Device Management guida alla risoluzione dei problemi
- AWS IoT Guida alla risoluzione di Device Advisor
- AWS IoT errori

AWS IoT Core guida alla risoluzione dei problemi

Aiutaci a migliorare questo argomento

Facci sapere che cosa contribuirebbe a migliorarlo

Questa è la sezione relativa alla risoluzione dei problemi per AWS IoT Core.

Argomenti

- Diagnosi dei problemi di connettività
- Diagnosi dei problemi relativi alle regole
- Diagnosi dei problemi relativi a Shadows
- Diagnosi dei problemi relativi alle operazioni del flusso di input Salesforce IoT
- Diagnosi dei limiti di flusso
- Risoluzione degli errori di disconnessione del parco istanze dei dispositivi

Diagnosi dei problemi di connettività

> Aiutaci a migliorare questo argomento
>
> Facci sapere che cosa contribuirebbe a migliorarlo

Una connessione riuscita a AWS IoT richiede:

- Una connessione valida
- Un certificato valido e attivo
- Una policy che consenta la connessione e il funzionamento desiderati

Connessione

Come trovo l'endpoint corretto?

- Il valore `endpointAddress` restituito da `aws iot describe-endpoint --endpoint-type iot:Data-ATS`

  or

- Il valore `domainName` restituito da `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"`

Come trovo il valore di SNI (Server Name Indication) corretto?

Il valore SNI corretto è il valore `endpointAddress` restituito dai comandi describe-endpoint o il valore `domainName` restituito dai comandi describe-domain-configuration. È lo stesso indirizzo dell'endpoint del passaggio precedente. Quando si collegano i dispositivi a AWS IoT Core, i client possono inviare l'estensione SNI (Server Name Indication), che non è richiesta ma è altamente consigliata. Per utilizzare funzionalità come la registrazione con più account, i domini personalizzati e gli endpoint VPC, è necessario utilizzare l'estensione SNI. Per ulteriori informazioni, vedere Transport Security in AWS IoT.

Come posso risolvere un problema di connettività che persiste?

È possibile utilizzare AWS Device Advisor per la risoluzione dei problemi. I test predefiniti di Device Advisor consentono di convalidare il software del dispositivo in base alle best practice per l'utilizzo di TLS, MQTT, AWS IoT Device Shadow e Processi AWS IoT.

Ecco un link ai contenuti esistenti di Device Advisor.

Autenticazione

I dispositivi devono essere autenticati per connettersi agli endpoint. AWS IoT Per i dispositivi che utilizzano Certificati client X.509 l'autenticazione, i certificati devono essere registrati AWS IoT ed essere attivi.

In che modo i miei dispositivi autenticano gli AWS IoT endpoint?

Aggiungi il certificato AWS IoT CA al trust store del tuo cliente. Fai riferimento alla documentazione sull'Autenticazione del server in AWS IoT Core e segui i collegamenti per scaricare il certificato CA appropriato.

Cosa viene controllato quando un dispositivo si connette a AWS IoT?

Quando un dispositivo tenta di connettersi a AWS IoT:

1. AWS IoT verifica la presenza di un certificato e di un valore SNI (Server Name Indication) validi.
2. AWS IoT verifica che il certificato utilizzato sia registrato presso l' AWS IoT Account e che sia stato attivato.
3. Quando un dispositivo tenta di eseguire un'azione AWS IoT, ad esempio sottoscrivere o pubblicare un messaggio, la politica allegata al certificato utilizzato per la connessione viene verificata per confermare che il dispositivo sia autorizzato a eseguire tale azione.

In che modo è possibile convalidare un certificato configurato correttamente?

Usa il comando `s_client` OpenSSL per testare una connessione all'endpoint di AWS IoT :

```
openssl s_client -connect custom_endpoint.iot.aws-region.amazonaws.com:8443 -
CAfile CA.pem -cert cert.pem -key privateKey.pem
```

Per ulteriori informazioni sull'utilizzo di `openssl s_client`, consulta la documentazione di OpenSSL s_client.

Come posso verificare lo stato di un certificato?

- Elenca i certificati

  Se non conosci l'ID del certificato, puoi visualizzare lo stato di tutti i certificati utilizzando il comando `aws iot list-certificates`.

- Visualizzazione dei dettagli di un certificato

  Se conosci l'ID del certificato, questo comando mostra informazioni più dettagliate sul certificato.

  ```
  aws iot describe-certificate --certificate-id "certificateId"
  ```

- Rivedi il certificato nella AWS IoT console

  Nella console AWS IoT del menu a sinistra seleziona Secure (Sicurezza) e quindi scegli Certificates (Certificati).

  Scegli il certificato che stai utilizzando per connetterti dall'elenco per aprire la relativa pagina dei dettagli.

  Nella pagina dei dettagli del certificato, è possibile visualizzarne lo stato attuale.

  Lo stato del certificato può essere modificato utilizzando il menu Actions (Operazioni) nell'angolo in alto a destra della pagina dei dettagli.

Autorizzazione

AWS IoT risorse utilizzate AWS IoT Core politiche per autorizzare tali risorse a eseguire azioni. Affinché un'azione sia autorizzata, alle AWS IoT risorse specificate deve essere allegato un documento di policy che conceda il permesso di eseguire tale azione.

Ho ricevuto una risposta PUBNACK o SUBNACK dal broker. Cosa devo fare?

Assicurati che sia allegata una politica al certificato che stai utilizzando per la chiamata AWS IoT. Tutte le publish/subscribe operazioni sono negate per impostazione predefinita.

Assicurati che la policy collegata autorizzi le operazioni che stai cercando di eseguire.

Assicurati che la policy collegata autorizzi le risorse che stanno tentando di eseguire le operazioni autorizzate.

Ho una voce AUTHORIZATION_FAILURE nei miei registri.

Assicurati che ci sia una politica allegata al certificato che stai usando per chiamare AWS IoT. Tutte le publish/subscribe operazioni sono negate per impostazione predefinita.

Assicurati che la policy collegata autorizzi le operazioni che stai cercando di eseguire.

Assicurati che la policy collegata autorizzi le risorse che stanno tentando di eseguire le operazioni autorizzate.

Come faccio a verificare ciò che la policy autorizza?

Nella AWS IoT console, nel menu a sinistra, scegli Sicurezza, quindi scegli Certificati.

Scegli il certificato che stai utilizzando per connetterti dall'elenco per aprire la relativa pagina dei dettagli.

Nella pagina dei dettagli del certificato, è possibile visualizzarne lo stato attuale.

Nel menu a sinistra della pagina dei dettagli del certificato, scegli Policies (Policy) per visualizzare le policy allegate al certificato.

Scegli la policy desiderata per visualizzarne la pagina dei dettagli.

Nella pagina dei dettagli della policy, esamina il Documento di policy per vedere cosa autorizza.

Scegli Edit policy document (Modifica del documento della policy) per apportare modifiche al documento della policy.

Sicurezza e identità

Quando fornisci i certificati del server per la configurazione AWS IoT personalizzata del dominio, i certificati hanno un massimo di quattro nomi di dominio.

Per ulteriori informazioni, consulta Endpoint e quote per AWS IoT Core.

Diagnosi dei problemi relativi alle regole

> Aiutaci a migliorare questo argomento
>
> Facci sapere che cosa contribuirebbe a migliorarlo

Questa sezione indica gli oggetti da verificare nel caso si riscontri un problema con la regola.

Configurazione dei CloudWatch log per la risoluzione dei problemi

Il modo migliore per risolvere i problemi relativi alle regole consiste nell'utilizzare i registri. CloudWatch Quando abiliti CloudWatch Logs for AWS IoT, puoi vedere quali regole vengono attivate

e il loro successo o fallimento. Puoi inoltre ottenere informazioni sulla corrispondenza delle condizioni delle clausole WHERE. Per ulteriori informazioni, consulta Monitora AWS IoT utilizzando i log CloudWatch.

La maggior parte dei problemi comuni delle regole riguarda l'autorizzazione. I log mostrano se il tuo ruolo non è autorizzato a svolgere AssumeRole sulla risorsa. Di seguito è illustrato un esempio di log generato dal logging granulare:

```
{
    "timestamp": "2017-12-09 22:49:17.954",
    "logLevel": "ERROR",
    "traceId": "ff563525-6469-506a-e141-78d40375fc4e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleExecution",
    "clientId": "iotconsole-123456789012-3",
    "topicName": "test-topic",
    "ruleName": "rule1",
    "ruleAction": "DynamoAction",
    "resources": {
        "ItemHashKeyField": "id",
        "Table": "trashbin",
        "Operation": "Insert",
        "ItemHashKeyValue": "id",
        "IsPayloadJSON": "true"
    },
    "principalId": "ABCDEFG1234567ABCD890:outis",
    "details": "User: arn:aws:sts::123456789012:assumed-role/dynamo-
testbin/5aUMInJH is not authorized to perform: dynamodb:PutItem on
 resource: arn:aws:dynamodb:us-east-1:123456789012:table/testbin (Service:
 AmazonDynamoDBv2; Status Code: 400; Error Code: AccessDeniedException; Request ID:
 AKQJ987654321AKQJ123456789AKQJ987654321AKQJ987654321)"
}
```

Di seguito è illustrato un esempio di log generato dal logging globale:

```
2017-12-09 22:49:17.954 TRACEID:ff562535-6964-506a-e141-78d40375fc4e
PRINCIPALID:ABCDEFG1234567ABCD890:outis [ERROR] EVENT:DynamoActionFailure
TOPICNAME:test-topic CLIENTID:iotconsole-123456789012-3
MESSAGE:Dynamo Insert record failed. The error received was User:
 arn:aws:sts::123456789012:assumed-role/dynamo-testbin/5aUMInJI is not authorized to
```

```
 perform: dynamodb:PutItem on resource: arn:aws:dynamodb:us-east-1:123456789012:table/
testbin
(Service: AmazonDynamoDBv2; Status Code: 400; Error Code: AccessDeniedException;
 Request ID: AKQJ987654321AKQJ987654321AKQJ987654321AKQJ987654321).
Message arrived on: test-topic, Action: dynamo, Table: trashbin, HashKeyField: id,
 HashKeyValue: id, RangeKeyField: None, RangeKeyValue: 123456789012
No newer events found at the moment. Retry.
```

Per ulteriori informazioni, consulta the section called “Visualizzazione dei AWS IoT log nella console CloudWatch”.

Diagnosi di servizi esterni

I servizi esterni sono controllati dall'utente finale. Prima dell'esecuzione della regola, assicurati che i servizi esterni collegati alla regola siano configurati e dispongano di unità di capacità e velocità effettiva sufficiente per l'applicazione.

Diagnosi dei problemi SQL

Se la query SQL non restituisce i dati che ci si aspetta:

- Esamina i registri per individuare gli eventuali messaggi di errore.
- Verifica che la sintassi SQL corrisponda al documento JSON nel messaggio.

  Esamina i nomi degli oggetti e delle proprietà utilizzati nella query con quelli utilizzati nel documento JSON del payload dei messaggi dell'argomento. Per ulteriori informazioni sul formato JSON nelle query SQL, consulta Estensioni JSON.
- Verifica se i nomi degli oggetti o delle proprietà JSON includono caratteri riservati o numerici.

  Per ulteriori informazioni sui caratteri riservati nei riferimenti agli oggetti JSON nelle query SQL, consulta Estensioni JSON.

Diagnosi dei problemi relativi a Shadows

> Aiutaci a migliorare questo argomento
>
> Facci sapere che cosa contribuirebbe a migliorarlo

Diagnosi di Shadows

Problema	Linee guida per la risoluzione dei problemi
Il documento shadow di un dispositivo viene rifiutato con `Invalid JSON document`.	Se non hai familiarità con JSON, modifica gli esempi forniti in questa guida per uso personale. Per ulteriori informazioni, consulta Esempi di documenti shadow.
Il codice JSON inviato è corretto, ma non viene archiviato, o viene archiviato solo in parte, nel documento della copia shadow del dispositivo.	Assicurati di rispettare le linee guida di formattazione per JSON. Solo i campi JSON nelle sezioni `desired` e `reported` vengono archiviati. I contenuti JSON (anche se formalmente corretti) al di fuori di queste sezioni vengono ignorati.
Si è verificato un errore che indica che la copia shadow del dispositivo supera le dimensioni permesse.	La copia shadow di un dispositivo supporta solo fino a 8 KB di dati. Prova ad accorciare i nomi di campo all'interno del documento JSON o semplicemente crea più copie shadow creando più oggetti. A un dispositivo può essere things/ shadows associato un numero illimitato di dispositivi. L'unico requisito è che il nome di ogni oggetto deve essere univoco nell'account.
Quando si riceve una copia shadow di un dispositivo, le sue dimensioni sono superiori a 8 KB. Come è possibile?	Al ricevimento, il AWS IoT servizio aggiunge metadati all'ombra del dispositivo. Il servizio include questi dati nella risposta, che non vengono tuttavia conteggiati per il raggiungi mento del limite di 8 KB. Solo i dati per gli stati `desired` e `reported` all'interno del documento sullo stato inviato alla copia shadow del dispositivo vengono conteggiati per il raggiungimento del limite.
La richiesta è stata rifiutata a causa di una versione errata. Cosa devo fare?	Esegui un'operazione GET per eseguire la sincronizzazione all'ultima versione del documento sullo stato. Quando usi MQTT,

Problema	Linee guida per la risoluzione dei problemi
	sottoscrivi l'argomento ./update/accepted per ricevere la versione più recente del documento JSON e le notifiche sulle modifiche dello stato.
Il timestamp è disattivato per alcuni secondi.	Il timestamp per i singoli campi e l'intero documento JSON viene aggiornato quando il documento viene ricevuto dal AWS IoT servizio o quando il documento di stato viene pubblicato su. /messaggio. update/accepted and ./update/delta I messaggi possono essere ritardati nella rete e in questo caso il timestamp è disattivato per alcuni secondi.
Il dispositivo può pubblicare e sottoscrivere gli argomenti delle copie shadow corrispondenti, ma quando si tenta di aggiornare il documento della copia shadow tramite l'API REST HTTP, si riceve l'errore HTTP 403.	Assicurati di aver creato delle politiche in IAM per consentire l'accesso a questi argomenti e all'azione corrispondente (UPDATE/GET/DELETE) per le credenziali che stai utilizzan do. Le policy IAM e le policy dei certificati sono indipendenti.
Altri problemi.	Il servizio Device Shadow registra gli errori in CloudWatch Logs. Per identificare i problemi relativi al dispositivo e alla configurazione, abilita CloudWatch i registri e visualizza i registri per le informazioni di debug.

Diagnosi dei problemi relativi alle operazioni del flusso di input Salesforce IoT

Aiutaci a migliorare questo argomento

Facci sapere che cosa contribuirebbe a migliorarlo

Traccia di esecuzione

Come è possibile visualizzare la traccia di esecuzione di un'operazione di Salesforce?

Consulta la sezione Monitora AWS IoT utilizzando i log CloudWatch. Una volta attivati i log, è possibile visualizzare la traccia di esecuzione dell'operazione di Salesforce.

Esito dell'operazione

Come è possibile controllare che i messaggi siano stati inviati correttamente a un flusso di input Salesforce IoT?

Visualizza i log generati dall'esecuzione dell'azione Salesforce in Logs. CloudWatch Se vedi`Action executed successfully`, significa che il motore AWS IoT delle regole ha ricevuto la conferma da Salesforce IoT che il messaggio è stato inviato correttamente al flusso di input di destinazione.

Se si verificano problemi con la piattaforma Salesforce IoT, contatta il supporto di Salesforce IoT.

Cosa è possibile fare se i messaggi non sono stati inviati correttamente a un flusso di input Salesforce IoT?

Visualizza i log generati dall'esecuzione dell'azione Salesforce in Logs. CloudWatch A seconda della voce di log, puoi provare le operazioni seguenti:

`Failed to locate the host`

Controlla che il parametro `url` dell'operazione sia corretto e che il flusso di input Salesforce IoT esista.

`Received Internal Server Error from Salesforce`

Riprova. Se il problema persiste, contatta il supporto di Salesforce IoT.

`Received Bad Request Exception from Salesforce`

Controlla se sono presenti errori nel payload inviato.

`Received Unsupported Media Type Exception from Salesforce`

Al momento, Salesforce IoT non supporta un payload binario. Controlla che venga inviato un payload JSON.

`Received Unauthorized Exception from Salesforce`

Controlla che il parametro `token` dell'operazione sia corretto e che il token sia ancora valido.

`Received Not Found Exception from Salesforce`

Controlla che il parametro `url` dell'operazione sia corretto e che il flusso di input Salesforce IoT esista.

Se ricevi un errore che non è elencato qui, contatta l' AWS IoT assistenza.

Diagnosi dei limiti di flusso

Risoluzione dei problemi relativi al «Limite di streaming superato per il tuo AWS account»

Se visualizzi l'errore `"Error: You have exceeded the limit for the number of streams in your AWS account."`, puoi eliminare i flussi inutilizzati nel tuo account invece di richiedere un aumento del limite.

Per ripulire uno stream inutilizzato che hai creato utilizzando AWS CLI o l'SDK:

```
aws iot delete-stream –stream-id value
```

Per ulteriori informazioni, consulta delete-stream.

> **Note**
>
> Puoi usare il `list-streams` comando per trovare lo stream. IDs

Risoluzione degli errori di disconnessione del parco istanze dei dispositivi

> **Aiutaci a migliorare questo argomento**
>
> Facci sapere che cosa contribuirebbe a migliorarlo

AWS IoT le disconnessioni del parco dispositivi possono avvenire per diversi motivi. Questo articolo spiega come diagnosticare un motivo di disconnessione e come gestire le disconnessioni causate dalla regolare manutenzione del AWS IoT servizio o da un limite di limitazione.

Diagnosi del motivo di una disconnessione

È possibile controllare il gruppo di AWSIotlog LogSv2 CloudWatchper identificare il motivo della disconnessione nel campo della voce di registro. `disconnectReason`

Puoi anche utilizzare la funzione degli eventi AWS IoT del ciclo di vita per identificare il motivo della disconnessione. Se ti sei iscritto all'evento di disconnessione del ciclo di vita (`$aws/events/presence/disconnected/clientId`), riceverai una notifica quando si verifica la disconnessione. AWS IoT È possibile identificare il motivo della disconnessione nel campo `disconnectReason` della notifica.

Per ulteriori informazioni, consulta le voci di CloudWatch AWS IoT registro e gli eventi del ciclo di vita.

Per risolvere i problemi di disconnessione dovuti alla manutenzione del servizio AWS IoT

Le disconnessioni causate dalla manutenzione AWS IoT del servizio vengono registrate come `SERVER_INITIATED_DISCONNECT` evento del ciclo di vita e. AWS IoT CloudWatch Per gestire queste disconnessioni, modifica la configurazione lato client per assicurarti che i dispositivi possano essere ricollegati automaticamente alla piattaforma. AWS IoT

Risoluzione dei problemi di disconnessione dovuti a un limite di limitazione

Le disconnessioni causate da un limite di limitazione vengono registrate come evento del ciclo di vita di In e. `THROTTLED` AWS IoT CloudWatch Per gestire queste disconnessioni, è possibile richiedere un aumento del limite del broker messaggi man mano che aumenta il numero dei dispositivi.

Per ulteriori informazioni, consulta Broker di messaggi AWS IoT.

AWS IoT Device Management guida alla risoluzione dei problemi

> Aiutaci a migliorare questo argomento
>
> Facci sapere che cosa contribuirebbe a migliorarlo

Questa è la sezione relativa alla risoluzione dei problemi per AWS IoT Device Management.

Argomenti

- AWS IoT Risoluzione dei problemi dei lavori

- Risoluzione dei problemi di indicizzazione della flotta
- AWS IoT Risoluzione dei problemi del Device Management Software Package Catalog

AWS IoT Risoluzione dei problemi dei lavori

Questa è la sezione di risoluzione dei problemi per AWS IoT Jobs.

Come posso individuare un endpoint AWS IoT Jobs?

Come posso localizzare l'endpoint del piano AWS IoT di controllo di Jobs?

AWS IoT Jobs supporta le operazioni dell'API del piano di controllo utilizzando il protocollo HTTPS. Verifica di esserti connesso all'endpoint del piano di controllo corretto utilizzando il protocollo HTTPS.

Per un elenco degli endpoint AWS specifici della regione, consulta AWS IoT Core - control plane endpoints.

Per un elenco degli endpoint del piano di controllo AWS IoT Jobs conformi a FIPS, vedere FIPS Endpoints by Service

> **Note**
>
> AWS IoT Esegue le operazioni e AWS IoT Core condivide gli stessi endpoint specifici della regione. AWS

Come posso localizzare l'endpoint del piano dati di AWS IoT Jobs?

AWS IoT Jobs supporta le operazioni dell'API del piano dati utilizzando i protocolli HTTPS e MQTT. Verifica di esserti connesso all'endpoint del piano dati corretto utilizzando il protocollo HTTPS o MQTT.

- Protocollo HTTPS
  - Utilizza il seguente comando describe-endpointCLI mostrato di seguito o l'API `DescribeEndpoint`REST. Per il tipo di endpoint, usa. `iot:Jobs`

```
aws iot describe-endpoint --endpoint-type iot:Jobs
```

- Protocollo MQTT
  - Utilizza il seguente comando describe-endpointCLI mostrato di seguito o l'API `DescribeEndpoint`REST. Per il tipo di endpoint, usa. `iot:Data-ATS`

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

Per un elenco degli endpoint del piano dati AWS IoT Jobs conformi a FIPS, vedere FIPS Endpoints by Service

Come posso monitorare l'attività di Jobs e fornire metriche AWS IoT ?

Il monitoraggio dell'attività AWS IoT di Jobs tramite Amazon CloudWatch offre visibilità in tempo reale sulle operazioni AWS IoT Jobs in corso e aiuta a controllare i costi con CloudWatch allarmi tramite AWS IoT Rules. È necessario configurare la registrazione prima di poter monitorare l'attività di AWS IoT Jobs e impostare CloudWatch gli allarmi. Per ulteriori informazioni sulla configurazione della registrazione, vedere. Configurare la registrazione AWS IoT

Per ulteriori informazioni su Amazon CloudWatch e su come configurare l'autorizzazione all'uso CloudWatch delle risorse tramite un ruolo utente IAM, consulta Gestione delle identità e degli accessi per Amazon CloudWatch.

Come posso configurare le metriche e il monitoraggio di AWS IoT Jobs utilizzando Amazon CloudWatch?

Per configurare la AWS IoT registrazione, segui i passaggi descritti in Configurare la registrazione. AWS IoT AWS IoT la configurazione della registrazione può essere eseguita nell'API Console di gestione AWS, AWS CLI o. AWS IoT la configurazione della registrazione per gruppi di oggetti specifici deve essere eseguita solo nell'API AWS CLI or.

La sezione AWS IoT Jobs metrics contiene le metriche AWS IoT Jobs utilizzate per monitorare AWS IoT l'attività di Jobs. Spiega come visualizzare le metriche nella e. Console di gestione AWS AWS CLI

Inoltre, puoi impostare CloudWatch allarmi per avvisarti di metriche specifiche che desideri monitorare attentamente. Per indicazioni sulla configurazione degli allarmi, consulta Utilizzo degli CloudWatch allarmi Amazon.

Flotte di dispositivi e risoluzione dei problemi relativi a singoli dispositivi

L'esecuzione di un lavoro mantiene uno stato indefinito **QUEUED**

Quando un'esecuzione di job con uno stato di status di `QUEUED` non passa allo stato di stato logico successivo`IN_PROGRESS`, ad esempio`FAILED`, oppure`TIMED_OUT`, la causa può essere uno dei seguenti scenari:

- Controlla l'attività del dispositivo nei CloudWatch registri che si trovano nella CloudWatch console. Per ulteriori informazioni, consulta Monitoraggio AWS IoT tramite CloudWatch registri.
- Il ruolo IAM associato al job e alla successiva esecuzione del lavoro potrebbero non disporre delle autorizzazioni corrette elencate in una delle dichiarazioni di policy della policy IAM allegate a quel ruolo IAM. Utilizza l'`describe-job`API per identificare il ruolo IAM collegato a quel job e alla successiva esecuzione del job e rivedi la policy IAM per le autorizzazioni corrette. Una volta aggiornate le istruzioni di autorizzazione delle policy, dovresti essere in grado di eseguire il comando `AssumeRole`API sulla risorsa.

Non è stata creata un'esecuzione di lavoro per il mio oggetto o il mio gruppo di oggetti

Quando un job aggiorna il suo stato di stato a`IN_PROGRESS`, inizierà la distribuzione del documento di lavoro su tutti i dispositivi del gruppo di destinazione. Questo aggiornamento dello stato creerà un'esecuzione del lavoro per ogni dispositivo di destinazione. Se non è stata creata un'esecuzione del lavoro per uno dei dispositivi di destinazione, fai riferimento alle seguenti indicazioni:

- Il job ha come target `thing` diretto il job, lo stato del `IN_PROGRESS` job è uguale a e il job è simultaneo? Se tutte e tre le condizioni sono soddisfatte, il processo continua a inviare esecuzioni di lavoro a tutti i dispositivi del gruppo target e quello specifico non `thing` ha ancora ricevuto l'esecuzione del lavoro.
  - Controlla i dispositivi del gruppo di destinazione per il processo e lo stato del lavoro nella Console di AWS gestione o utilizza il comando `describe-job`API.
  - Utilizza il comando `describe-job`API per verificare se la `IsConcurrent` proprietà del job è impostata su true o false. Per ulteriori informazioni, consulta Job limits.
- Non *`thing`* è direttamente preso di mira dal lavoro.
  - Se `Thing` è stato aggiunto a un `ThingGroup` e il lavoro era destinato a`ThingGroup`, verifica `Thing` che faccia parte di`ThingGroup`.

- Se il processo è un processo istantaneo con uno stato di stato pari a `IN_PROGRESS` ed è simultaneo, il processo sta comunque inviando esecuzioni di job a tutti i dispositivi del gruppo di destinazione e quello specifico non `Thing` ha ancora ricevuto l'esecuzione del job.
- Se il processo è un processo continuo con uno stato pari a `IN_PROGRESS` ed è simultaneo, invia comunque esecuzioni di lavori a tutti i dispositivi del gruppo di destinazione e quello specifico non `Thing` ha ancora ricevuto l'esecuzione del lavoro. Solo per i lavori continui, puoi anche rimuovere il `Thing` da `ThingGroup` e quindi aggiungerlo `Thing` nuovamente a. `ThingGroup`
- Se il lavoro è un processo istantaneo con uno stato di stato pari a `IN_PROGRESS` e non è simultaneo, è probabile che la relazione di `ThingGroup` appartenenza `Thing` o di appartenenza non sia riconosciuta da Jobs. AWS IoT Si consiglia di aggiungere alcuni secondi di attesa dopo la `AddThingToThingGroup` chiamata prima di creare la. `Job` In alternativa, è possibile cambiare la selezione del target in `Continuous` modo che il servizio risolva l'evento ritardato `Thing` e il collegamento `ThingGroup` all'iscrizione.

Il nuovo processo non riesce a causa di un errore **`LimitedExceededException`**

Se la creazione del lavoro fallisce con una risposta di errore di`LimitedExceededException`, chiama l'`list-jobs`API ed esamina tutti i lavori `isConcurrent=true` per determinare se hai raggiunto il limite di disponibilità di posti di lavoro. Vedi Job limits per ulteriori informazioni sui lavori simultanei. Per visualizzare i limiti di posti di lavoro simultanei e per richiedere un aumento del limite, consulta Limiti e AWS IoT Device Management quote dei lavori.

Limite di dimensione del documento Job

La dimensione del documento di lavoro è limitata dalla dimensione del payload MQTT. Se hai bisogno di un documento di lavoro di dimensioni superiori a 32 kB (kilobyte), 32.000 B (byte), crea e archivia il documento di lavoro in Amazon S3 e aggiungi un URL dell'oggetto Amazon S3 nel campo per l'API o utilizzando il`documentSource`. `CreateJob` AWS CLI Per la Console di gestione AWS, aggiungi l'URL di un oggetto Amazon S3 nella casella di testo URL Amazon S3 quando crei un lavoro.

- Console di gestione AWS creare documentazione sulle offerte di lavoro: crea e gestisci i lavori utilizzando il Console di gestione AWS
- AWS CLI creare documentazione sulle offerte di lavoro: creare e gestire i lavori utilizzando il AWS CLI

- `CreateJob`Documentazione API: CreateJob

Limiti di limitazione delle richieste di messaggi MQTT lato dispositivo

Se viene visualizzato il codice di errore 400`ThrottlingException`, il messaggio MQTT lato dispositivo non è riuscito a causa del raggiungimento del limite di richieste simultanee lato dispositivo. Consulta i limiti e le quote di AWS IoT Device Management lavoro per ulteriori informazioni sui limiti dell'acceleratore e se sono regolabili.

Errore di timeout della connessione

Un codice di errore 400 `RequestExpired` indica un errore di connessione dovuto a valori di latenza elevati o di timeout sul lato client bassi.

- Vedi Test della connettività con l'endpoint di dati del dispositivo per informazioni sul test della connessione tra il lato client e il lato server.

Comando API non valido

Conferma che sia stato immesso il comando API corretto per evitare che venga visualizzato un messaggio di errore che indica che il comando API non è valido. Consulta l'AWS IoT API Reference per un elenco completo di tutti i comandi AWS IoT API.

Errore di connessione lato servizio

Il codice di errore 503 `ServiceUnavailable` indica che l'errore ha avuto origine dal lato server.

- Vedi Dashboard AWS Health (tutti i AWS servizi) per lo stato attuale di tutti i AWS servizi.
- Vedi Dashboard AWS Health (personale Account AWS) per lo stato attuale dei tuoi dati personali Account AWS.

Risoluzione dei problemi di indicizzazione della flotta

Risoluzione dei problemi delle query di aggregazione per il servizio di indicizzazione del parco istanze

Se riscontri errori di mancata corrispondenza dei tipi, puoi utilizzare CloudWatch Logs per risolvere il problema. CloudWatch I registri devono essere abilitati prima che i registri vengano scritti dal servizio Fleet Indexing. Per ulteriori informazioni, consulta Monitora AWS IoT utilizzando i log CloudWatch.

Per eseguire query di aggregazione su campi non gestiti, devi specificare un campo definito nell'argomento `customFields` passato a `UpdateIndexingConfiguration` o update-indexing-configuration. Se il valore del campo non è coerente con il tipo di dati di campo configurato, questo valore viene ignorato quando si esegue una query di aggregazione.

Se un campo non può essere indicizzato a causa di un tipo non corrispondente, il servizio Fleet Indexing invia un log degli errori a Logs. CloudWatch Il log degli errori contiene il nome del campo, il valore che non può essere convertito e il nome dell'oggetto per il dispositivo. Di seguito è riportato un esempio di log degli errori.

```
{
  "timestamp": "2017-02-20 20:31:22.932",
  "logLevel": "ERROR",
  "traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
  "accountId": "000000000000",
  "status": "SucceededWithIssues",
  "eventType": "IndexingCustomFieldFailed",
  "thingName": "thing0",
  "failedCustomFields": [
    {
      "Name": "attributeName1",
      "Value": "apple",
      "ExpectedType": "String"
    },
    {
      "Name": "attributeName2",
      "Value": "2",
      "ExpectedType": "Boolean"
    }
  ]
}
```

Se un dispositivo è stato disconnesso per circa un'ora, il valore `timestamp` dello stato connettività potrebbe mancare. Per le sessioni persistenti, il valore potrebbe mancare dopo che un client è stato disconnesso più a lungo del periodo configurato time-to-live (TTL) per la sessione persistente. I dati sulo stato della connessione sono indicizzati solo per le connessioni in cui l'ID client dispone di un nome oggetto corrispondente. (L'ID client è il valore utilizzato per connettere un dispositivo.) AWS IoT Core

Risoluzione dei problemi di configurazione dell'indicizzazione del parco istanze

Impossibile eseguire il downgrade della configurazione di indicizzazione del parco istanze

Il downgrade della configurazione di indicizzazione del parco istanze non è supportato quando desideri rimuovere le origini dati associate a un parametro del parco istanze o a un gruppo dinamico.

Ad esempio, se la configurazione di indicizzazione include dati di registro, dati shadow e dati di connettività ed esiste un parametro del parco istanze con la query `thingName:TempSensor* AND shadow.desired.temperature>80`, l'aggiornamento della configurazione di indicizzazione per includere solo i dati del registro genera un errore.

La modifica dei campi personalizzati utilizzati dai parametri esistenti del parco istanze non è supportata.

Impossibile aggiornare la configurazione di indicizzazione a causa di parametri del parco istanze o gruppi dinamici incompatibili

Se non è possibile aggiornare la configurazione di indicizzazione a causa di parametri del parco istanze o gruppi dinamici incompatibili, elimina il parametri del parco istanze o i gruppi dinamici incompatibili prima di aggiornare la configurazione di indicizzazione.

Risoluzione dei problemi relativi all'indicizzazione della posizione e alle geoquery

Per risolvere gli errori di tipo non corrispondenti nell'indicizzazione della posizione e nelle geoquery, puoi abilitare i log. CloudWatch Per ulteriori informazioni su come monitorare l'utilizzo, segui la guida. AWS IoT CloudWatch step-by-step

Quando si indicizzano i dati sulla posizione utilizzando le geoquery, i campi di posizione specificati `geoLocations` devono corrispondere ai campi di posizione a cui si passa. `UpdateIndexingConfiguration` In caso di mancata corrispondenza, l'indicizzazione della flotta invia un errore di tipo non corrispondente a. CloudWatch Il log degli errori contiene il nome del campo, il valore che non può essere convertito e il nome dell'oggetto per il dispositivo.

Di seguito è riportato un esempio di log degli errori.

```
{
"timestamp": "2023-11-09 01:39:43.466",
    "logLevel": "ERROR",
    "traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "IndexingGeoLocationFieldFailed",
    "thingName": "thing0",
    "failedGeolocationFields": [
        {
"Name": "attributeName1",
            "Value": "apple",
            "ExpectedType": "Geopoint"
        }
    ],
    "reason": "failed to index the field because it could not be converted to one of
 the expected geoLocation formats."
}
```

Per ulteriori informazioni, consulta Indicizzazione dei dati sulla posizione.

Risoluzione dei problemi dei parametri del parco istanze

Impossibile visualizzare i punti dati in CloudWatch

Se riesci a creare una metrica del parco dati ma non riesci a visualizzarne i punti dati CloudWatch, è probabile che tu non abbia nulla che soddisfi i criteri della stringa di query.

Visualizza questo comando di esempio per creare un parametro del parco istanze:

```
aws iot create-fleet-metric --metric-name "example_FM" --query-string
 "thingName:TempSensor* AND attributes.temperature>80" --period 60 --aggregation-field
 "attributes.temperature" --aggregation-type name=Statistics,values=count
```

Se non si dispone di un oggetto che soddisfa i criteri della stringa di query `--query-string "thingName:TempSensor* AND attributes.temperature>80"`:

- Con`values=count`, sarai in grado di creare una metrica della flotta e ci saranno punti dati in cui mostrare. CloudWatch I punti dati del valore `count` è sempre 0.

- Con `values` other than`count`, sarai in grado di creare una metrica della flotta ma non visualizzerai la metrica della flotta CloudWatch e non ci saranno punti dati da mostrare. CloudWatch

AWS IoT Risoluzione dei problemi del Device Management Software Package Catalog

Questa è la sezione di risoluzione dei problemi per AWS IoT Device Management Software Package Catalog.

Messaggi di errore generali sulla risoluzione dei problemi

Questa sezione elenca gli errori più comuni riscontrati durante il ciclo di vita della versione del pacchetto software.

`HeadBucket` errors (Errori CloudTrail)

I seguenti messaggi di errore vengono visualizzati quando si chiama l'operazione `HeadBucket` API o il comando `head-bucket` CLI per convalidare il bucket Amazon S3 utilizzato per il caricamento di file durante una distribuzione di lavoro.

Per ulteriori informazioni sull'utilizzo di un bucket Amazon S3 per il caricamento di file durante l'implementazione di un processo, consulta. URL predefinito per il caricamento dei file

```
InvalidRoleException
    "Permission denied when attempting to use role %s to access bucket %s."
```

```
InvalidRequestException
    "Cross region S3 bucket is not supported for presigned url upload placeholder"
```

```
InvalidRequestException
    "S3 bucket in job document presigned url upload placeholder not found"
```

```
InvalidRequestException
    "Given S3 bucket name is invalid."
```

```
InvalidRequestException
    "Provided S3 bucket is not valid: %s. Error: %s"
```

Amazon S3 GetObject

Il seguente messaggio di errore si verifica quando viene fornito un argomento non valido, causando il fallimento dell'operazione dell'API Amazon `GetObject` S3.

```
InvalidRequestException
    "Provided argument for presigned url is invalid"
```

Supporto ID versione Amazon S3

Quando richiedi l'accesso a un bucket Amazon S3 utilizzando il controllo delle versioni, assicurati di includere il `versionId` tuo o il seguente errore potrebbe essere compilato.

Per ulteriori informazioni sui bucket Amazon S3 che utilizzano il controllo delle versioni, consulta Uso del controllo delle versioni nei bucket Amazon S3

```
InvalidRequestException
    "VersionId not found when attempting to access s3 url"
```

Segnaposto all'interno di un URL predefinito per il caricamento di file

I seguenti messaggi di errore vengono visualizzati quando si verificano problemi con un segnaposto all'interno di un URL predefinito utilizzato per caricare file su un bucket Amazon S3 di destinazione durante l'implementazione di un processo. Per ulteriori informazioni sull'utilizzo di un bucket Amazon S3 per il caricamento di file durante l'implementazione di un processo e cos'è un segnaposto locale, consulta. URL predefinito per il caricamento dei file

Il seguente messaggio di errore viene visualizzato quando il segnaposto locale non viene riconosciuto.

```
InvalidJobDocumentException
    "Undefined placeholder, ${...}, inside of presign url upload parameter"
```

Il seguente messaggio di errore viene visualizzato quando si tenta di utilizzare il segnaposto locale in un URL predefinito non destinato al caricamento di file.

```
InvalidJobDocumentException
    "Local placeholder, ${...}, is only valid inside of presign url upload"
```

URL Amazon S3 annidato in modo errato

Il seguente messaggio di errore viene visualizzato quando l'URL di Amazon S3 è erroneamente annidato all'interno di un altro segnaposto.

```
InvalidJobDocumentException
    "${aws:%s[...]} should not be the second layer pattern."
```

Versione del pacchetto Artifact Nesting

Il seguente messaggio di errore viene visualizzato quando l'URL prefirmato dell'artefatto della versione del pacchetto viene annidato erroneamente all'interno di un altro segnaposto.

```
InvalidJobDocumentException
    "${aws:iot:package:[...]:artifact:s3-presigned-url} cannot be nested inside another
 placeholder."
```

Artifact della versione del pacchetto mancante

Il seguente messaggio di errore viene visualizzato quando l'elemento della versione del pacchetto di riferimento non viene trovato.

```
InvalidJobDocumentException
    "Package %s version %s does not have an associated artifact to generate an S3
 presigned url."
```

Segnaposto per pacchetti software e versioni di pacchetti

Il seguente messaggio di errore viene visualizzato quando il segnaposto del documento di lavoro per il pacchetto software e la versione del pacchetto non riesce a risolvere i valori validi desiderati per la distribuzione del lavoro a causa di più pacchetti software e versioni del pacchetto a cui si fa riferimento nel *`destinationPackageVersions`* parametro o nella scheda Version ARN nella pagina dei dettagli della versione del pacchetto.

```
InvalidJobDocumentException
    "Cannot resolve empty package name and version name given multiple elements in
 destination package versions."
```

Utilizzo del pacchetto software vuoto e della versione del pacchetto

Il seguente messaggio di errore viene visualizzato quando si tenta di utilizzare un pacchetto o una versione del pacchetto vuoto senza l'altro in un documento di lavoro.

```
InvalidJobDocumentException
    "Empty package name and version name have to be used in pair."
```

Utilizzo nullo nel documento Job

Il seguente messaggio di errore viene visualizzato quando si tenta di specificare `$null` una versione del pacchetto nel documento di lavoro. `$null`può essere utilizzato solo all'interno del `destinationPackageVersions` parametro quando si utilizza l'operazione `CreateJob` API.

```
InvalidJobDocumentException
    "$null is not allowed to be referenced as a package version in job documents."
```

Tutti gli attributi in una versione del Package

Il seguente messaggio di errore viene visualizzato quando si tenta di utilizzare tutti gli attributi in una versione del pacchetto e di circondarla con testo o segnaposto aggiuntivi.

Per ulteriori informazioni sull'utilizzo di tutti gli attributi in una versione di pacchetto software, vedere Parametri di sostituzione per i lavori AWS IoT

```
InvalidJobDocumentException
    "The package version attribute placeholder for all attributes has to be a
 json value by itself and not appended with other strings or nested with other
 placeholders."
```

Limite di segnaposto locale nell'URL predefinito per il caricamento di file

Il seguente messaggio di errore viene visualizzato quando si supera il limite per il numero di segnaposti locali utilizzati in un URL predefinito per il caricamento di file durante la distribuzione di un lavoro.

Per ulteriori informazioni sull'utilizzo di un URL predefinito per il caricamento di file durante la distribuzione di un lavoro, vedere URL predefinito per il caricamento dei file

```
InvalidJobDocumentException
    "The occurrence of local placeholder %s within S3 presigned url upload placeholder
 exceeds limit of %d."
```

Segnaposti locali in un bucket Amazon S3

Il seguente messaggio di errore viene visualizzato quando si tenta di inserire un URL segnaposto locale nel nome del bucket Amazon S3 per un segnaposto URL predefinito utilizzato per il caricamento di file durante una distribuzione di lavoro.

Per ulteriori informazioni sull'utilizzo di un URL predefinito per il caricamento di file durante l'implementazione di un processo, consulta URL predefinito per il caricamento dei file

```
InvalidJobDocumentException
    "S3 bucket name in presigned url upload is not allowed to contain any placeholders"
```

Parentesi di apertura e chiusura

Il seguente messaggio di errore viene visualizzato quando aggiungete un parametro o un segnaposto a un documento di lavoro senza una parentesi graffa di chiusura «}».

```
InvalidJobDocumentException
    "One or more parameters or placeholders are not terminated."
```

Ruolo IAM con Amazon S3 Presigned URL

Il seguente messaggio di errore viene visualizzato quando si tenta di utilizzare un URL prefirmato Amazon S3 in un documento di lavoro senza un ruolo IAM.

Per ulteriori informazioni su Amazon S3 presigned URLs, consulta Working with presigned. URLs

```
InvalidRequestException
    "presignedUrlConfig role ARN is required to generate an S3 presigned url in job
 document."
```

Ruolo IAM con l'URL predefinito di Amazon S3 per la versione del pacchetto Artifact

Il seguente messaggio di errore viene visualizzato quando si tenta di utilizzare un URL prefirmato Amazon S3 che rappresenta un elemento della versione del pacchetto in un documento di lavoro senza un ruolo IAM.

```
InvalidRequestException
    "presignedUrlConfig role ARN is required to generate an S3 presigned url in job
 document for package %s version %s artifact."
```

Messaggi di errore relativi alla distinta base del software

Questa sezione elenca gli errori più comuni associati a una distinta base del software (SBOM) collegata a una versione del pacchetto.

Convalida dell'input per la richiesta di associazione SBOM

Il seguente messaggio di errore viene visualizzato quando si utilizza l'operazione `AssociateSbomWithPackageVersion` API e il `s3Location` parametro è nullo.

```
InvalidRequestException "Associate request needs to include SBOM reference"
```

Per ulteriori informazioni sul funzionamento dell'`AssociateSbomWithPackageVersion`API, vedere AssociateSbomWithPackageVersion.

Errori di convalida SBOM

Questa sezione elenca gli errori più comuni riscontrati durante la convalida iniziale della distinta base del software (SBOM) quando associata a una versione del pacchetto software.

Il seguente messaggio di errore viene visualizzato quando si utilizza l'operazione `AssociateSbomWithPackageVersion` API e `bucket` nel parametro è null. `s3Location`

```
InvalidRequestException "S3 bucket name for SBOM cannot be null"
```

Il seguente messaggio di errore viene visualizzato quando la stringa `bucket` inserita nel `s3Location` parametro per l'operazione `AssociateSbomWithPackageVersion` API è troppo lunga.

```
InvalidRequestException "S3 bucket name for SBOM is illegal. String length exceeds
 limit"
```

Il seguente messaggio di errore viene visualizzato quando il `key` parametro è nullo.

```
InvalidRequestException "S3 key name for SBOM cannot be null"
```

Il seguente messaggio di errore viene visualizzato quando la stringa `key` inserita nel `s3Location` parametro per l'operazione `AssociateSbomWithPackageVersion` API è troppo lunga.

```
InvalidRequestException "S3 key name for SBOM is illegal. String length exceeds limit"
```

Il seguente messaggio di errore viene visualizzato quando la stringa `version` inserita nel `s3Location` parametro per l'operazione `AssociateSbomWithPackageVersion` API è nulla.

```
InvalidRequestException "S3 object version for SBOM cannot be null"
```

Il seguente messaggio di errore viene visualizzato quando la stringa `version` inserita nel `s3Location` parametro per l'operazione `AssociateSbomWithPackageVersion` API è troppo lunga.

```
InvalidRequestException "S3 object version for SBOM is illegal. String length exceeds
 limit"
```

Il seguente messaggio di errore viene visualizzato quando la dimensione del file di archivio zip SBOM archiviato nel bucket Amazon S3 è troppo grande.

```
InvalidRequestException "S3 object file size exceeds limit"
```

Il seguente messaggio di errore viene visualizzato quando si utilizza l'operazione `AssociateSbomWithPackageVersion` API e il numero attuale di convalide SBOM in corso è già al limite massimo.

```
LimitExceededException "Too many ongoing SBOM validation workflows. Please wait and
 retry"
```

Problemi di accesso con il file SBOM nel bucket Amazon S3

Il seguente messaggio di errore viene visualizzato quando un'altra entità non riesce ad accedere al bucket Amazon S3 perché il bucket Amazon S3 non esiste o non sono state concesse le autorizzazioni appropriate per accedere al bucket Amazon S3.

Per ulteriori informazioni sulla politica di autorizzazione richiesta per l'accesso al bucket Amazon S3, consulta. Software, distinta dei materiali, archiviazione

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the
 bucket exists and S3 permission is granted."
```

Il seguente messaggio di errore viene visualizzato quando un'altra entità non riesce ad accedere al file di archivio zip SBOM nel `key` parametro perché il bucket Amazon S3 non esiste o non sono state concesse le autorizzazioni appropriate per accedere ai contenuti archiviati nel bucket Amazon S3.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the key
 exists and S3 permission is granted."
```

Il seguente messaggio di errore viene visualizzato quando un'altra entità non riesce ad accedere al bucket Amazon S3 perché il bucket, la chiave e l'ID di versione non esistono o non sono state concesse le autorizzazioni appropriate per accedere al bucket Amazon S3. Inoltre, questo messaggio di errore può apparire se le autorizzazioni concesse non sono sufficienti per accedere al file di archivio zip SBOM nel bucket Amazon S3.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the
 bucket/key/version exists and S3 permission is granted."
```

Il seguente messaggio di errore viene visualizzato quando un'altra entità non riesce ad accedere al bucket Amazon S3 perché il bucket si trova in un'altra regione.

```
InvalidRequestException "Cross-region S3 bucket for %s is not supported."
```

Il seguente messaggio di errore viene visualizzato quando un'altra entità non riesce ad accedere al bucket Amazon S3 a causa dell'errata ortografia `version` dei parametri `bucketkey`, o durante l'utilizzo dell'operazione API. `AssociateSbomWithPackageVersion`

```
InvalidRequestException "Please make sure SBOM S3 bucket name/key length/version is
 valid"
```

AWS IoT Guida alla risoluzione di Device Advisor

Aiutaci a migliorare questo argomento

Facci sapere che cosa contribuirebbe a migliorarlo

Ambito generale

D: È possibile eseguire più suite di test in contemporanea?

R: Sì. Ora Device Advisor supporta l'esecuzione di più suite di test su dispositivi diversi utilizzando un endpoint a livello di dispositivo. Se utilizzi l'endpoint a livello di account, puoi eseguire una suite

alla volta perché è disponibile un endpoint Device Advisor per account. Per ulteriori informazioni, consulta la sezione Configurazione del dispositivo.

D: Ho visto dal mio dispositivo che la connessione TLS è stata negata da Device Advisor. È previsto?

R: Sì. Device Advisor nega la connessione TLS prima e dopo ogni esecuzione di un test. Consigliamo agli utenti di implementare il meccanismo di nuovo tentativo del dispositivo per un'esperienza di test completamente automatizzata con Device Advisor. Se esegui una suite di test con più di un test case, ad esempio TLS connect, MQTT connect e MQTT publish, ti consigliamo di disporre di un meccanismo progettato per il tuo dispositivo. Il meccanismo può provare a connettersi al nostro endpoint di test ogni 5 secondi per un minuto o due. In questo modo, puoi eseguire più test case in sequenza in modo automatico.

D: È possibile ottenere uno storico di tutte le chiamate alle API effettuate sull'account per analizzare la sicurezza e per la risoluzione dei problemi operativi?

R: Sì. Per ricevere una cronologia delle chiamate API di Device Advisor effettuate sul tuo account, devi semplicemente attivarla CloudTrail nella Console di AWS IoT gestione e filtrare la fonte dell'evento desiderata`iotdeviceadvisor.amazonaws.com`.

D: Come posso visualizzare gli accessi di Device Advisor? CloudWatch

R: I log generati durante l'esecuzione di una suite di test vengono caricati CloudWatch se aggiungi la policy richiesta (ad esempio, CloudWatchFullAccess) al tuo ruolo di servizio (vediConfigurazione). Se nella suite di test è presente almeno un test case, viene creato un gruppo di log "aws/iot/deviceadvisor/$testSuiteId" con due flussi di log. Uno stream è «$testRunId" e include i registri delle azioni intraprese prima e dopo l'esecuzione dei casi di test nella suite di test, come le fasi di configurazione e pulizia. L'altro flusso di log è «$ suiteRunId _$»testRunId, che è specifico per l'esecuzione di una suite di test. Gli eventi inviati dai dispositivi AWS IoT Core verranno registrati in questo flusso di log.

D: Qual è lo scopo del ruolo di autorizzazione del dispositivo?

R: Device Advisor si colloca tra il dispositivo di test e la simulazione AWS IoT Core di scenari di test. Accetta connessioni e messaggi dai tuoi dispositivi di test e li inoltra a AWS IoT Core assumendo il ruolo di autorizzazione del dispositivo e avviando una connessione per conto dell'utente. È importante assicurarsi che le autorizzazioni del ruolo del dispositivo siano le stesse del certificato utilizzato per l'esecuzione dei test. AWS IoT le politiche relative ai certificati non vengono applicate quando Device Advisor avvia una connessione per AWS IoT Core conto dell'utente utilizzando il ruolo di autorizzazione del dispositivo. Tuttavia, vengono applicate le autorizzazioni del ruolo di autorizzazione del dispositivo impostato.

D: In quali regioni è supportato Device Advisor?

R: Device Advisor è supportato nelle regioni us-east-1, us-west-2, ap-northeast-1 e eu-west-1.

D: Perché vedo risultati incoerenti?

R: Una delle cause principali di risultati incoerenti è l'impostazione di un test `EXECUTION_TIMEOUT` a un valore troppo basso. Per ulteriori informazioni sui valori consigliati e di default `EXECUTION_TIMEOUT`, consulta Test case di Device Advisor.

D: Quale protocollo MQTT è supportato da Device Advisor?

R: Device Advisor supporta MQTT versione 3.1.1 con certificati client X509.

D: Che cosa succede se il test case non è riuscito con un messaggio di timeout di esecuzione, anche se ho provato a collegare il dispositivo all'endpoint di test?

R: Convalida tutti i passaggi indicati in Creazione di un ruolo IAM da utilizzare come ruolo del dispositivo. Se il test continua a fallire, è possibile che il dispositivo non invii l'estensione Server Name Indication (SNI) corretta, necessaria per il funzionamento di Device Advisor. Il valore SNI corretto è l'indirizzo dell'endpoint restituito quando si segue la sezione Configura il dispositivo. AWS IoT richiede inoltre che i dispositivi inviino l'estensione SNI (Server Name Indication) al protocollo Transport Layer Security (TLS). Per ulteriori informazioni, vedere Transport security in. AWS IoT

D: La mia connessione MQTT si interrompe con un errore "libaws-c-mqtt: AWS_ERROR _MQTT_UNEXPECTED_HANGUP» (oppure) la connessione MQTT del mio dispositivo viene disconnessa automaticamente dall'endpoint Device Advisor. Come si può risolvere questo problema?

R: Questo particolare codice di errore e le disconnessioni impreviste possono essere causati da numerosi fattori, ma molto probabilmente sono correlati al ruolo assegnato al dispositivo. I seguenti punti di controllo (in ordine di priorità) risolveranno il problema.

- Il ruolo assegnato al dispositivo deve disporre delle autorizzazioni IAM minime richieste per eseguire i test. Device Advisor utilizzerà il ruolo assegnato al dispositivo per eseguire operazioni AWS IoT con protocollo MQTT per conto del dispositivo di test. Se le autorizzazioni richieste non sono presenti, allora verrà visualizzato l'errore `AWS_ERROR_MQTT_UNEXPECTED_HANGUP` o si verificheranno disconnessioni impreviste mentre il dispositivo prova a connettersi all'endpoint Device Advisor. Ad esempio, se avete scelto di eseguire il test case MQTT Publish, entrambe le azioni Connect e Publish devono essere incluse nel ruolo con il corrispondente ClientId e l'argomento (potete fornire più valori utilizzando virgole per separare i valori e potete fornire valori di prefisso utilizzando un carattere jolly (*). Ad esempio, per fornire

l'autorizzazione per la pubblicazione su qualsiasi argomento che inizia con `TestTopic`, puoi fornire "`TestTopic*`" come valore della risorsa. Ecco alcuni esempi di policy.

- Mancata corrispondenza tra i valori definiti nel ruolo del dispositivo per i tipi di risorse e i valori effettivi utilizzati nel codice. Ad esempio: una mancata corrispondenza tra la ClientId definizione del ruolo e quella effettivamente ClientId utilizzata nel codice del dispositivo. Valori come ClientId Topic e TopicFilter devono essere identici nel ruolo e nel codice del dispositivo.
- Il certificato del dispositivo collegato al dispositivo deve essere attivo e disporre di una policy collegata con le autorizzazioni delle operazioni necessarie per le risorse. Tieni presente che la policy relativa ai certificati dei dispositivi concede o nega l'accesso alle AWS IoT risorse e alle operazioni del piano AWS IoT Core dati. Device Advisor richiede che tu abbia un certificato del dispositivo attivo collegato al dispositivo che conceda le autorizzazioni delle operazioni utilizzate durante un test case.

AWS IoT errori

Aiutaci a migliorare questo argomento

Facci sapere che cosa contribuirebbe a migliorarlo

Questa sezione elenca i codici di errore inviati da AWS IoT.

Codici di errore del broker di messaggi

Codice di errore	Descrizione dell'errore.
400	Richiesta non valida.
401	Non autorizzato.
403	Accesso negato.
426	Aggiornamento richiesto.
503	Servizio non disponibile.

Codici di errore relativi a identità e sicurezza

Codice di errore	Descrizione dell'errore.
401	Non autorizzato.

Codici di errore relativi a Device Shadow

Codice di errore	Descrizione dell'errore.
400	Richiesta non valida.
401	Non autorizzato.
403	Accesso negato.
404	Non trovato.
409	Conflitto.
413	Richiesta di dimensioni troppo grandi.
422	Impossibile elaborare la richiesta.
429	Troppe richieste.
500	Errore interno.
503	Servizio non disponibile.

AWS IoT Client per dispositivi SDKs SDKs, dispositivi mobili e AWS IoT dispositivi mobili

Questa pagina riassume i AWS IoT dispositivi SDKs, le librerie open source, le guide per gli sviluppatori, le app di esempio e le guide al porting per aiutarti a creare soluzioni IoT innovative con piattaforme AWS IoT hardware a tua scelta.

Questi SDKs sono per l'uso sul tuo dispositivo IoT. Se stai sviluppando un'app IoT per l'utilizzo su un dispositivo mobile, consulta AWS Mobile SDKs. Se stai sviluppando un'app IoT o un programma lato server, consulta AWS SDKs.

AWS IoT Device SDKs

Il AWS IoT dispositivo SDKs include librerie open source, guide per sviluppatori con esempi e guide al porting in modo da poter creare prodotti o soluzioni IoT innovativi sulle piattaforme hardware di tua scelta.

> **Note**
>
> Il AWS IoT dispositivo SDKs ha rilasciato un client MQTT 5. Il AWS IoT dispositivo SDKs non supporta l'utilizzo di TLS 1.3 su macOS.

Questi SDKs consentono di connettere i dispositivi IoT all' AWS IoT utilizzo dei protocolli MQTT e WSS.

C++

AWS IoT SDK per dispositivi C++

Il AWS IoT C++ Device SDK consente agli sviluppatori di creare applicazioni connesse utilizzando e. AWS AWS IoT APIs In particolare, questo SDK è stato progettato per i dispositivi che non hanno vincoli di risorse e che richiedono caratteristiche avanzate come l'accodamento dei messaggi, il supporto per il multithreading e le più recenti caratteristiche di linguaggio. Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT Device SDK C++ v2 attivo GitHub
- AWS IoT Readme del dispositivo SDK C++ v2

- AWS IoT Esempi di Device SDK C++ v2
- AWS IoT Documentazione sull'API Device SDK C++ v2

Python

AWS IoT Device SDK per Python

Il AWS IoT Device SDK for Python consente agli sviluppatori di scrivere script Python per utilizzare i propri dispositivi per accedere alla piattaforma tramite MQTT o MQTT tramite AWS IoT il protocollo. WebSocket Collegando i propri dispositivi a AWS IoT, gli utenti possono lavorare in sicurezza con il broker di messaggi, le regole e le ombre fornite da AWS IoT e con altri AWS servizi come AWS Lambda Kinesis e Amazon S3 e molti altri.

- AWS IoT Device SDK per Python v2 attivo GitHub
- AWS IoT Device SDK per Python v2 Readme
- AWS IoT Device SDK per esempi in Python v2
- AWS IoT Documentazione dell'API Device SDK per Python v2

JavaScript

AWS IoT Device SDK per JavaScript

Il aws-iot-device-sdk pacchetto.js consente agli sviluppatori di scrivere JavaScript applicazioni che accedono AWS IoT tramite MQTT o MQTT tramite il protocollo. WebSocket Questo SDK può essere usato nelle applicazioni di tipo browser e negli ambienti Node.js. Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT Device SDK per v2 attivo JavaScript GitHub
- AWS IoT SDK del dispositivo per Readme v2 JavaScript
- AWS IoT Device SDK per esempi per la versione 2 JavaScript
- AWS IoT Documentazione sull'API Device SDK per la versione 2 JavaScript

Java

AWS IoT SDK per dispositivi per Java

Il AWS IoT Device SDK for Java consente agli sviluppatori Java di accedere alla piattaforma tramite MQTT o MQTT tramite AWS IoT il protocollo. WebSocket L'SDK è sviluppato con il

supporto delle copie shadow. Puoi accedere alle copie shadow tramite i metodi HTTP, tra cui GET, UPDATE e DELETE. L'SDK supporta anche un modello di accesso semplificato alle copie shadow, che permette agli sviluppatori di scambiare i dati con le copie shadow semplicemente usando i metodi getter e setter, senza dover serializzare o deserializzare documenti JSON.

> **Note**
>
> Il AWS IoT Device SDK for Java v2 ora supporta lo sviluppo Android. Per ulteriori informazioni, consulta AWS IoT Device SDK for Android.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT SDK del dispositivo per Java v2 attivo GitHub
- AWS IoT SDK del dispositivo per Java v2 Readme
- AWS IoT Esempi di Device SDK for Java v2
- AWS IoT Documentazione sull'API Device SDK for Java v2

Swift

AWS IoT Device SDK per Swift

Il AWS IoT Device SDK for Swift consente agli sviluppatori Swift di creare AWS IoT applicazioni per piattaforme Linux e Apple macOS, iOS e tvOS utilizzando il protocollo MQTT 5.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT Device SDK per Swift attivo GitHub
- AWS IoT SDK del dispositivo per Swift Readme
- AWS IoT Device SDK per esempi per Swift

AWS IoT SDK per dispositivi per Embedded C

> **Note**
>
> Questo SDK è destinato all'uso da parte di sviluppatori di software integrati esperti.

Il SDK per dispositivi AWS IoT per Embedded C (C-SDK) è una raccolta di file sorgente C con licenza open source MIT che possono essere utilizzati in applicazioni integrate per connettere in modo sicuro i dispositivi IoT. AWS IoT Core Include un client MQTT, JSON Parser e AWS IoT Device Shadow, AWS IoT Jobs, AWS IoT Fleet Provisioning e librerie. AWS IoT Device Defender Questo SKD viene distribuito come codice sorgente e può essere integrato nel firmware del cliente con il codice dell'applicazione, altre librerie ed eventualmente un sistema operativo a scelta.

SDK per dispositivi AWS IoT per Embedded C È generalmente destinato a dispositivi con risorse limitate che richiedono un runtime ottimizzato in linguaggio C. È possibile utilizzare l'SDK su qualsiasi sistema operativo e ospitarlo su qualsiasi tipo di processore (ad esempio, MCUs e). MPUs

Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT Device SDK per Embedded C su GitHub
- AWS IoT SDK del dispositivo per il file Readme C incorporato
- AWS IoT SDK del dispositivo per esempi C incorporati

AWS Mobile SDKs

The AWS Mobile SDKs fornisce agli sviluppatori di app mobili un supporto specifico per la piattaforma APIs dei servizi AWS IoT Core , la comunicazione dei dispositivi IoT tramite MQTT e altri servizi. APIs AWS

Android

AWS Mobile SDK per Android

AWS Mobile SDK per Android Contiene una libreria, esempi e documentazione per gli sviluppatori con cui creare applicazioni mobili connesse utilizzando. AWS Questo SDK include anche il supporto per le comunicazioni con i dispositivi MQTT e la chiamata ai APIs servizi. AWS IoT Core Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS Mobile SDK per Android su GitHub
- AWS Mobile SDK per Android Readme
- AWS Mobile SDK per Android Esempi
- AWS Mobile SDK per Android Documentazione di riferimento dell'API
- AWSIoTClient Documentazione di riferimento della classe

iOS

AWS Mobile SDK for iOS

AWS Mobile SDK for iOS È un kit di sviluppo software open source, distribuito con una licenza Apache Open Source. AWS Mobile SDK for iOS Fornisce una libreria, esempi di codice e documentazione per aiutare gli sviluppatori a creare applicazioni mobili connesse utilizzando. AWS Questo SDK include anche il supporto per le comunicazioni con i dispositivi MQTT e la chiamata ai APIs servizi. AWS IoT Core Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS Mobile SDK for iOS su GitHub
- AWS Mobile SDK for iOS Readme
- AWS Mobile SDK for iOS Esempi
- AWSIoDocumenti di riferimento della classe T in AWS Mobile SDK for iOS

AWS IoT Client del dispositivo

Il AWS IoT Device Client fornisce il codice per aiutare il dispositivo a connettersi AWS IoT, eseguire attività di approvvigionamento del parco veicoli, supportare le politiche di sicurezza dei dispositivi, connettersi tramite tunneling sicuro ed elaborare i processi sul dispositivo. È possibile installare questo software sul dispositivo per gestire queste attività di routine del dispositivo in modo da concentrarsi sulla soluzione specifica.

> **Note**
>
> Il AWS IoT Device Client funziona con dispositivi IoT basati su microprocessore con processori x86_64 o ARM e sistemi operativi Linux comuni.

C++

AWS IoT Device Client

Per ulteriori informazioni sul AWS IoT Device Client in C++, vedere quanto segue:

- AWS IoT Device Client nel codice sorgente C++ su GitHub
- AWS IoT Device Client nel file Readme in C++

Versioni precedenti AWS IoT del dispositivo SDKs

Si tratta di versioni precedenti di AWS IoT Device SDKs che sono state sostituite dalle versioni più recenti elencate sopra. Queste SDKs ricevono solo aggiornamenti di manutenzione e sicurezza. Non verranno aggiornati per includere nuove caratteristiche e non devono essere utilizzati su nuovi progetti.

- AWS IoT SDK per dispositivi C++ attivo GitHub
- AWS IoT Leggimi SDK per dispositivi C++
- AWS IoT Device SDK per Python v1 attivo GitHub
- AWS IoT Device SDK per Python v1 Readme
- AWS IoT SDK del dispositivo per Java attivo GitHub
- AWS IoT Device SDK for Java Readme
- AWS IoT Device SDK per attivare JavaScript GitHub
- AWS IoT SDK del dispositivo per Readme JavaScript
- Arduino Yún SDK attivo GitHub
- File Readme dell'SDK Arduino Yún

Esempi di codice per AWS IoT l'utilizzo AWS SDKs

I seguenti esempi di codice mostrano come utilizzarlo AWS IoT con un kit di sviluppo AWS software (SDK).

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi di codice

- Esempi di base per l' AWS IoT utilizzo AWS SDKs
  - Ciao AWS IoT
  - Scopri le nozioni di base di AWS IoT con un SDK AWS
  - Azioni per l'utilizzo AWS IoT AWS SDKs
    - Utilizzo AttachThingPrincipal con un AWS SDK o una CLI
    - Utilizzo CreateKeysAndCertificate con un AWS SDK o una CLI
    - Utilizzo CreateThing con un AWS SDK o una CLI
    - Utilizzo CreateTopicRule con un AWS SDK o una CLI
    - Utilizzo DeleteCertificate con un AWS SDK o una CLI
    - Utilizzo DeleteThing con un AWS SDK o una CLI
    - Utilizzo DeleteTopicRule con un AWS SDK o una CLI
    - Utilizzo DescribeEndpoint con un AWS SDK o una CLI
    - Utilizzo DescribeThing con un AWS SDK o una CLI
    - Utilizzo DetachThingPrincipal con un AWS SDK o una CLI
    - Utilizzo ListCertificates con un AWS SDK o una CLI
    - Utilizzo ListThings con un AWS SDK o una CLI
    - Utilizzo SearchIndex con un AWS SDK o una CLI

- Utilizzo UpdateIndexingConfiguration con un AWS SDK o una CLI
- Utilizzo UpdateThing con un AWS SDK o una CLI

Esempi di base per l' AWS IoT utilizzo AWS SDKs

I seguenti esempi di codice mostrano come utilizzare le nozioni di base di AWS IoT with. AWS SDKs

Esempi

- Ciao AWS IoT
- Scopri le nozioni di base di AWS IoT con un SDK AWS
- Azioni per l'utilizzo AWS IoT AWS SDKs
  - Utilizzo AttachThingPrincipal con un AWS SDK o una CLI
  - Utilizzo CreateKeysAndCertificate con un AWS SDK o una CLI
  - Utilizzo CreateThing con un AWS SDK o una CLI
  - Utilizzo CreateTopicRule con un AWS SDK o una CLI
  - Utilizzo DeleteCertificate con un AWS SDK o una CLI
  - Utilizzo DeleteThing con un AWS SDK o una CLI
  - Utilizzo DeleteTopicRule con un AWS SDK o una CLI
  - Utilizzo DescribeEndpoint con un AWS SDK o una CLI
  - Utilizzo DescribeThing con un AWS SDK o una CLI
  - Utilizzo DetachThingPrincipal con un AWS SDK o una CLI
  - Utilizzo ListCertificates con un AWS SDK o una CLI
  - Utilizzo ListThings con un AWS SDK o una CLI
  - Utilizzo SearchIndex con un AWS SDK o una CLI
  - Utilizzo UpdateIndexingConfiguration con un AWS SDK o una CLI
  - Utilizzo UpdateThing con un AWS SDK o una CLI

Ciao AWS IoT

L’esempio di codice seguente mostra come iniziare a utilizzare AWS IoT.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/// <summary>
/// Hello AWS IoT example.
/// </summary>
public class HelloIoT
{
    /// <summary>
    /// Main method to run the Hello IoT example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var iotClient = new AmazonIoTClient();

        try
        {
            Console.WriteLine("Hello AWS IoT! Let's list your IoT Things:");
            Console.WriteLine(new string('-', 80));

            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await iotClient.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
 paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
```

```
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await iotClient.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            if (things is { Count: > 0 })
            {
                Console.WriteLine($"Found {things.Count} IoT Things:");
                foreach (var thing in things)
                {
                    Console.WriteLine($"- Thing Name: {thing.ThingName}");
                    Console.WriteLine($"  Thing ARN: {thing.ThingArn}");
                    Console.WriteLine($"  Thing Type: {thing.ThingTypeName ?? "No
type specified"}");
                    Console.WriteLine($"  Version: {thing.Version}");

                    if (thing.Attributes?.Count > 0)
                    {
                        Console.WriteLine("  Attributes:");
                        foreach (var attr in thing.Attributes)
                        {
                            Console.WriteLine($"    {attr.Key}: {attr.Value}");
                        }
                    }
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No IoT Things found in your account.");
                Console.WriteLine("You can create IoT Things using the IoT Basics
scenario example.");
            }

            Console.WriteLine("Hello IoT completed successfully.");
```

```
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            Console.WriteLine($"Request throttled, please try again later:
 {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list Things. Here's why: {ex.Message}");
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta listThings nella documentazione di riferimento dell'API AWS SDK per .NET .

C++

SDK per C++

Codice per il CMake file CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
```

```
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Codice per il file origine hello_iot.cpp.

```
#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
 *  A "Hello IoT" starter application which initializes an AWS IoT client and
 *  lists the AWS IoT topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_iot'
 *
 */

int main(int argc, char **argv) {
```

```
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
iotClient.ListThings(
                    listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
                          << listThingsOutcome.GetError().GetMessage() <<
std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allThings.size() << " thing(s) found." << std::endl;
        for (auto const &thing: allThings) {
            std::cout << thing.GetThingName() << std::endl;
        }
    }
```

```
    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Per informazioni dettagliate sull'API, consulta listThings nella documentazione di riferimento dell'API AWS SDK per C++ .

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import software.amazon.awssdk.services.iot.paginators.ListThingsIterable;

import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
 Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        listAllThings(iotClient);
    }

    public static void listAllThings(IotClient iotClient) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                System.out.println("Thing name: " + attribute.thingName());
                System.out.println("Thing ARN: " + attribute.thingArn());
            });
    }
}
```

- Per informazioni dettagliate sull'API, consulta listThings nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

Kotlin

SDK per Kotlin

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
```

```
            maxResults = 10
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta listThings nella documentazione di riferimento dell'API AWS SDK per Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
def hello_iot():
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT client and list
    up to 10 things in your AWS IoT account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    try:
        iot_client = boto3.client("iot")
        response = iot_client.list_things(maxResults=10)
        things = response.get("things", [])
```

```
        print("Hello, AWS IoT! Here are your things:")
        if things:
            for i, thing in enumerate(things, 1):
                print(f"{i}. {thing['thingName']}")
        else:
            print("No things found in your AWS IoT account.")
    except ClientError as e:
        if e.response["Error"]["Code"] == "UnauthorizedException":
            print("You don't have permission to access AWS IoT.")
        else:
            print(f"Couldn't access AWS IoT. Error: {e}")
    except NoCredentialsError:
        print("No AWS credentials found. Please configure your credentials.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

- Per i dettagli sull'API, consulta ListThings in AWS SDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scopri le nozioni di base di AWS IoT con un SDK AWS

Gli esempi di codice seguenti mostrano come:

- Crea una AWS IoT cosa.
- Generare un certificato del dispositivo.
- Aggiornare un AWS IoT oggetto con attributi.
- Restituire un endpoint univoco.
- Elenca i tuoi AWS IoT certificati.
- Aggiorna un' AWS IoT ombra.
- Scrivere le informazioni sullo stato.
- Crea una regola.
- Elencare le regole.

- Cercare gli oggetti utilizzando il nome dell'oggetto.
- Eliminare un AWS IoT oggetto.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo che dimostri le AWS IoT funzionalità.

```
/// <summary>
/// Scenario class for AWS IoT basics.
/// </summary>
public class IoTBasics
{
    public static bool IsInteractive = true;
    public static IoTWrapper? Wrapper = null;
    public static IAmazonCloudFormation? CloudFormationClient = null;
    public static ILogger<IoTBasics> logger = null!;
    private static IoTWrapper _iotWrapper = null!;
    private static IAmazonCloudFormation _amazonCloudFormation = null!;
    private static ILogger<IoTBasics> _logger = null!;

    private static string _stackName = "IoTBasicsStack";
    private static string _stackResourcePath = "../../../../../../scenarios/
basics/iot/iot_usecase/resources/cfn_template.yaml";

    /// <summary>
    /// Main method for the IoT Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIoT>(new AWSOptions() { Region =
 RegionEndpoint.USEast1 })
                    .AddAWSService<IAmazonCloudFormation>()
                        .AddTransient<IoTWrapper>()
                        .AddLogging(builder => builder.AddConsole())
                        .AddSingleton<IAmazonIotData>(sp =>
                        {
                            var iotService = sp.GetRequiredService<IAmazonIoT>();
                            var request = new DescribeEndpointRequest
                            {
                                EndpointType = "iot:Data-ATS"
                            };
                            var response =
 iotService.DescribeEndpointAsync(request).Result;
                            return new AmazonIotDataClient($"https://
{response.EndpointAddress}/");
                        })
            )
            .Build();

        logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<IoTBasics>();

        Wrapper = host.Services.GetRequiredService<IoTWrapper>();
        CloudFormationClient =
 host.Services.GetRequiredService<IAmazonCloudFormation>();

        // Set the private fields for backwards compatibility
        _logger = logger;
        _iotWrapper = Wrapper;
        _amazonCloudFormation = CloudFormationClient;

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the AWS IoT example scenario.");
        Console.WriteLine("This example program demonstrates various interactions
 with the AWS Internet of Things (IoT) Core service.");
        Console.WriteLine();
        if (IsInteractive)
        {
            Console.WriteLine("Press Enter to continue...");
            Console.ReadLine();
        }
        Console.WriteLine(new string('-', 80));
```

```
        try
        {
            await RunScenarioAsync();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem running the scenario.");
            Console.WriteLine($"\nAn error occurred: {ex.Message}");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The AWS IoT scenario has successfully completed.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Run the IoT Basics scenario.
    /// </summary>
    /// <returns>A Task object.</returns>
    public static async Task RunScenarioAsync()
    {
        // Use static properties if available, otherwise use private fields
        var iotWrapper = Wrapper ?? _iotWrapper;
        var cloudFormationClient = CloudFormationClient ?? _amazonCloudFormation;
        var scenarioLogger = logger ?? _logger;

        await RunScenarioInternalAsync(iotWrapper, cloudFormationClient,
 scenarioLogger);
    }

    /// <summary>
    /// Internal method to run the IoT Basics scenario with injected
 dependencies.
    /// </summary>
    /// <param name="iotWrapper">The IoT wrapper instance.</param>
    /// <param name="cloudFormationClient">The CloudFormation client instance.</
param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    private static async Task RunScenarioInternalAsync(IoTWrapper iotWrapper,
 IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        string thingName = $"iot-thing-{Guid.NewGuid():N}";
```

```
        string certificateArn = "";
        string certificateId = "";
        string ruleName = $"iotruledefault";
        string snsTopicArn = "";

        try
        {
            // Step 1: Create an AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("1. Create an AWS IoT Thing.");
            Console.WriteLine("An AWS IoT Thing represents a virtual entity in
the AWS IoT service that can be associated with a physical device.");
            Console.WriteLine();

            if (IsInteractive)
            {
                Console.Write("Enter Thing name: ");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    thingName = userInput;
            }
            else
            {
                Console.WriteLine($"Using default Thing name: {thingName}");
            }

            var thingArn = await iotWrapper.CreateThingAsync(thingName);
            Console.WriteLine($"{thingName} was successfully created. The ARN
value is {thingArn}");
            Console.WriteLine(new string('-', 80));

            // Step 1.1: List AWS IoT Things
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("2. List AWS IoT Things.");
            Console.WriteLine("Now let's list the IoT Things to see the Thing we
just created.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var things = await iotWrapper.ListThingsAsync();
```

```
            Console.WriteLine($"Found {things.Count} IoT Things:");
            foreach (var thing in things.Take(10)) // Show first 10 things
            {
                Console.WriteLine($"Thing Name: {thing.ThingName}");
                Console.WriteLine($"Thing ARN: {thing.ThingArn}");
                if (thing.Attributes != null && thing.Attributes.Any())
                {
                    Console.WriteLine("Attributes:");
                    foreach (var attr in thing.Attributes)
                    {
                        Console.WriteLine($"  {attr.Key}: {attr.Value}");
                    }
                }
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 2: Generate a Device Certificate
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("3. Generate a device certificate.");
            Console.WriteLine("A device certificate performs a role in securing
the communication between devices (Things) and the AWS IoT platform.");
            Console.WriteLine();

            var createCert = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to create a certificate for
{thingName}? (y/n)");
                createCert = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Creating certificate for {thingName}...");
            }

            if (createCert?.ToLower() == "y")
            {
                var certificateResult = await
iotWrapper.CreateKeysAndCertificateAsync();
                if (certificateResult.HasValue)
                {
                    var (certArn, certPem, certId) = certificateResult.Value;
```

```
                    certificateArn = certArn;
                    certificateId = certId;

                    Console.WriteLine($"\nCertificate:");
                    // Show only first few lines of certificate for brevity
                    var lines = certPem.Split('\n');
                    for (int i = 0; i < Math.Min(lines.Length, 5); i++)
                    {
                        Console.WriteLine(lines[i]);
                    }
                    if (lines.Length > 5)
                    {
                        Console.WriteLine("...");
                    }

                    Console.WriteLine($"\nCertificate ARN:");
                    Console.WriteLine(certificateArn);

                    // Step 3: Attach the Certificate to the AWS IoT Thing
                    Console.WriteLine("Attach the certificate to the AWS IoT
Thing.");
                    var attachResult = await
iotWrapper.AttachThingPrincipalAsync(thingName, certificateArn);
                    if (attachResult)
                    {
                        Console.WriteLine("Certificate attached to Thing
successfully.");
                    }
                    else
                    {
                        Console.WriteLine("Failed to attach certificate to
Thing.");
                    }

                    Console.WriteLine("Thing Details:");
                    Console.WriteLine($"Thing Name: {thingName}");
                    Console.WriteLine($"Thing ARN: {thingArn}");
                }
                else
                {
                    Console.WriteLine("Failed to create certificate.");
                }
            }
            Console.WriteLine(new string('-', 80));
```

```
            // Step 4: Update an AWS IoT Thing with Attributes
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("4. Update an AWS IoT Thing with Attributes.");
            Console.WriteLine("IoT Thing attributes, represented as key-value
pairs, offer a pivotal advantage in facilitating efficient data");
            Console.WriteLine("management and retrieval within the AWS IoT
ecosystem.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var attributes = new Dictionary<string, string>
            {
                { "Location", "Seattle" },
                { "DeviceType", "Sensor" },
                { "Firmware", "1.2.3" }
            };

            await iotWrapper.UpdateThingAsync(thingName, attributes);
            Console.WriteLine("Thing attributes updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 5: Return a unique endpoint specific to the Amazon Web
Services account
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("5. Return a unique endpoint specific to the Amazon
Web Services account.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var endpoint = await iotWrapper.DescribeEndpointAsync();
            if (endpoint != null)
            {
                var subdomain = endpoint.Split('.')[0];
                Console.WriteLine($"Extracted subdomain: {subdomain}");
                Console.WriteLine($"Full Endpoint URL: https://{endpoint}");
```

```
            }
            else
            {
                Console.WriteLine("Failed to retrieve endpoint.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 6: List your AWS IoT certificates
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("6. List your AWS IoT certificates");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var certificates = await iotWrapper.ListCertificatesAsync();
            foreach (var cert in certificates.Take(5)) // Show first 5
certificates
            {
                Console.WriteLine($"Cert id: {cert.CertificateId}");
                Console.WriteLine($"Cert Arn: {cert.CertificateArn}");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 7: Create an IoT shadow
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("7. Update an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowPayload = JsonSerializer.Serialize(new
            {
                state = new
                {
                    desired = new
                    {
                        temperature = 25,
```

```
                    humidity = 50
                }
            }
        });

        await iotWrapper.UpdateThingShadowAsync(thingName, shadowPayload);
        Console.WriteLine("Thing Shadow updated successfully.");
        Console.WriteLine(new string('-', 80));

        // Step 8: Write out the state information, in JSON format
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("8. Write out the state information, in JSON
format.");
        if (IsInteractive)
        {
            Console.WriteLine("Press Enter to continue...");
            Console.ReadLine();
        }

        var shadowData = await iotWrapper.GetThingShadowAsync(thingName);
        Console.WriteLine($"Received Shadow Data: {shadowData}");
        Console.WriteLine(new string('-', 80));

        // Step 9: Set up resources (SNS topic and IAM role) and create a
rule
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("9. Set up resources and create a rule");
        Console.WriteLine();

        // Deploy CloudFormation stack to create SNS topic and IAM role
        Console.WriteLine("Deploying CloudFormation stack to create SNS topic
and IAM role...");

        var deployStack = !IsInteractive || GetYesNoResponse("Would you like
to deploy the CloudFormation stack? (y/n) ");
        if (deployStack)
        {
            if (IsInteractive)
            {
                Console.Write(
                    $"Enter stack resource file path (or press Enter for
default '{_stackResourcePath}'): ");
                var userResourcePath = Console.ReadLine();
                if (!string.IsNullOrEmpty(userResourcePath))
```

```
                    _stackResourcePath = userResourcePath;
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);

                if (deploySuccess)
                {
                    // Get stack outputs
                    var stackOutputs = await GetStackOutputs(_stackName,
cloudFormationClient, scenarioLogger);
                    if (stackOutputs != null)
                    {
                        snsTopicArn = stackOutputs["SNSTopicArn"];
                        string roleArn = stackOutputs["RoleArn"];

                        Console.WriteLine($"Successfully deployed stack. SNS
topic: {snsTopicArn}");
                        Console.WriteLine($"Successfully deployed stack. IAM
role: {roleArn}");

                        if (IsInteractive)
                        {
                            Console.Write($"Enter Rule name (press Enter for
default '{ruleName}'): ");
                            var userRuleName = Console.ReadLine();
                            if (!string.IsNullOrEmpty(userRuleName))
                                ruleName = userRuleName;
                        }
                        else
                        {
                            Console.WriteLine($"Using default rule name:
{ruleName}");
                        }

                        // Now create the IoT rule with the CloudFormation
outputs
                        var ruleResult = await
iotWrapper.CreateTopicRuleAsync(ruleName, snsTopicArn, roleArn);
                        if (ruleResult)
                        {
                            Console.WriteLine("IoT Rule created successfully.");
```

```
                        }
                        else
                        {
                            Console.WriteLine("Failed to create IoT rule.");
                        }
                    }
                    else
                    {
                        Console.WriteLine("Failed to get stack outputs. Skipping
rule creation.");
                    }
                }
                else
                {
                    Console.WriteLine("Failed to deploy CloudFormation stack.
Skipping rule creation.");
                }
            }
            else
            {
                Console.WriteLine("Skipping CloudFormation stack deployment and
rule creation.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 10: List your rules
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("10. List your rules.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var rules = await iotWrapper.ListTopicRulesAsync();
            Console.WriteLine("List of IoT Rules:");
            foreach (var rule in rules.Take(5)) // Show first 5 rules
            {
                Console.WriteLine($"Rule Name: {rule.RuleName}");
                Console.WriteLine($"Rule ARN: {rule.RuleArn}");
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));
```

```
            // Step 11: Search things using the Thing name
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("11. Search things using the Thing name.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var searchResults = await iotWrapper.SearchIndexAsync($"thingName:
{thingName}");
            if (searchResults.Any())
            {
                Console.WriteLine($"Thing id found using search is
 {searchResults.First().ThingId}");
            }
            else
            {
                Console.WriteLine($"No search results found for Thing:
 {thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 12: Cleanup - Detach and delete certificate
            if (!string.IsNullOrEmpty(certificateArn))
            {
                Console.WriteLine(new string('-', 80));
                var deleteCert = "y";
                if (IsInteractive)
                {
                    Console.Write($"Do you want to detach and delete the
 certificate for {thingName}? (y/n)");
                    deleteCert = Console.ReadLine();
                }
                else
                {
                    Console.WriteLine($"Detaching and deleting certificate for
 {thingName}...");
                }

                if (deleteCert?.ToLower() == "y")
                {
```

```
                    Console.WriteLine("12. You selected to detach and delete the
certificate.");
                    if (IsInteractive)
                    {
                        Console.WriteLine("Press Enter to continue...");
                        Console.ReadLine();
                    }

                    await iotWrapper.DetachThingPrincipalAsync(thingName,
certificateArn);
                    Console.WriteLine($"{certificateArn} was successfully removed
from {thingName}");

                    await iotWrapper.DeleteCertificateAsync(certificateId);
                    Console.WriteLine($"{certificateArn} was successfully
deleted.");
                }
                Console.WriteLine(new string('-', 80));
            }

            // Step 13: Delete the AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("13. Delete the AWS IoT Thing.");
            var deleteThing = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to delete the IoT Thing? (y/n)");
                deleteThing = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Deleting IoT Thing {thingName}...");
            }

            if (deleteThing?.ToLower() == "y")
            {
                await iotWrapper.DeleteThingAsync(thingName);
                Console.WriteLine($"Deleted Thing {thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 14: Clean up CloudFormation stack
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
```

```
                Console.WriteLine(new string('-', 80));
                Console.WriteLine("14. Clean up CloudFormation stack.");
                Console.WriteLine("Deleting the CloudFormation stack and all
 resources...");

                var cleanup = !IsInteractive || GetYesNoResponse("Do you want to
 delete the CloudFormation stack and all resources? (y/n) ");
                if (cleanup)
                {
                    var ruleCleanupSuccess = await
 iotWrapper.DeleteTopicRuleAsync(ruleName);

                    var stackCleanupSuccess = await
 DeleteCloudFormationStack(_stackName, cloudFormationClient, scenarioLogger);
                    if (ruleCleanupSuccess && stackCleanupSuccess)
                    {
                        Console.WriteLine("Successfully cleaned up CloudFormation
 stack and all resources.");
                    }
                    else
                    {
                        Console.WriteLine("Some cleanup operations failed. Check
 the logs for details.");
                    }
                }
                else
                {
                    Console.WriteLine($"Resources will remain. Stack name:
 {_stackName}");
                }
                Console.WriteLine(new string('-', 80));
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "Error occurred during scenario
 execution.");

            // Cleanup on error
            if (!string.IsNullOrEmpty(certificateArn) && !
string.IsNullOrEmpty(thingName))
            {
                try
                {
```

```
                    await iotWrapper.DetachThingPrincipalAsync(thingName,
certificateArn);
                    await iotWrapper.DeleteCertificateAsync(certificateId);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
                }
            }

            if (!string.IsNullOrEmpty(thingName))
            {
                try
                {
                    await iotWrapper.DeleteThingAsync(thingName);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during Thing
cleanup.");
                }
            }

            // Clean up CloudFormation stack on error
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                try
                {
                    await _iotWrapper.DeleteTopicRuleAsync(ruleName);
                    await DeleteCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during
CloudFormation stack cleanup.");
                }
            }

            throw;
        }
    }

    /// <summary>
```

```
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
                Capabilities = new List<string>
{ Capability.CAPABILITY_NAMED_IAM }
            };

            var response = await cloudFormationClient.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await
WaitForStackCompletion(response.StackId, cloudFormationClient, scenarioLogger);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created
successfully.");
                    return true;
                }
                else
                {
                    scenarioLogger.LogError($"CloudFormation stack creation
failed: {stackName}");
                    return false;
                }
            }
```

```
            else
            {
                scenarioLogger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            scenarioLogger.LogWarning($"CloudFormation stack '{stackName}'
already exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName,
cloudFormationClient, scenarioLogger);
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };
```

```
            var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError("Timed out waiting for CloudFormation stack
creation to complete.");
        return false;
    }

    /// <summary>
    /// Gets the outputs from the CloudFormation stack.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>A dictionary of stack outputs.</returns>
    private static async Task<Dictionary<string, string>?> GetStackOutputs(string
stackName, IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics>
scenarioLogger)
    {
        try
        {
            var describeStacksRequest = new DescribeStacksRequest
```

```
            {
                StackName = stackName
            };

            var response = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (response.Stacks.Count > 0)
            {
                var outputs = new Dictionary<string, string>();
                foreach (var output in response.Stacks[0].Outputs)
                {
                    outputs[output.OutputKey] = output.OutputValue;
                }
                return outputs;
            }

            return null;
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"Failed to get stack outputs for
{stackName}");
            return null;
        }
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        try
        {
            var request = new DeleteStackRequest
            {
                StackName = stackName
            };
```

```
            await cloudFormationClient.DeleteStackAsync(request);
            Console.WriteLine($"CloudFormation stack '{stackName}' is being
deleted. This may take a few minutes.");

            bool stackDeleted = await WaitForStackDeletion(stackName,
cloudFormationClient, scenarioLogger);

            if (stackDeleted)
            {
                Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
                return true;
            }
            else
            {
                scenarioLogger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
                return false;
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deleting the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
```

```
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
```

```
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (IsInteractive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press
 Enter for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = new System.Text.RegularExpressions.Regex("[a-zA-Z][-
a-zA-Z0-9]*");
                if (!regex.IsMatch(input))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
 {_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }
}
```

Una classe wrapper per i metodi SDK. AWS IoT

```
/// <summary>
/// Wrapper methods to use Amazon IoT Core with .NET.
/// </summary>
public class IoTWrapper
{
    private readonly IAmazonIoT _amazonIoT;
    private readonly IAmazonIotData _amazonIotData;
    private readonly ILogger<IoTWrapper> _logger;

    /// <summary>
    /// Constructor for the IoT wrapper.
```

```
    /// </summary>
    /// <param name="amazonIoT">The injected IoT client.</param>
    /// <param name="amazonIotData">The injected IoT Data client.</param>
    /// <param name="logger">The injected logger.</param>
    public IoTWrapper(IAmazonIoT amazonIoT, IAmazonIotData amazonIotData,
 ILogger<IoTWrapper> logger)
    {
        _amazonIoT = amazonIoT;
        _amazonIotData = amazonIotData;
        _logger = logger;
    }

    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</
returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
 {response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
 {ex.Message}");
            return null;
        }
```

```
    }

    /// <summary>
    /// Creates a device certificate for AWS IoT.
    /// </summary>
    /// <returns>The certificate details including ARN and certificate PEM, or
 null if creation failed.</returns>
    public async Task<(string CertificateArn, string CertificatePem, string
 CertificateId)?> CreateKeysAndCertificateAsync()
    {
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
            };

            var response = await
 _amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
 {response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
 response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
 {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</
param>
```

```
   /// <returns>True if successful, false otherwise.</returns>
   public async Task<bool> AttachThingPrincipalAsync(string thingName, string
certificateArn)
   {
       try
       {
           var request = new AttachThingPrincipalRequest
           {
               ThingName = thingName,
               Principal = certificateArn
           };

           await _amazonIoT.AttachThingPrincipalAsync(request);
           _logger.LogInformation($"Attached certificate {certificateArn} to
Thing {thingName}");
           return true;
       }
       catch (Amazon.IoT.Model.ResourceNotFoundException ex)
       {
           _logger.LogError($"Cannot attach certificate - resource not found:
{ex.Message}");
           return false;
       }
       catch (Exception ex)
       {
           _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
{ex.Message}");
           return false;
       }
   }

   /// <summary>
   /// Updates an IoT Thing with attributes.
   /// </summary>
   /// <param name="thingName">The name of the Thing to update.</param>
   /// <param name="attributes">Dictionary of attributes to add.</param>
   /// <returns>True if successful, false otherwise.</returns>
   public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
   {
       try
       {
           var request = new UpdateThingRequest
           {
```

```
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
```

```
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Lists all certificates associated with the account.
    /// </summary>
    /// <returns>List of certificate information, or empty list if listing
failed.</returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
        try
        {
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);

            _logger.LogInformation($"Retrieved {response.Certificates.Count}
certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
{ex.Message}");
            return new List<Certificate>();
        }
    }
```

```
    /// <summary>
    /// Updates the Thing's shadow with new state information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="shadowPayload">The shadow payload in JSON format.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingShadowAsync(string thingName, string
 shadowPayload)
    {
        try
        {
            var request = new UpdateThingShadowRequest
            {
                ThingName = thingName,
                Payload = new
 MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
            };

            await _amazonIotData.UpdateThingShadowAsync(request);
            _logger.LogInformation($"Updated shadow for Thing {thingName}");
            return true;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing shadow - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing shadow. Here's why:
 {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the Thing's shadow information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <returns>The shadow data as a string, or null if retrieval failed.</
returns>
    public async Task<string?> GetThingShadowAsync(string thingName)
    {
```

```
        try
        {
            var request = new GetThingShadowRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIotData.GetThingShadowAsync(request);
            using var reader = new StreamReader(response.Payload);
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
 {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</
param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
 snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
```

```
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };

            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteTopicRuleAsync(string ruleName)
    {
        try
```

```
        {
            var request = new DeleteTopicRuleRequest
            {
                RuleName = ruleName,
            };

            await _amazonIoT.DeleteTopicRuleAsync(request);
            _logger.LogInformation($"Deleted IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogWarning($"Rule {ruleName} not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Lists all IoT topic rules.
    /// </summary>
    /// <returns>List of topic rules, or empty list if listing failed.</returns>
    public async Task<List<TopicRuleListItem>> ListTopicRulesAsync()
    {
        try
        {
            var request = new ListTopicRulesRequest();
            var response = await _amazonIoT.ListTopicRulesAsync(request);

            _logger.LogInformation($"Retrieved {response.Rules.Count} IoT
rules");
            return response.Rules;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list topic rules. Here's why:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
    }

    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count}
Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
```

```
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting
for the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
    {
        try
        {
            // Update indexing configuration to REGISTRY mode
            _logger.LogInformation("Setting up IoT search indexing
configuration...");
            await _amazonIoT.UpdateIndexingConfigurationAsync(
                new UpdateIndexingConfigurationRequest()
                {
                    ThingIndexingConfiguration = new ThingIndexingConfiguration()
                    {
                        ThingIndexingMode = ThingIndexingMode.REGISTRY
                    }
                });

            _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

            // Wait for the index to be set up - this can take some time
            const int maxRetries = 10;
            const int retryDelaySeconds = 10;

            for (int attempt = 1; attempt <= maxRetries; attempt++)
            {
```

```
                try
                {
                    _logger.LogInformation($"Waiting for index to be ready
(attempt {attempt}/{maxRetries})...");
                    await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                    // Try to get the current indexing configuration to see if
it's ready
                    var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                    if
(configResponse.ThingIndexingConfiguration?.ThingIndexingMode ==
ThingIndexingMode.REGISTRY)
                    {
                        // Try the search again
                        var request = new SearchIndexRequest
                        {
                            QueryString = queryString
                        };

                        var response = await
_amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }
```

```
            _logger.LogWarning("Timeout waiting for search index to be ready
 after configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
 why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"Couldn't detach certificate from Thing. Here's
why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
```

```
        }
    }

    /// <summary>
    /// Deletes an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
```

```
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
 paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
 next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }

}
```

C++

SDK per C++

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea qualsiasi AWS IoT cosa.

```
    Aws::String thingName = askQuestion("Enter a thing name: ");

    if (!createThing(thingName, clientConfiguration)) {
        std::cerr << "Exiting because createThing failed." << std::endl;
        cleanup("", "", "", "", "", false, clientConfiguration);
        return false;
    }
```

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

Generare e associare un certificato del dispositivo.

```
    Aws::String certificateARN;
    Aws::String certificateID;
    if (askYesNoQuestion("Would you like to create a certificate for your thing?
(y/n) ")) {
        Aws::String outputFolder;
        if (askYesNoQuestion(
                "Would you like to save the certificate and keys to file? (y/n)
")) {
            outputFolder = std::filesystem::current_path();
            outputFolder += "/device_keys_and_certificates";

            std::filesystem::create_directories(outputFolder);

            std::cout << "The certificate and keys will be saved to the folder: "
                      << outputFolder << std::endl;
        }

        if (!createKeysAndCertificate(outputFolder, certificateARN,
certificateID,
                                       clientConfiguration)) {
            std::cerr << "Exiting because createKeysAndCertificate failed."
                      << std::endl;
            cleanup(thingName, "", "", "", "", false, clientConfiguration);
            return false;
        }

        std::cout << "\nNext, the certificate will be attached to the thing.\n"
                  << std::endl;
        if (!attachThingPrincipal(certificateARN, thingName,
clientConfiguration)) {
            std::cerr << "Exiting because attachThingPrincipal failed." <<
std::endl;
            cleanup(thingName, certificateARN, certificateID, "", "",
                    false,
                    clientConfiguration);
```

```
            return false;
        }
    }
```

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;
```

```
            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Esegui varie operazioni sulla AWS IoT cosa.

```
    if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
 "v2.0"} }, clientConfiguration)) {
        std::cerr << "Exiting because updateThing failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
```

```
   printAsterisksLine();

   std::cout << "Now an endpoint will be retrieved for your account.\n" <<
std::endl;
   std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
Locator that serves as the entry point\n"
   << "for communication between IoT devices and the AWS IoT service." <<
std::endl;

   askQuestion("Press Enter to continue:", alwaysTrueTest);

   Aws::String endpoint;
   if (!describeEndpoint(endpoint, clientConfiguration)) {
       std::cerr << "Exiting because getEndpoint failed." << std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }
   std::cout <<"Your endpoint is " << endpoint << "." << std::endl;
   printAsterisksLine();

   std::cout << "Now the certificates in your account will be listed." <<
std::endl;
   askQuestion("Press Enter to continue:", alwaysTrueTest);

   if (!listCertificates(clientConfiguration)) {
       std::cerr << "Exiting because listCertificates failed." << std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }

   printAsterisksLine();

   std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
   std::cout << "A thing shadow refers to a feature that enables you to create a
virtual representation, or \"shadow,\"\n"
   << "of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between\n"
   << "the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow." << std::endl;
   askQuestion("Press Enter to continue:", alwaysTrueTest);
```

```
    if (!updateThingShadow(thingName, R"({"state":{"reported":
{"temperature":25,"humidity":50}}})", clientConfiguration)) {
        std::cerr << "Exiting because updateThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now, the state information for the shadow will be retrieved.\n"
 << std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String shadowState;
    if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
        std::cerr << "Exiting because getThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
    std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

    printAsterisksLine();

    std::cout << "A rule with now be added to to the thing.\n" << std::endl;
    std::cout << "Any user who has permission to create rules will be able to
 access data processed by the rule." << std::endl;
    std::cout << "In this case, the rule will use an Simple Notification Service
(SNS) topic and an IAM rule." << std::endl;
    std::cout << "These resources will be created using a CloudFormation
 template." << std::endl;
    std::cout << "Stack creation may take a few minutes." << std::endl;

    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    Aws::Map<Aws::String, Aws::String> outputs
 =createCloudFormationStack(STACK_NAME,clientConfiguration);
    if (outputs.empty()) {
        std::cerr << "Exiting because createCloudFormationStack failed." <<
 std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
```

```
    // Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
    auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
    auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
    if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
        std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
        "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack."
 << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    Aws::String topicArn = topicArnIter->second;
    Aws::String roleArn = roleArnIter->second;
    Aws::String sqlStatement = "SELECT * FROM '";
    sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
    sqlStatement += "'";

    printAsterisksLine();

    std::cout << "Now a rule will be created.\n" << std::endl;
    std::cout << "Rules are an administrator-level action. Any user who has
 permission\n"
                 << "to create rules will be able to access data processed by the
 rule." << std::endl;
    std::cout << "In this case, the rule will use an SNS topic" << std::endl;
    std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
 std::endl;
    std::cout << "For more information on IoT SQL, see https://
docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html" <<
 std::endl;
    Aws::String ruleName = askQuestion("Enter a rule name: ");
    if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
 clientConfiguration)) {
        std::cerr << "Exiting because createRule failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
```

```
    std::cout << "Now your rules will be listed.\n" << std::endl;
    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    if (!listTopicRules(clientConfiguration)) {
        std::cerr << "Exiting because listRules failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
    Aws::String queryString = "thingName:" + thingName;
    std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
    << queryString << "'.\n" << std::endl;
    std::cout << "For query information, see https://docs.aws.amazon.com/iot/
latest/developerguide/query-syntax.html" << std::endl;

    std::cout << "For this query to work, thing indexing must be enabled in your
 account.\n"
    << "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
       << "or it can be done programmatically." << std::endl;
    std::cout << "For more information, see https://docs.aws.amazon.com/iot/
latest/developerguide/managing-index.html" << std::endl;
    if (askYesNoQuestion("Do you want to enable thing indexing in your account?
 (y/n) "))
    {
        Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

 thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGI

 thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnec
        // The ThingGroupIndexingConfiguration object is ignored if not set.
        Aws::IoT::Model::ThingGroupIndexingConfiguration
 thingGroupIndexingConfiguration;
        if (!updateIndexingConfiguration(thingIndexingConfiguration,
 thingGroupIndexingConfiguration, clientConfiguration)) {
            std::cerr << "Exiting because updateIndexingConfiguration failed." <<
 std::endl;
            cleanup(thingName, certificateARN, certificateID, STACK_NAME,
                    ruleName, false,
                    clientConfiguration);
            return false;
```

```
        }
    }

    if (!searchIndex(queryString, clientConfiguration)) {

        std::cerr << "Exiting because searchIndex failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }
```

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;
```

```
    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}

//! Update the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient
 iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
```

```
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});
```

```
    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

//! Lists the AWS IoT topic rules.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listTopicRules(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

    Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::ListTopicRulesOutcome outcome =
 iotClient.ListTopicRules(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListTopicRulesResult &result =
 outcome.GetResult();
            allRules.insert(allRules.end(),
                            result.GetRules().cbegin(),
                            result.GetRules().cend());

            nextToken = result.GetNextToken();
        }
```

```
        else {
            std::cerr << "ListTopicRules error: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
              << std::endl;
    for (auto &rule: allRules) {
        std::cout << "  Rule name: " << rule.GetRuleName() << ", rule ARN: "
                  << rule.GetRuleArn() << "." << std::endl;
    }

    return true;
}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);
```

```
        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

Eliminare le risorse.

```
bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
 &certificateARN,
                     const Aws::String &certificateID, const Aws::String
 &stackName,
                     const Aws::String &ruleName, bool askForConfirmation,
                     const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
                              askYesNoQuestion("Delete the rule '" + ruleName +
                                               "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
```

```
    }

    Aws::CloudFormation::CloudFormationClient
 cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
                               askYesNoQuestion(
                                       "Delete the CloudFormation stack '" +
 stackName +
                                       "'? (y/n) "))) {
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
                                    askYesNoQuestion("Delete the certificate '" +
                                                     certificateARN + "'? (y/n)
 "))) {
        result &= detachThingPrincipal(certificateARN, thingName,
 clientConfiguration);
        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||
                               askYesNoQuestion("Delete the thing '" + thingName
 +
                                                "'? (y/n) "))) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}
```

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
```

```
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
            request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Java

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo che dimostri le AWS IoT funzionalità.

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
```

```
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
 with AWS IOT.
                    snsAction  - An ARN of an SNS topic.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
```

```
        System.out.println("Welcome to the AWS IoT basics scenario.");
        System.out.println("""
            This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series
of steps,
            including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
            It utilizes the AWS SDK for Java V2 and incorporates functionality
for creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS
IoT capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

            Let's get started...

            """);
        System.out.println(DASHES);

        System.out.println("1. Create an AWS IoT Thing.");
        System.out.println("""
            An AWS IoT Thing represents a virtual entity in the AWS IoT service
that can be associated with
            a physical device.
            """);
        // Prompt the user for input.
        System.out.print("Enter Thing name: ");
        thingName = scanner.nextLine();
        iotActions.createIoTThing(thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Generate a device certificate.");
        System.out.println("""
            A device certificate performs a role in securing the communication
between devices (Things)
            and the AWS IoT platform.
            """);

        System.out.print("Do you want to create a certificate for " +thingName
+"? (y/n)");
        String certAns = scanner.nextLine();
        String certificateArn="" ;
        if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
            certificateArn = iotActions.createCertificate();
```

```
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(thingName, certificateArn);
        } else {
            System.out.println("A device certificate was not created.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Update an AWS IoT Thing with Attributes.");
        System.out.println("""
             IoT Thing attributes, represented as key-value pairs, offer a
pivotal advantage in facilitating efficient data
             management and retrieval within the AWS IoT ecosystem.
            """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Return a unique endpoint specific to the Amazon
Web Services account.");
        System.out.println("""
            An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
        System.out.println("""
            A Thing Shadow refers to a feature that enables you to create a
virtual representation, or "shadow,"
            of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
            the cloud and the device itself. and the AWS IoT service. For
example, you can write and retrieve JSON data from a Thing Shadow.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON
format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Creates a rule");
        System.out.println("""
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """);
        System.out.print("Enter Rule name: ");
        ruleName = scanner.nextLine();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        if (certificateArn.length() > 0) {
            System.out.print("Do you want to detach and delete the certificate
for " +thingName +"? (y/n)");
            String delAns = scanner.nextLine();
            if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
                System.out.println("11. You selected to detach amd delete the
certificate.");
                waitForInputToContinue(scanner);
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } else {
                System.out.println("11. You selected not to delete the
certificate.");
            }
        } else {
            System.out.println("11. You did not create a certificate so there is
nothing to delete.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete the AWS IoT Thing.");
        System.out.print("Do you want to delete the IoT Thing? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            iotActions.deleteIoTThing(thingName);
        } else {
            System.out.println("The IoT Thing was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS IoT workflow has successfully completed.");
```

```
        System.out.println(DASHES);
    }


    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Una classe wrapper per i metodi SDK. AWS IoT

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotAsyncClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.Certificate;
import
 software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
```

```
import software.amazon.awssdk.services.iot.model.CreateTopicRuleResponse;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DeleteThingResponse;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IotActions {

    private static IotAsyncClient iotAsyncClient;

    private static IotDataPlaneAsyncClient iotAsyncDataPlaneClient;

    private static final String TOPIC = "your-iot-topic";

    private static IotDataPlaneAsyncClient getAsyncDataPlaneClient() {
```

```
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncDataPlaneClient == null) {
            iotAsyncDataPlaneClient = IotDataPlaneAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();
```

```
        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT
certificate.
     * If the request is successful, it prints the certificate details and
returns the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
```

```
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }

    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Attaches a certificate to an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to attach.
     *
     * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
     * If the request is successful, it prints a confirmation message and
additional information about the Thing.
     * If an exception occurs, it prints the error message.
     */
    public void attachCertificateToThing(String thingName, String certificateArn)
{
        AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();

        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing
successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
```

```
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }

    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an
IoT Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
 the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

```
    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\
\.com");

        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);

        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
        }
        return "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
 getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
```

```
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }

    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a
Thing's shadow.
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }

    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();
```

```
        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }

    /**
     * Lists IoT rules asynchronously.
     *
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
     * If an exception occurs, it prints the error message.
     */
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                System.out.println("List of IoT Rules:");
```

```
                List<TopicRuleListItem> ruleList =
listTopicRulesResponse.rules();
                for (TopicRuleListItem rule : ruleList) {
                    System.out.println("Rule Name: " + rule.ruleName());
                    System.out.println("Rule ARN: " + rule.ruleArn());
                    System.out.println("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });

        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
```

```
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }

    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
```

```
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed
from " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully
deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
```

```
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    // Get the cert Id  from the Cert ARN value.
    private String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
```

```
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") +
 1);
    }
}
```

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
```

```
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development
 environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-
getting-started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
 IOT.
            snsAction  - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
```

```
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
        This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service.
        The program guides you through a series of steps, including creating an
IoT thing, generating a device certificate,
        updating the thing with attributes, and so on.

        It utilizes the AWS SDK for Kotlin and incorporates functionality for
creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that
can be associated with a physical device.
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
```

```
        A device certificate performs a role in securing the communication
between devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    println(
        """
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    println(
        """
        An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
```

```
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
    if (certificateArn!!.isNotEmpty()) {
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
    }
    println(DASHES)

    println(DASHES)
    println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
    println(
        """
        A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
        of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
        the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow.

        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateShawdowThing(thingName)
    println(DASHES)

    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
    println("8. Creates a rule")
    println(
        """
```

```
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
    createIoTRule(roleARN, ruleName, snsAction)
    println(DASHES)

    println(DASHES)
    println("9. List your rules.")
    print("Press Enter to continue...")
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName?
(y/n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
delete.")
    }
```

```
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
```

```
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
```

```
        val listTopicRulesResponse =
 iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
```

```
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
```

```
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
```

```
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
```

```
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

Python

SDK per Python (Boto3)

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea una classe wrapper IoT per gestire le operazioni.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)


    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
 thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

```
    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response =
self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
```

```
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response


    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response =
self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]


    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
```

```
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn":
role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def list_topic_rules(self):
        """
        Lists AWS IoT topic rules.

        :return: The list of topic rules.
        """
        try:
            rules = []
            paginator = self.iot_client.get_paginator("list_topic_rules")
            for page in paginator.paginate():
```

```
                rules.extend(page["rules"])
            logger.info("Retrieved %s topic rules.", len(rules))
            return rules
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list topic rules. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])


    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
```

```
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update_thing_shadow(self, thing_name, shadow_state):
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_thing_shadow(self, thing_name):
        """
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
```

```
            response =
 self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return shadow
```

Esegui uno scenario interattivo che dimostri le basi dell'IoT.

```
class IoTScenario:
    """Runs an interactive scenario that shows how to use AWS IoT."""

    is_interactive = True

    def __init__(self, iot_wrapper, iot_data_client, cfn_client,
 stack_name="IoTBasicsStack", template_path=None):
        """
        :param iot_wrapper: An instance of the IoTWrapper class.
        :param iot_data_client: A Boto3 IoT Data Plane client.
        :param cfn_client: A Boto3 CloudFormation client.
        :param stack_name: Name for the CloudFormation stack.
        :param template_path: Path to the CloudFormation template file.
        """
        self.iot_wrapper = iot_wrapper
        self.iot_data_client = iot_data_client
        self.cfn_client = cfn_client
        self.thing_name = None
        self.certificate_arn = None
        self.certificate_id = None
        self.rule_name = None
        self.stack_name = stack_name
```

```
        self.template_path = template_path or "../../../scenarios/basics/iot/
iot_usecase/resources/cfn_template.yaml"

    def _deploy_stack(self):
        """Deploy CloudFormation stack and return outputs."""
        with open(self.template_path, "r") as f:
            template_body = f.read()

        try:
            self.cfn_client.create_stack(
                StackName=self.stack_name,
                TemplateBody=template_body,
                Capabilities=["CAPABILITY_NAMED_IAM"]
            )

            waiter = self.cfn_client.get_waiter("stack_create_complete")
            waiter.wait(StackName=self.stack_name)

            response = self.cfn_client.describe_stacks(StackName=self.stack_name)
            outputs = {output["OutputKey"]: output["OutputValue"]
                       for output in response["Stacks"][0]["Outputs"]}
            return outputs["SNSTopicArn"], outputs["RoleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                response =
 self.cfn_client.describe_stacks(StackName=self.stack_name)
                outputs = {output["OutputKey"]: output["OutputValue"]
                           for output in response["Stacks"][0]["Outputs"]}
                return outputs["SNSTopicArn"], outputs["RoleArn"]
            raise

    def _cleanup_stack(self):
        """Delete CloudFormation stack."""
        try:
            self.cfn_client.delete_stack(StackName=self.stack_name)
            waiter = self.cfn_client.get_waiter("stack_delete_complete")
            waiter.wait(StackName=self.stack_name)
            print("CloudFormation stack deleted successfully.")
        except ClientError as err:
            logger.error(f"Failed to delete stack: {err}")

    def run_scenario(self, thing_name, rule_name):
        """
        Runs the IoT basics scenario.
```

```
        :param thing_name: The name of the thing to create.
        :param rule_name: The name of the topic rule to create.
        """
        print("-" * 88)
        print("Welcome to the AWS IoT basics scenario!")
        print("-" * 88)
        print(
            "This scenario demonstrates how to interact with AWS IoT using the
AWS SDK for Python (Boto3).\n"
            "AWS IoT provides secure, bi-directional communication between
Internet-connected devices\n"
            "and the AWS cloud. You can manage device connections, process device
data, and build IoT applications.\n"
        )

        self.thing_name = thing_name
        self.rule_name = rule_name

        try:
            print("\nDeploying CloudFormation stack...")
            sns_topic_arn, role_arn = self._deploy_stack()
            print(f"Stack deployed. SNS Topic: {sns_topic_arn}")

            input("\nNext, we'll create an AWS IoT thing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("1. Create an AWS IoT thing")
            print("-" * 88)
            response = self.iot_wrapper.create_thing(thing_name)
            print(f"Created thing: {response['thingName']}")
            print(f"Thing ARN: {response['thingArn']}")

            input("\nNext, we'll list things. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("2. List things")
            print("-" * 88)
            things = self.iot_wrapper.list_things()
            print(f"Found {len(things)} thing(s) in your account")
            for thing in things[:5]:  # Show first 5
                print(f"  Thing name: {thing['thingName']}")
```

```
            input("\nNext, we'll generate a device certificate. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("3. Generate a device certificate")
            print("-" * 88)
            cert_response = self.iot_wrapper.create_keys_and_certificate()
            self.certificate_arn = cert_response["certificateArn"]
            self.certificate_id = cert_response["certificateId"]
            print(f"Created certificate: {self.certificate_id}")

            input("\nNext, we'll attach the certificate to the thing. Press Enter
to continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("4. Attach the certificate to the thing")
            print("-" * 88)
            self.iot_wrapper.attach_thing_principal(thing_name,
self.certificate_arn)
            print(f"Attached certificate to thing: {thing_name}")

            input("\nNext, we'll update the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("5. Update the thing shadow")
            print("-" * 88)
            shadow_state = {"state": {"reported": {"temperature": 25, "humidity":
50}}}
            self.iot_wrapper.update_thing_shadow(thing_name, shadow_state)
            print(f"Updated shadow for thing: {thing_name}")

            input("\nNext, we'll get the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("6. Get the thing shadow")
            print("-" * 88)
            shadow = self.iot_wrapper.get_thing_shadow(thing_name)
            print(f"Shadow state: {json.dumps(shadow['state'], indent=2)}")

            input("\nNext, we'll get the AWS IoT endpoint. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("7. Get the AWS IoT endpoint")
            print("-" * 88)
            endpoint = self.iot_wrapper.describe_endpoint()
            print(f"IoT endpoint: {endpoint}")
```

```
            input("\nNext, we'll list certificates. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("8. List certificates")
            print("-" * 88)
            certificates = self.iot_wrapper.list_certificates()
            print(f"Found {len(certificates)} certificate(s)")
            for cert in certificates:
                print(f"  Certificate ID: {cert['certificateId']}")
                print(f"  Certificate ARN: {cert['certificateArn']}")
                print(f"  Status: {cert['status']}")

            input("\nNext, we'll create a topic rule. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("9. Create a topic rule")
            print("-" * 88)
            self.iot_wrapper.create_topic_rule(
                rule_name, f"device/{thing_name}/data", sns_topic_arn, role_arn
            )
            print(f"Created topic rule: {rule_name}")

            input("\nNext, we'll list topic rules. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("10. List topic rules")
            print("-" * 88)
            rules = self.iot_wrapper.list_topic_rules()
            print(f"Found {len(rules)} topic rule(s)")
            for rule in rules:
                print(f"  Rule name: {rule['ruleName']}")
                print(f"  Rule ARN: {rule['ruleArn']}")

            input("\nNext, we'll configure thing indexing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("11. Configure thing indexing")
            print("-" * 88)
            self.iot_wrapper.update_indexing_configuration()
            print("Enabled thing indexing")
            print("Waiting for indexing to be ready...")
            time.sleep(10)
```

```
            input("\nNext, we'll search for things. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("12. Search for things")
            print("-" * 88)
            try:
                things = self.iot_wrapper.search_index(f"thingName:{thing_name}")
                if things:
                    print(f"Found {len(things)} thing(s) matching the query")
                    for thing in things:
                        print(f"  Thing name: {thing.get('thingName', 'N/A')}")
                        print(f"  Thing ID: {thing.get('thingId', 'N/A')}")
                else:
                    print("No things found. Indexing may take a few minutes.")
            except ClientError as err:
                if err.response["Error"]["Code"] in [
                    "IndexNotReadyException",
                    "InvalidRequestException",
                ]:
                    print("Search index not ready yet. This is expected.")
                else:
                    raise

        except ClientError as err:
            logger.error(
                "Scenario failed: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        finally:
            self._cleanup()

    def _cleanup(self):
        """Cleans up resources created during the scenario."""
        if not self.thing_name:
            return

        print("\n" + "-" * 88)
        print("Cleanup")
        print("-" * 88)

        if q.ask("Do you want to delete the resources? (y/n) ", q.is_yesno):
            try:
```

```
                if self.certificate_arn:
                    print(f"Detaching certificate from thing: {self.thing_name}")
                    self.iot_wrapper.detach_thing_principal(
                        self.thing_name, self.certificate_arn
                    )

                if self.certificate_id:
                    print(f"Deleting certificate: {self.certificate_id}")
                    self.iot_wrapper.delete_certificate(self.certificate_id)

                if self.thing_name:
                    print(f"Deleting thing: {self.thing_name}")
                    self.iot_wrapper.delete_thing(self.thing_name)

                if self.rule_name:
                    print(f"Deleting topic rule: {self.rule_name}")
                    self.iot_wrapper.delete_topic_rule(self.rule_name)

                self._cleanup_stack()
                print("Resources deleted successfully.")
            except ClientError as err:
                logger.error(
                    "Cleanup failed: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
        else:
            print("Resources will remain in your account.")

        print("\n" + "-" * 88)
        print("Thanks for using AWS IoT!")
        print("-" * 88)
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Azioni per l'utilizzo AWS IoT AWS SDKs

I seguenti esempi di codice mostrano come eseguire singole AWS IoT azioni con AWS SDKs. Ogni esempio include un collegamento a GitHub, dove sono disponibili le istruzioni per la configurazione e l'esecuzione del codice.

Gli esempi seguenti includono solo le azioni più comunemente utilizzate. Per un elenco completo, consulta la documentazione di riferimento dell'API AWS IoT.

Esempi

- Utilizzo AttachThingPrincipal con un AWS SDK o una CLI
- Utilizzo CreateKeysAndCertificate con un AWS SDK o una CLI
- Utilizzo CreateThing con un AWS SDK o una CLI
- Utilizzo CreateTopicRule con un AWS SDK o una CLI
- Utilizzo DeleteCertificate con un AWS SDK o una CLI
- Utilizzo DeleteThing con un AWS SDK o una CLI
- Utilizzo DeleteTopicRule con un AWS SDK o una CLI
- Utilizzo DescribeEndpoint con un AWS SDK o una CLI
- Utilizzo DescribeThing con un AWS SDK o una CLI
- Utilizzo DetachThingPrincipal con un AWS SDK o una CLI
- Utilizzo ListCertificates con un AWS SDK o una CLI
- Utilizzo ListThings con un AWS SDK o una CLI
- Utilizzo SearchIndex con un AWS SDK o una CLI
- Utilizzo UpdateIndexingConfiguration con un AWS SDK o una CLI
- Utilizzo UpdateThing con un AWS SDK o una CLI

Utilizzo **`AttachThingPrincipal`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `AttachThingPrincipal`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Per i dettagli sull'API, AttachThingPrincipalconsulta AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
```

```
                    outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, AttachThingPrincipalconsulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Come associare un certificato a un oggetto

L'`attach-thing-principal`esempio seguente allega un certificato all' MyTemperatureSensor oggetto. Il certificato è identificato da un ARN. Puoi trovare l'ARN per un certificato nella console AWS IoT.

```
aws iot attach-thing-principal \
    --thing-name MyTemperatureSensor \
    --principal arn:aws:iot:us-
west-2:123456789012:cert/2e1eb273792174ec2b9bf4e9b37e6c6c6923454995060002a35159767055278e8
```

Questo comando non produce alcun output.

Per ulteriori informazioni, consulta Come gestire gli oggetti con il registro nella Guida per gli sviluppatori di AWS IoT.

- Per i dettagli sull'API, consulta AttachThingPrincipal AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
    * If the request is successful, it prints a confirmation message and
additional information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn)
{
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();

       CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
       future.whenComplete((attachResponse, ex) -> {
           if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
               System.out.println("Certificate attached to Thing
successfully.");

               // Print additional information about the Thing.
               describeThing(thingName);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " +
cause.getMessage());
               } else {
                   System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
                           attachResponse.sdkHttpResponse().statusCode());
               }
```

```
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, AttachThingPrincipalconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- Per i dettagli sull'API, AttachThingPrincipalconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal,
 thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta AttachThingPrincipal AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateKeysAndCertificate`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `CreateKeysAndCertificate`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Creates a device certificate for AWS IoT.
   /// </summary>
   /// <returns>The certificate details including ARN and certificate PEM, or
null if creation failed.</returns>
   public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
   {
```

```
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
            };

            var response = await
_amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
{response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Per i dettagli sull'API, CreateKeysAndCertificateconsulta AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
```

```
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, CreateKeysAndCertificateconsulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Come creare una coppia di chiavi RSA e generare un certificato X.509

L'esempio `create-keys-and-certificate` seguente crea una coppia di chiavi RSA a 2.048 bit e genera un certificato X.509 utilizzando la chiave pubblica emessa. Poiché è l'unica volta che l' AWS IoT fornisce la chiave privata per questo certificato, assicurati di conservarlo in un luogo sicuro.

```
aws iot create-keys-and-certificate \
    --certificate-pem-outfile "myTest.cert.pem" \
    --public-key-outfile "myTest.public.key" \
    --private-key-outfile "myTest.private.key"
```

Output:

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2
    "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\nMMEXAMPLEuuN/
```

```
dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y
+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQ
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB
\n-----END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omittted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

Per ulteriori informazioni, consulta Creare e registrare un certificato per dispositivi AWS IoT nella Guida per sviluppatori AWS IoT.

- Per i dettagli sull'API, consulta CreateKeysAndCertificate AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Creates an IoT certificate asynchronously.
    *
    * @return The ARN of the created certificate.
    * <p>
    * This method initiates an asynchronous request to create an IoT
certificate.
    * If the request is successful, it prints the certificate details and
returns the certificate ARN.
    * If an exception occurs, it prints the error message.
    */
   public String createCertificate() {
       CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
       final String[] certificateArn = {null};
       future.whenComplete((response, ex) -> {
```

```
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }
```

- Per i dettagli sull'API, CreateKeysAndCertificateconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- Per i dettagli sull'API, CreateKeysAndCertificateconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
```

```
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response =
self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response
```

- Per i dettagli sull'API, consulta CreateKeysAndCertificate AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateThing`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `CreateThing`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</
returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
 {response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
 {ex.Message}");
            return null;
        }
```

```
    }
```

- Per i dettagli sull'API, CreateThingconsulta AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, CreateThingconsulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Esempio 1: come creare un record oggetto nel registro

L'`create-thing`esempio seguente crea una voce per un dispositivo nel registro degli AWS oggetti IoT.

```
aws iot create-thing \
    --thing-name SampleIoTThing
```

Output:

```
{
    "thingName": "SampleIoTThing",
    "thingArn": "arn:aws:iot:us-west-2: 123456789012:thing/SampleIoTThing",
    "thingId": " EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE "
}
```

Esempio 2: come definire un oggetto associato a un tipo di oggetto

L'esempio `create-thing` seguente crea un oggetto con il tipo specificato e i relativi attributi.

```
aws iot create-thing \
    --thing-name "MyLightBulb" \
    --thing-type-name "LightBulb" \
    --attribute-payload "{"attributes": {"wattage":"75", "model":"123"}}"
```

Output:

```
{
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797"
}
```

Per ulteriori informazioni, consulta Come gestire gli oggetti con il registro e Tipi di oggetti nella Guida per gli sviluppatori di AWS IoT.

- Per i dettagli sull'API, vedere CreateThingin AWS CLI Command Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Creates an IoT Thing with the specified name asynchronously.
    *
    * @param thingName The name of the IoT Thing to create.
    *
    * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
    * If the request is successful, it prints the name of the thing and its ARN
value.
    * If an exception occurs, it prints the error message.
    */
   public void createIoTThing(String thingName) {
       CreateThingRequest createThingRequest = CreateThingRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
       future.whenComplete((createThingResponse, ex) -> {
           if (createThingResponse != null) {
               System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " +
cause.getMessage());
```

```
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, CreateThingconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Per i dettagli sull'API, CreateThingconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
 thing_name)
                return None
```

```
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Per i dettagli sull'API, consulta CreateThing AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateTopicRule`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `CreateTopicRule`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</
param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
```

```
   public async Task<bool> CreateTopicRuleAsync(string ruleName, string
snsTopicArn, string roleArn)
   {
       try
       {
           var request = new CreateTopicRuleRequest
           {
               RuleName = ruleName,
               TopicRulePayload = new TopicRulePayload
               {
                   Sql = "SELECT * FROM 'topic/subtopic'",
                   Description = $"Rule created by .NET example: {ruleName}",
                   Actions = new List<Amazon.IoT.Model.Action>
                   {
                       new Amazon.IoT.Model.Action
                       {
                           Sns = new SnsAction
                           {
                               TargetArn = snsTopicArn,
                               RoleArn = roleArn
                           }
                       },
                   RuleDisabled = false
               }
           };

           await _amazonIoT.CreateTopicRuleAsync(request);
           _logger.LogInformation($"Created IoT rule {ruleName}");
           return true;
       }
       catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
       {
           _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
           return false;
       }
       catch (Exception ex)
       {
           _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
           return false;
       }
   }
```

- Per i dettagli sull'API, CreateTopicRuleconsulta AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);
```

```
    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la CreateTopicRulesezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Come creare una regola che invii un avviso Amazon SNS

L'esempio `create-topic-rule` seguente crea una regola che invia un messaggio Amazon SNS quando le letture del livello di umidità del terreno, rilevate in un device shadow, sono basse.

```
aws iot create-topic-rule \
    --rule-name "LowMoistureRule" \
    --topic-rule-payload file://plant-rule.json
```

L'esempio richiede il salvataggio del seguente codice JSON in un file denominato `plant-rule.json`:

```
{
    "sql": "SELECT * FROM '$aws/things/MyRPi/shadow/update/accepted' WHERE
 state.reported.moisture = 'low'\n",
```

```
    "description": "Sends an alert whenever soil moisture level readings are too
 low.",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
            "sns": {
                "targetArn": "arn:aws:sns:us-
west-2:123456789012:MyRPiLowMoistureTopic",
                "roleArn": "arn:aws:iam::123456789012:role/service-role/
MyRPiLowMoistureTopicRole",
                "messageFormat": "RAW"
            }
    }]
}
```

Questo comando non produce alcun output.

Per ulteriori informazioni, consulta Creating an AWS IoT Rule nella AWS IoT Developers Guide.

- Per i dettagli sull'API, consulta CreateTopicRule AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
 actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
```

```
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
```

```
        });

        future.join();
    }
```

- Per i dettagli sull'API, consulta la CreateTopicRulesezione AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }
```

```
    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- Per i dettagli sull'API, CreateTopicRuleconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)
```

```
    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn":
role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta CreateTopicRule AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteCertificate`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DeleteCertificate`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/// <summary>
/// Deletes an IoT certificate.
/// </summary>
/// <param name="certificateId">The ID of the certificate to delete.</param>
/// <returns>True if successful, false otherwise.</returns>
public async Task<bool> DeleteCertificateAsync(string certificateId)
{
    try
    {
        // First, update the certificate to inactive state
        var updateRequest = new UpdateCertificateRequest
        {
            CertificateId = certificateId,
            NewStatus = CertificateStatus.INACTIVE
        };
        await _amazonIoT.UpdateCertificateAsync(updateRequest);

        // Then delete the certificate
        var deleteRequest = new DeleteCertificateRequest
        {
            CertificateId = certificateId
        };

        await _amazonIoT.DeleteCertificateAsync(deleteRequest);
        _logger.LogInformation($"Deleted certificate {certificateId}");
        return true;
    }
    catch (Amazon.IoT.Model.ResourceNotFoundException ex)
```

```
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Per i dettagli sull'API, consulta la DeleteCertificatesezione AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);
```

```
    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la DeleteCertificatesezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Come eliminare un certificato del dispositivo

L'esempio `delete-certificate` seguente elimina il certificato di dispositivo con l'ID specificato.

```
aws iot delete-certificate \
    --certificate-
id c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbee1428d216d54d53ac9
```

Questo comando non produce alcun output.

Per ulteriori informazioni, consulta la DeleteCertificatesezione AWS IoT API Reference.

- Per i dettagli sull'API, consulta DeleteCertificate AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Deletes a certificate asynchronously.
    *
    * @param certificateArn The ARN of the certificate to delete.
    *
    * This method initiates an asynchronous request to delete a certificate.
    * If the deletion is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void deleteCertificate(String certificateArn) {
       DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
           .certificateId(extractCertificateId(certificateArn))
           .build();

       CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully
deleted.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + ex.getMessage());
               }
           }
       });
```

```
        future.join();
    }
```

- Per i dettagli sull'API, consulta la DeleteCertificatesezione AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- Per i dettagli sull'API, DeleteCertificateconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta DeleteCertificate AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteThing`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DeleteThing`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/// <summary>
/// Deletes an IoT Thing.
/// </summary>
/// <param name="thingName">The name of the Thing to delete.</param>
/// <returns>True if successful, false otherwise.</returns>
public async Task<bool> DeleteThingAsync(string thingName)
{
    try
    {
        var request = new DeleteThingRequest
        {
            ThingName = thingName
        };

        await _amazonIoT.DeleteThingAsync(request);
        _logger.LogInformation($"Deleted Thing {thingName}");
        return true;
    }
    catch (Amazon.IoT.Model.ResourceNotFoundException ex)
    {
```

```
            _logger.LogError($"Cannot delete Thing - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }
```

- Per i dettagli sull'API, consulta la DeleteThingsezione AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
```

```
                    outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la DeleteThingsezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Come visualizzare informazioni dettagliate su un oggetto

L'`delete-thing`esempio seguente elimina un elemento dal registro AWS IoT per il tuo AWS account.

as iot delete-thing --thing-name» FourthBulb

Questo comando non produce alcun output.

Per ulteriori informazioni, consulta Come gestire gli oggetti con il registro nella Guida per gli sviluppatori di AWS IoT.

- Per i dettagli sull'API, consulta Command Reference. DeleteThingAWS CLI

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
```

```
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, consulta la DeleteThingsezione AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun deleteIoTThing(thingNameVal: String) {
```

```
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- Per i dettagli sull'API, DeleteThingconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_thing(self, thing_name):
```

```
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta DeleteThing AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteTopicRule`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DeleteTopicRule`.

C++

SDK per C++

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la DeleteTopicRulesezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Come eliminare una regola

L'esempio `delete-topic-rule` seguente elimina la regola specificata.

```
aws iot delete-topic-rule \
    --rule-name "LowMoistureRule"
```

Questo comando non produce alcun output.

Per ulteriori informazioni, consulta Eliminazione di una regola nella Guida per gli sviluppatori di AWS IoT.

- Per i dettagli sull'API, consulta DeleteTopicRule AWS CLICommand Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta DeleteTopicRule AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeEndpoint`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeEndpoint`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/// <summary>
/// Gets the AWS IoT endpoint URL.
/// </summary>
/// <returns>The endpoint URL, or null if retrieval failed.</returns>
public async Task<string?> DescribeEndpointAsync()
{
    try
    {
        var request = new DescribeEndpointRequest
        {
            EndpointType = "iot:Data-ATS"
```

```
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Per i dettagli sull'API, consulta la DescribeEndpointsezione AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la DescribeEndpointsezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Esempio 1: per ottenere l' AWS endpoint attuale

L'`describe-endpoint`esempio seguente recupera l' AWS endpoint predefinito a cui vengono applicati tutti i comandi.

```
aws iot describe-endpoint
```

Output:

```
{
    "endpointAddress": "abc123defghijk.iot.us-west-2.amazonaws.com"
}
```

Per ulteriori informazioni, consulta DescribeEndpointla AWS IoT Developer Guide.

Esempio 2: come ottenere l'endpoint ATS

L'esempio `describe-endpoint` seguente recupera l'endpoint Amazon Trust Services (ATS).

```
aws iot describe-endpoint \
    --endpoint-type iot:Data-ATS
```

Output:

```
{
    "endpointAddress": "abc123defghijk-ats.iot.us-west-2.amazonaws.com"
}
```

Per ulteriori informazioni, consulta Certificati X.509 e IoT nella AWSAWS IoT Developer Guide.

- Per i dettagli sull'API, consulta AWS CLI Command DescribeEndpointReference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
the IoT service.
```

```
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- Per i dettagli sull'API, consulta la DescribeEndpointsezione AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- Per i dettagli sull'API, DescribeEndpointconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
```

```
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response =
self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]
```

- Per i dettagli sull'API, consulta DescribeEndpoint AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error>
 {
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;

    println!("Endpoint address: {}", resp.endpoint_address.unwrap());

    println!();

    Ok(())
}
```

- Per i dettagli sulle API, consulta il riferimento DescribeEndpointall'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeThing`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeThing`.

C++

SDK per C++

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Describe an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);

    Aws::IoT::Model::DescribeThingOutcome outcome =
 iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing '" << result.GetThingName() << "'" <<
 std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
                  << std::endl;
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << "  attribute: " << attribute.first << "=" <<
 attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error describing thing " << thingName << ": " <<
```

```
                    outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la DescribeThingsezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Come visualizzare informazioni dettagliate su un oggetto

L'`describe-thing`esempio seguente mostra informazioni su un oggetto (dispositivo) definito nel registro AWS IoT per l' AWS account.

aws iot describe-thing --thing-name "» MyLightBulb

Output:

```
{
    "defaultClientId": "MyLightBulb",
    "thingName": "MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    },
    "version": 1
}
```

Per ulteriori informazioni, consulta Come gestire gli oggetti con il registro nella Guida per gli sviluppatori di AWS IoT.

- Per i dettagli sull'API, consulta Command Reference. DescribeThingAWS CLI

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
```

```
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, consulta la DescribeThingsezione AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- Per i dettagli sull'API, DescribeThingconsulta AWS SDK for Kotlin API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DetachThingPrincipal`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DetachThingPrincipal`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
```

```
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's
 why: {ex.Message}");
            return false;
        }
    }
```

- Per i dettagli sull'API, consulta la DetachThingPrincipalsezione AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
```

```
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
           detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la DetachThingPrincipalsezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Staccare una cosa certificate/principal da una cosa

L'esempio `detach-thing-principal` seguente rimuove un certificato che rappresenta un principale dall'oggetto specificato.

```
aws iot detach-thing-principal \
    --thing-name "MyLightBulb" \
    --principal "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
```

Questo comando non produce alcun output.

Per ulteriori informazioni, consulta Come gestire gli oggetti con il registro nella Guida per gli sviluppatori di AWS IoT.

- Per i dettagli sull'API, consulta DetachThingPrincipal AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
                .principal(certificateArn)
                .thingName(thingName)
                .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed
from " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, consulta la DetachThingPrincipalsezione AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- Per i dettagli sull'API, DetachThingPrincipalconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
 thing_name)
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta DetachThingPrincipal AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListCertificates`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListCertificates`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Lists all certificates associated with the account.
    /// </summary>
    /// <returns>List of certificate information, or empty list if listing
failed.</returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
```

```
        try
        {
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);

            _logger.LogInformation($"Retrieved {response.Certificates.Count}
 certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
 {ex.Message}");
            return new List<Certificate>();
        }
    }
```

- Per i dettagli sull'API, consulta la ListCertificatessezione AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
```

```
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}
```

- Per i dettagli sull'API, consulta la ListCertificatessezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Esempio 1: per elencare i certificati registrati nel tuo AWS account

L'esempio `list-certificates` seguente elenca tutti i certificati registrati nel tuo account. Se disponi di un limite di paging superiore rispetto al valore predefinito di 25, puoi utilizzare il valore di risposta `nextMarker` di questo comando e fornirlo al comando successivo per ottenere il successivo batch di risultati. Ripeti l'operazione finché `nextMarker` non restituisce alcun valore.

```
aws iot list-certificates
```

Output:

```
{
    "certificates": [
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
            "certificateId":
 "604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36",
            "status": "ACTIVE",
            "creationDate": 1556810537.617
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e
            "certificateId":
 "262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e",
            "status": "ACTIVE",
            "creationDate": 1546447050.885
        },
        {
            "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "certificateId":
 "b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "status": "ACTIVE",
```

```
            "creationDate": 1546292258.322
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42
            "certificateId":
 "7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42",
            "status": "ACTIVE",
            "creationDate": 1541457693.453
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3
            "certificateId":
 "54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3",
            "status": "ACTIVE",
            "creationDate": 1541113568.611
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e
            "certificateId":
 "4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
            "status": "ACTIVE",
            "creationDate": 1541022751.983
        }
    ]
}
```

- Per i dettagli sull'API, consulta ListCertificates AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
```

```
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, consulta la ListCertificatessezione AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- Per i dettagli sull'API, ListCertificatesconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
```

```
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Per i dettagli sull'API, consulta ListCertificates AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListThings`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListThings`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
```

```
                request.Marker = response.NextMarker;
                response = await _amazonIoT.ListThingsAsync(request);
            }
            else
                hasMoreResults = false;
        }

        _logger.LogInformation($"Retrieved {things.Count} Things");
        return things;
    }
    catch (Amazon.IoT.Model.ThrottlingException ex)
    {
        _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
        return new List<ThingAttribute>();
    }
    catch (Exception ex)
    {
        _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
        return new List<ThingAttribute>();
    }
}
```

- Per i dettagli sull'API, consulta la ListThingssezione AWS SDK per .NET API Reference.

CLI

AWS CLI

Esempio 1: come elencare tutti gli oggetti presenti nel registro

L'`list-things`esempio seguente elenca gli oggetti (dispositivi) definiti nel registro AWS IoT per il tuo AWS account.

```
aws iot list-things
```

Output:

```
{
    "things": [
        {
```

```
            "thingName": "ThirdBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/ThirdBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 2
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        },
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
        "thingName": "SampleIoTThing",
        "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/SampleIoTThing",
        "attributes": {},
        "version": 1
        }
    ]
}
```

Esempio 2: come elencare gli oggetti definiti con un attributo specifico

L'esempio `list-things` seguente visualizza un elenco degli oggetti contenenti un attributo denominato `wattage`.

```
aws iot list-things \
    --attribute-name wattage
```

Output:

```
{
    "things": [
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        }
    ]
}
```

Per ulteriori informazioni, consulta Come gestire gli oggetti con il registro nella Guida per gli sviluppatori di AWS IoT.

- Per i dettagli sull'API, consulta ListThings AWS CLICommand Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
```

```
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Per i dettagli sull'API, consulta ListThings AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
async fn show_things(client: &Client) -> Result<(), Error> {
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
        println!(
            "  Name:  {}",
            thing.thing_name.as_deref().unwrap_or_default()
        );
        println!(
            "  Type:  {}",
            thing.thing_type_name.as_deref().unwrap_or_default()
        );
        println!(
            "  ARN:   {}",
            thing.thing_arn.as_deref().unwrap_or_default()
        );
```

```
        println!();
    }

    println!();

    Ok(())
}
```

- Per i dettagli sulle API, consulta il riferimento ListThingsall'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`SearchIndex`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `SearchIndex`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Searches for IoT Things using the search index.
   /// </summary>
   /// <param name="queryString">The search query string.</param>
   /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
   public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
   {
       try
       {
           // First, try to perform the search
           var request = new SearchIndexRequest
```

```
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count}
Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting
for the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
```

```
   {
       try
       {
           // Update indexing configuration to REGISTRY mode
           _logger.LogInformation("Setting up IoT search indexing
configuration...");
           await _amazonIoT.UpdateIndexingConfigurationAsync(
               new UpdateIndexingConfigurationRequest()
               {
                   ThingIndexingConfiguration = new ThingIndexingConfiguration()
                   {
                       ThingIndexingMode = ThingIndexingMode.REGISTRY
                   }
               });

           _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

           // Wait for the index to be set up - this can take some time
           const int maxRetries = 10;
           const int retryDelaySeconds = 10;

           for (int attempt = 1; attempt <= maxRetries; attempt++)
           {
               try
               {
                   _logger.LogInformation($"Waiting for index to be ready
(attempt {attempt}/{maxRetries})...");
                   await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                   // Try to get the current indexing configuration to see if
it's ready
                   var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                   if
(configResponse.ThingIndexingConfiguration?.ThingIndexingMode ==
ThingIndexingMode.REGISTRY)
                   {
                       // Try the search again
                       var request = new SearchIndexRequest
                       {
                           QueryString = queryString
                       };
```

```
                        var response = await
_amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready
after configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }
```

- Per i dettagli sull'API, consulta la SearchIndexsezione AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
```

```
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

- Per i dettagli sull'API, consulta la SearchIndexsezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Come eseguire una query sull'indice dell'oggetto

L'esempio `search-index` seguente esegue una query sull'indice `AWS_Things` per cercare gli oggetti il cui tipo è `LightBulb`.

```
aws iot search-index \
    --index-name "AWS_Things" \
    --query-string "thingTypeName:LightBulb"
```

Output:

```
{
    "things": [
        {
            "thingName": "MyLightBulb",
```

```
            "thingId": "40da2e73-c6af-406e-b415-15acae538797",
            "thingTypeName": "LightBulb",
            "thingGroupNames": [
                "LightBulbs",
                "DeadBulbs"
            ],
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
            "thingName": "ThirdBulb",
            "thingId": "615c8455-33d5-40e8-95fd-3ee8b24490af",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingId": "6dae0d3f-40c1-476a-80c4-1ed24ba6aa11",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        }
    ]
}
```

Per ulteriori informazioni, consulta Gestione dell'indicizzazione degli oggetti nella Guida per gli sviluppatori di AWS IoT.

- Per i dettagli sull'API, consulta SearchIndex AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, consulta la SearchIndexsezione AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
```

```
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}
```

- Per i dettagli sull'API, SearchIndexconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
```

```
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])
```

- Per i dettagli sull'API, consulta SearchIndex AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`UpdateIndexingConfiguration`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `UpdateIndexingConfiguration`.

C++

SDK per C++

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Update the indexing configuration.
/*!
```

```
  \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
 ignored if not set.
  \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration
 object which is ignored if not set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateIndexingConfiguration(
        const Aws::IoT::Model::ThingIndexingConfiguration
 &thingIndexingConfiguration,
        const Aws::IoT::Model::ThingGroupIndexingConfiguration
 &thingGroupIndexingConfiguration,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }

    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {

 request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
 iotClient.UpdateIndexingConfiguration(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la UpdateIndexingConfigurationsezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Come abilitare l'indicizzazione degli oggetti

L'`update-indexing-configuration`esempio seguente abilita l'indicizzazione degli oggetti per supportare la ricerca nei dati di registro, nei dati shadow e nello stato della connettività degli oggetti utilizzando l'indice AWS_Things.

```
aws iot update-indexing-configuration
    --thing-indexing-
configuration thingIndexingMode=REGISTRY_AND_SHADOW,thingConnectivityIndexingMode=STATUS
```

Questo comando non produce alcun output.

Per ulteriori informazioni, consulta Gestione dell'indicizzazione degli oggetti nella Guida per gli sviluppatori di AWS IoT.

- Per i dettagli sull'API, vedere UpdateIndexingConfigurationin AWS CLI Command Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
```

```
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta UpdateIndexingConfiguration AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`UpdateThing`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `UpdateThing`.

.NET

SDK per .NET (v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
```

```
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Per i dettagli sull'API, consulta la UpdateThingsezione AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la UpdateThingsezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Come associare un oggetto a un tipo di oggetto

L'`update-thing`esempio seguente associa un oggetto nel registro AWS IoT a un tipo di oggetto. Quando crei l'associazione, fornisci valori per gli attributi definiti dal tipo di oggetto.

```
aws iot update-thing \
    --thing-name "MyOtherLightBulb" \
    --thing-type-name "LightBulb" \
    --attribute-payload "{"attributes": {"wattage":"75", "model":"123"}}"
```

Questo comando non produce alcun output. Per visualizzare il risultato, utilizza il comando `describe-thing`.

Per ulteriori informazioni, consulta Tipi di oggetti nella Guida per gli sviluppatori di AWS IoT.

- Per i dettagli sull'API, vedere UpdateThingin AWS CLI Command Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- Per i dettagli sull'API, UpdateThingconsulta AWS SDK for Kotlin API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo AWS IoT con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

AWS IoT quote

Puoi trovare informazioni sulle AWS IoT quote nella Guida AWS generale.

- Per informazioni sulle AWS IoT Core quote, consulta AWS IoT Core Endpoints and Quotas.
- Per informazioni sulle AWS IoT Device Management quote, consulta AWS IoT Device Management Endpoints and Quotas.
- Per informazioni sulle AWS IoT Device Defender quote, consulta AWS IoT Device Defender Endpoints and Quotas.

AWS IoT Core prezzi

Puoi trovare informazioni sui AWS IoT Core prezzi nella pagina AWS Marketing e nel Calcolatore dei AWS prezzi.

- Per controllare le informazioni AWS IoT Core sui prezzi, consulta la sezione AWS IoT Core Prezzi.
- Per stimare il costo della soluzione della tua architettura, consulta il Calcolatore dei prezzi di AWS.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.