aws

Guida per gli sviluppatori

Integrazioni gestite per AWS IoT Device Management

Integrazioni gestite per AWS IoT Device Management: Guida per gli sviluppatori

Table of Contents

A cosa servono le integrazioni gestite? AWS IoT Device Management

Integrazioni gestite per AWS IoT Device Management aiutare i fornitori di soluzioni IoT a unificare il controllo e la gestione dei dispositivi IoT di centinaia di produttori. Puoi utilizzare le integrazioni gestite per automatizzare i flussi di lavoro di configurazione dei dispositivi e supportare l'interoperabilità tra molti dispositivi, indipendentemente dal fornitore del dispositivo o dal protocollo di connettività. Con le integrazioni gestite, puoi utilizzare un'unica interfaccia utente e un set di strumenti APIs per controllare, gestire e utilizzare una vasta gamma di dispositivi.

Argomenti

Regioni supportate

Le integrazioni gestite per AWS IoT Device Management sono supportate nelle seguenti regioni:

- Canada (Centrale)
- Europa (Irlanda)

Sei un utente alle integrazioni gestite per la prima volta?

Se utilizzi per la prima volta le integrazioni gestite, ti consigliamo di iniziare leggendo le seguenti sezioni:

- Configura integrazioni gestite
- Inizia con le integrazioni gestite per AWS IoT Device Management

Panoramica delle integrazioni gestite

L'immagine seguente fornisce una panoramica di alto livello delle integrazioni gestite

Terminologia delle integrazioni gestite

Nell'ambito delle integrazioni gestite, ci sono molti concetti e termini fondamentali da comprendere per gestire le implementazioni dei propri dispositivi. Le sezioni seguenti descrivono questi concetti e termini chiave per fornire una migliore comprensione delle integrazioni gestite.

Terminologia generale delle integrazioni gestite

Un concetto importante da comprendere per le integrazioni gestite è un oggetto gestito rispetto a un oggetto. AWS IoT Core

- AWS IoT Core thing: An AWS IoT Core Thing è un AWS IoT Core costrutto che fornisce la rappresentazione digitale. Ci si aspetta che gli sviluppatori gestiscano le politiche, l'archiviazione dei dati, le regole, le azioni, gli argomenti MQTT e la trasmissione dello stato del dispositivo all'archiviazione dei dati. Per ulteriori informazioni su cos'è An AWS IoT Core Thing, consulta Gestire i dispositivi con AWS IoT.
- Integrazioni gestite Managed Thing: con Managed Thing, forniamo un'astrazione per semplificare le interazioni con i dispositivi e non richiediamo allo sviluppatore di creare elementi come regole, azioni, argomenti MQTT e politiche.

Cloud-to-cloud terminologia

I dispositivi fisici che si integrano con le integrazioni gestite possono provenire da un provider di servizi cloud di terze parti. Per integrare tali dispositivi nelle integrazioni gestite e comunicare con il provider di servizi cloud di terze parti, la terminologia seguente copre alcuni dei concetti chiave che supportano tali flussi di lavoro:

- Cloud-to-cloud Connettore (C2C): un connettore C2C stabilisce una connessione tra le integrazioni gestite e il provider di servizi cloud di terze parti.
- Provider cloud di terze parti: per i dispositivi prodotti e gestiti al di fuori delle integrazioni gestite, un provider cloud terzo consente il controllo di questi dispositivi per l'utente finale e le integrazioni gestite comunicano con il provider cloud di terze parti per vari flussi di lavoro, come i comandi dei dispositivi.

Terminologia dei modelli di dati

Le integrazioni gestite utilizzano modelli di dati per organizzare i dati e le end-to-end comunicazioni tra i dispositivi. La terminologia seguente copre alcuni dei concetti chiave per la comprensione di questi due modelli di dati:

- Dispositivo: un'entità che rappresenta un dispositivo fisico (come un campanello con videocamera) che ha più nodi che lavorano insieme per fornire un set completo di funzionalità.
- Endpoint: un endpoint racchiude una funzionalità autonoma (suoneria, rilevamento del movimento, illuminazione di un campanello con videocamera).
- Capacità: un'entità che rappresenta i componenti necessari per rendere disponibile una funzionalità in un endpoint (pulsante o luce e segnale acustico nella funzione campanello con videocamera).
- Azione: un'entità che rappresenta un'interazione con una funzionalità di un dispositivo (suona il campanello o visualizza chi c'è alla porta).
- Evento: un'entità che rappresenta un evento derivante da una funzionalità di un dispositivo. Un dispositivo può inviare un evento (segnalarlo incident/alarm, an activity from a sensor etc. (e.g. there is knock/ring alla porta).
- Proprietà: un'entità che rappresenta un particolare attributo nello stato del dispositivo (il campanello suona, la luce del portico è accesa, la videocamera sta registrando).
- Modello di dati: il livello dati corrisponde agli elementi di dati e verbi che aiutano a supportare la funzionalità dell'applicazione. L'Applicazione opera su queste strutture di dati quando c'è l'intenzione di interagire con il dispositivo. Per ulteriori informazioni, vedere connectedhomeip sul sito Web. GitHub
- Schema: uno schema è una rappresentazione del modello di dati in formato JSON.

Configura integrazioni gestite

Le seguenti sezioni illustrano la configurazione iniziale per l'utilizzo delle integrazioni gestite per AWS IoT Device Management.

Argomenti

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 Signing in as the root user della 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 Centro identità IAM.

   Per istruzioni, consulta Abilitazione di AWS IAM Identity Center nella Guida per l'utente di AWS IAM Identity Center .
2. In IAM Identity Center, 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 con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

  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.

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni conforme alla best practice dell'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 .

Inizia con le integrazioni gestite per AWS IoT Device Management

Le sezioni seguenti descrivono i passaggi da eseguire per iniziare a utilizzare le integrazioni gestite.

Argomenti

Tipi di dispositivi

Le integrazioni gestite gestiscono molti tipi di dispositivi. Ogni dispositivo rientra in una delle seguenti tre categorie:

- Dispositivi con connessione diretta: questo tipo di dispositivo si connette direttamente a un endpoint di integrazioni gestite. In genere, questi dispositivi sono costruiti e gestiti dai produttori di dispositivi che includono l'SDK per dispositivi finali con integrazioni gestite per la connettività diretta.
- Dispositivi connessi all'hub: questi dispositivi si connettono alle integrazioni gestite tramite un hub che esegue l'SDK Hub per le integrazioni gestite, che gestisce le funzioni di rilevamento, onboarding e controllo dei dispositivi. Gli utenti finali possono effettuare l'onboarding di questi dispositivi tramite la pressione di un pulsante o la scansione di codici a barre.

  I seguenti due flussi di lavoro sono supportati per l'onboarding di un dispositivo connesso all'hub:
  - Premere un pulsante avviato dall'utente finale per avviare l'individuazione del dispositivo
  - Scansione basata su codici a barre per eseguire l'associazione del dispositivo
- Cloud-to-cloud Dispositivi (C2C): si tratta di dispositivi progettati e gestiti da fornitori che gestiscono la propria infrastruttura cloud e applicazioni mobili di marca per il controllo dei dispositivi. I clienti delle integrazioni gestite possono accedere a un catalogo di connettori C2C predefiniti o crearne di propri, per sviluppare soluzioni IoT che funzionano con più cloud di fornitori di terze parti tramite un'interfaccia unificata.

  Quando l'utente finale accende un dispositivo C2C per la prima volta, deve affidarsi al rispettivo provider di servizi cloud di terze parti per le integrazioni gestite al fine di ottenere le funzionalità e

i metadati del dispositivo. Dopo aver completato il flusso di lavoro di provisioning, le integrazioni gestite possono comunicare con il dispositivo cloud e il provider cloud terzo per conto dell'utente finale.

> **Note**
>
> Un hub non è un tipo di dispositivo specifico come elencato sopra. Il suo scopo è svolgere il ruolo di controller dei dispositivi domestici intelligenti e facilitare la connessione tra integrazioni gestite e provider di cloud di terze parti. Può svolgere il ruolo sia come tipo di dispositivo, come elencato sopra, sia come hub.

Configura la chiave di crittografia

La sicurezza è di fondamentale importanza per i dati instradati tra l'utente finale, le integrazioni gestite e i cloud di terze parti. Uno dei metodi che supportiamo per proteggere i dati del dispositivo è la crittografia che utilizza una chiave di end-to-end crittografia sicura per il routing dei dati.

In qualità di cliente di integrazioni gestite, hai le seguenti due opzioni per l'utilizzo delle chiavi di crittografia:

- Utilizza la chiave di crittografia gestita dalle integrazioni gestite di default.
- Fornisci un file che hai creato AWS KMS key .

Per ulteriori informazioni sul servizio AWS KMS, consulta Key management service (KMS)

La chiamata all'PutDefaultEncryptionConfigurationAPI nella Managed Integrations API Reference Guide ti consente di accedere all'aggiornamento dell'opzione di chiave di crittografia che desideri utilizzare. Per impostazione predefinita, le integrazioni gestite utilizzano la chiave di crittografia gestita predefinita delle integrazioni gestite. Puoi aggiornare la configurazione della chiave di crittografia in qualsiasi momento utilizzando l'PutDefaultEncryptionConfigurationAPI.

Inoltre, la chiamata al comando GetDefaultEncryptionConfigurationAPI restituisce informazioni sulla configurazione di crittografia per l' AWS account nella regione predefinita o specificata.

Tecniche di onboarding

Di seguito sono elencati i tipi di onboarding:

Onboarding dei dispositivi con connessione diretta

Consulta Provisionario la procedura per l'onboarding di un dispositivo con connessione diretta.

Inserimento nell'hub

Consulta Integra i tuoi hub verso integrazioni gestite i passaggi per l'onboarding dell'hub.

Onboarding di dispositivi connessi all'hub

Consulta la procedura Incorpora i dispositivi e gestiscili nell'hub per effettuare l'onboard di un dispositivo connesso all'hub.

Cloud-to-cloud onboarding del dispositivo

Consulta i passaggi Usa un connettore C2C (Cloud-to-Cloud) per effettuare l'onboarding di un dispositivo cloud da un fornitore di servizi cloud di terze parti alle integrazioni gestite.

Provisioning dei dispositivi

Il provisioning dei dispositivi facilita il processo di onboarding dei dispositivi, supervisiona l'intero ciclo di vita dei dispositivi e stabilisce un archivio centralizzato per le informazioni sui dispositivi accessibile ad altri aspetti delle integrazioni gestite. Le integrazioni gestite forniscono un'interfaccia unificata per la gestione di vari tipi di dispositivi, adattandosi ai dispositivi dei clienti proprietari collegati direttamente tramite un kit di sviluppo software (SDK) o dispositivi (COTS) collegati indirettamente tramite un dispositivo hub. commercial-off-the-shelf

Ogni dispositivo, indipendentemente dal tipo di dispositivo, nelle integrazioni gestite ha un identificatore univoco globale chiamato a. `managedThingId` Questo identificatore viene utilizzato per l'onboarding e la gestione del dispositivo per l'intero ciclo di vita del dispositivo. È completamente gestito da integrazioni gestite ed è unico per quel dispositivo specifico in tutte le integrazioni gestite. Regioni AWS Quando un dispositivo viene inizialmente aggiunto alle integrazioni gestite, questo identificatore viene creato e allegato all'elemento gestito nelle integrazioni gestite. Un oggetto gestito è una rappresentazione digitale del dispositivo fisico all'interno delle integrazioni gestite per rispecchiare tutti i metadati del dispositivo fisico. Per i dispositivi di terze parti, possono avere un proprio identificatore univoco separato specifico per il cloud di terze parti, oltre a quello `managedThingId` archiviato nelle integrazioni gestite che rappresentano il dispositivo fisico.

I dispositivi oggetto di provisioning possono avere stati diversi a seconda della fase del flusso di onboarding in cui si trovano. L'elenco seguente descrive ogni stato di approvvigionamento:

- ATTIVATO: Il dispositivo è stato trovato e il comando e il controllo sono disponibili.
- SCOPERTO: Il dispositivo è stato trovato ma il comando e il controllo non sono ancora disponibili.
- NON ASSOCIATO: L'oggetto gestito è stato creato ma richiede ulteriori azioni per essere scoperto. Non è raggiungibile dai controller (hub) Cloud AWS o dai controller (hub) di AWS IoT Managed Integrations
- PRE_ASSOCIATED: L'oggetto gestito è stato creato ed è pronto per il rilevamento automatico una volta acceso o connesso. Non è raggiungibile dai controller (hub) Cloud AWS o dai controller (hub) di AWS IoT Managed Integrations.
- DELETE_IN_PROGRESS: processo di cancellazione asincrono avviato.
- ELIMINATO: il dispositivo è stato eliminato da. Cloud AWS
- ISOLATO: oggetto gestito precedentemente scoperto o attivato che non è più raggiungibile. Ad esempio, un dispositivo per un cloud di terze parti le cui associazioni di connettori sono state tutte eliminate.

Il seguente flusso di onboarding serve a fornire all'hub integrazioni gestite:

Integra i tuoi hub verso integrazioni gestite: configura il core provisioner e i plug-in specifici del protocollo che interagiscono per gestire l'autenticazione, la comunicazione e la configurazione del dispositivo.

I seguenti flussi di onboarding sono forniti per fornire integrazioni gestite ai dispositivi collegati all'hub:

- Configurazione semplice (SS): l'utente finale accende il dispositivo IoT e ne scansiona il codice QR utilizzando l'applicazione del produttore del dispositivo. Il dispositivo viene quindi registrato nel cloud delle integrazioni gestite e si connette all'hub IoT.
- Configurazione Zero-touch (ZTS): Il dispositivo è pre-associato a monte nella catena di fornitura. Ad esempio, anziché essere gli utenti finali a scansionare il codice QR del dispositivo, questo passaggio viene completato in precedenza per collegare preventivamente il dispositivo agli account dei clienti.
- Configurazione guidata dall'utente (UGS): L'utente finale accende il dispositivo e segue i passaggi interattivi per integrarlo nelle integrazioni gestite. Ciò potrebbe includere la pressione di un pulsante sull'hub IoT, l'utilizzo di un'app del produttore del dispositivo o la pressione di pulsanti sia sull'hub che sul dispositivo. È possibile utilizzare questo metodo se la configurazione semplice fallisce.

> **Note**
>
> Il flusso di lavoro di provisioning dei dispositivi nelle integrazioni gestite è indipendente dai requisiti di onboarding di un dispositivo. Le integrazioni gestite forniscono un'interfaccia utente semplificata per l'onboarding e la gestione di un dispositivo, indipendentemente dal tipo di dispositivo o dal protocollo del dispositivo.

Ciclo di vita del dispositivo e del profilo del dispositivo

La gestione del ciclo di vita dei dispositivi e dei profili dei dispositivi garantisce la sicurezza e l'efficienza del parco dispositivi.

Argomenti

Dispositivo

Durante l'onboarding iniziale, le integrazioni gestite creano un gemello digitale del dispositivo fisico chiamato Managed Thing. Il Managed Thing dispone di un `managedThingID` identificatore univoco globale per identificare il dispositivo nelle integrazioni gestite in tutte le regioni. Il dispositivo si associa all'hub locale durante il provisioning per la comunicazione in tempo reale con integrazioni gestite o a un cloud di terze parti per dispositivi di terze parti. Un dispositivo è anche associato a un proprietario, come identificato dal `owner` parametro reso pubblico APIs per un oggetto gestito come. `GetManagedThing` Il dispositivo è collegato al profilo del dispositivo corrispondente in base al tipo di dispositivo.

> **Note**
>
> Un dispositivo fisico può avere più record se viene fornito più volte a clienti diversi.

Il ciclo di vita del dispositivo inizia con la creazione dell'oggetto gestito nelle integrazioni gestite utilizzando l'`CreateManagedThing`API e termina quando il cliente elimina l'oggetto gestito utilizzando l'API. `DeleteManagedThing` Il ciclo di vita di un dispositivo è gestito dal pubblico seguente: APIs

- `CreateManagedThing`
- `ListManagedThings`
- `GetManagedThing`
- `UpdateManagedThing`
- `DeleteManagedThing`

Profilo del dispositivo

Un profilo di dispositivo rappresenta un tipo specifico di dispositivo, ad esempio una lampadina o un campanello. È associato a un produttore e contiene le funzionalità del dispositivo. Il profilo del dispositivo memorizza i materiali di autenticazione necessari per le richieste di configurazione della connettività del dispositivo con integrazioni gestite. I materiali di autenticazione utilizzati sono il codice a barre del dispositivo.

Durante il processo di produzione del dispositivo, il produttore può registrare i profili dei dispositivi con integrazioni gestite. Ciò consente al produttore di ottenere i materiali necessari per i dispositivi dalle integrazioni gestite durante i flussi di lavoro di onboarding e provisioning. I metadati del profilo del dispositivo vengono archiviati sul dispositivo fisico o stampati sull'etichetta del dispositivo. Il ciclo di vita del profilo del dispositivo termina quando il produttore lo elimina nelle integrazioni gestite.

Modelli di dati

Un modello di dati rappresenta la gerarchia organizzativa del modo in cui i dati sono organizzati all'interno di un sistema. Inoltre, supporta la end-to-end comunicazione attraverso l'intera implementazione del dispositivo. Per le integrazioni gestite, vengono utilizzati due modelli di dati. Il modello di dati delle integrazioni gestite e l' AWS implementazione del Matter Data Model. Presentano delle somiglianze, ma presentano anche delle sottili differenze, descritte nei seguenti argomenti.

Per i dispositivi di terze parti, entrambi i modelli di dati vengono utilizzati per la comunicazione tra l'utente finale, le integrazioni gestite e il provider di servizi cloud di terze parti. Per tradurre messaggi quali comandi ed eventi del dispositivo dai due modelli di dati, viene sfruttata la funzionalità Cloud-to-Cloud Connector.

Argomenti

Modello di dati di integrazioni gestite

Il modello di dati delle integrazioni gestite gestisce tutte le comunicazioni tra l'utente finale e le integrazioni gestite.

Gerarchia dei dispositivi

Gli elementi `endpoint` e `capability` dati vengono utilizzati per descrivere un dispositivo nel modello di dati delle integrazioni gestite.

`endpoint`

`endpoint`Rappresenta le interfacce o i servizi logici offerti dalla funzionalità.

```
{
    "endpointId": { "type":"string" },
    "capabilities": Capability[]
}
```

Capability

`capability`Rappresenta le funzionalità del dispositivo.

```
{
    "$id": "string",                // Schema identifier (e.g. /schema-versions/
capability/matter.OnOff@1.4)
    "name": "string",               // Human readable name
    "version": "string",            // e.g. 1.0
    "properties": Property[],
    "actions": Action[],
    "events": Event[]
}
```

Per l'elemento `capability` dati, ci sono tre elementi che lo compongono: `property`, `action`, e`event`. Possono essere utilizzati per interagire e monitorare il dispositivo.

- Proprietà: Stati mantenuti dal dispositivo, ad esempio l'attributo del livello di luminosità corrente di una luce dimmerabile.
  - ```
    {
        "name":                      // Property Name is outside of Property Entity
        "value": Value,              // value represented in any type e.g. 4, "A", []
        "lastChangedAt": Timestamp   // ISO 8601 Timestamp upto milliseconds yyyy-MM-
    ddTHH:mm:ss.ssssssZ
        "mutable": boolean,
        "retrievable": boolean,
        "reportable": boolean

    }
    ```
- Azione: Attività che possono essere eseguite, come bloccare una porta su una serratura. Le azioni possono generare risposte e risultati.
  - ```
    {
        "name": { "$ref": "/schema-versions/definition/aws.name@1.0" }, //required
        "parameters": Map<String name, JSONNode value>,
        "responseCode": HTTPResponseCode,
        "errors": {
            "code": "string",
            "message": "string"
        }
    }
    ```

- Evento: essenzialmente una registrazione delle transizioni di stato passate. Sebbene `property` rappresentino gli stati attuali, gli eventi sono un diario del passato e includono un contatore che aumenta in modo monotono, un timestamp e una priorità. Consentono di catturare le transizioni di stato e la modellazione dei dati che non è facilmente realizzabile. `property`
  - ```
    {
        "name": { "$ref": "/schema-versions/definition/aws.name@1.0" },        //
    required
        "parameters": Map<String name, JSONNode value>
    }
    ```

AWS implementazione del modello di dati Matter

L' AWS implementazione del Matter Data Model gestisce tutte le comunicazioni tra le integrazioni gestite e i provider cloud di terze parti.

Per ulteriori informazioni, consulta Matter Data Model: Developer Resources.

Gerarchia dei dispositivi

Esistono due elementi di dati utilizzati per descrivere un dispositivo: `endpoint` `ecluster`.

`endpoint`

`endpoint`Rappresenta le interfacce o i servizi logici offerti dalla funzionalità.

```
{
    "id": { "type":"string"},
    "clusters": Cluster[]
}
```

`cluster`

`cluster`Rappresenta le funzionalità del dispositivo.

```
{
    "id": "hexadecimalString",
   "revision": "string"        // optional
    "attributes": AttributeMap<String attributeId, JSONNode>,
   "commands": CommandMap<String commandId, JSONNode>,
   "events": EventMap<String eventId, JsonNode>
```
```

```
}
```

Per l'elemento `cluster` dati, ci sono tre elementi che lo compongono: `attributecommand`, `eevent`. Possono essere utilizzati per interagire e monitorare il dispositivo.

- Attributo: stati mantenuti dal dispositivo, ad esempio l'attributo del livello di luminosità corrente di una luce dimmerabile.
  - ```
    {
        "id" (hexadecimalString): (JsonNode) value
    }
    ```
- Comando: attività che possono essere eseguite, come bloccare una porta su una serratura. I comandi possono generare risposte e risultati.
  - ```
    "id": {
        "fieldId": "fieldValue",
        ...
        "responseCode": HTTPResponseCode,
        "errors": {
            "code": "string",
            "message": "string"
        }
    }
    ```
- Evento: essenzialmente una registrazione delle transizioni di stato passate. Sebbene `attributes` rappresentino gli stati attuali, gli eventi sono un diario del passato e includono un contatore che aumenta in modo monotono, un timestamp e una priorità. Consentono di catturare le transizioni di stato e la modellazione dei dati che non è facilmente realizzabile. `attributes`
  - ```
    "id": {
        "fieldId": "fieldValue",
        ...
    }
    ```

Schemi di modelli di dati

Le integrazioni gestite supportano due tipi di schema: funzionalità e definizione del tipo. Se stai creando un modello di dati personalizzato, utilizzi un documento dello schema JSON per definire entrambi i tipi di schema. Ogni documento dello schema ha un limite di 50.000 caratteri.
    ```

Schema di capacità

Una funzionalità è un elemento fondamentale che rappresenta funzionalità specifiche all'interno di un endpoint. Grazie alle funzionalità, è possibile modellare gli stati e i comportamenti dei dispositivi utilizzando proprietà, azioni ed eventi. Le proprietà consentono di modellare gli attributi di stato del dispositivo in modo flessibile con qualsiasi tipo di dati dichiarativo. Le azioni e gli eventi modellano il comportamento del dispositivo, inclusi i comandi che può eseguire e i segnali che può segnalare.

Di seguito viene illustrata una struttura di alto livello di uno schema di funzionalità.

```
Capability
  |
  |-- Action
  |-- Event
  |-- Property
```

Azione

Un'entità che rappresenta un'interazione con una funzionalità di un dispositivo. Ad esempio, suona il campanello o guarda chi c'è alla porta.

Evento

Un'entità che rappresenta un evento da una funzionalità del dispositivo. Un dispositivo può inviare un evento per segnalare un incidente, un allarme o un'attività tramite un sensore, ad esempio quando bussa alla porta.

Proprietà

Un'entità che rappresenta un particolare attributo nello stato del dispositivo. Ad esempio, suona un campanello o la luce del portico è accesa

Ogni funzionalità include un identificatore univoco con namespace, informazioni sulla versione e una descrizione del suo scopo. Il documento dello schema utilizza il controllo delle versioni semantiche per mantenere la compatibilità con le versioni precedenti e abilitare nuove funzionalità.

Per ulteriori informazioni, consulta Schema per le definizioni delle funzionalità.

Schema di definizione del tipo

Una definizione di tipo è un tipo di dati strutturato dichiarativo che consente la riutilizzabilità e la componibilità. Definisce come le informazioni devono essere formattate e vincolate. Usa le definizioni dei tipi per creare formati di dati standardizzati per la tua soluzione IoT.

Ogni definizione di tipo include:

- Un identificatore univoco con namespace
- Titolo
- Descrizione
- Proprietà che definiscono la formattazione e i vincoli dei dati

I tipi possono essere primitive semplici, come numeri interi o stringhe con limiti definiti, oppure strutture complesse come enumerazioni o oggetti personalizzati con più campi. Le definizioni dei tipi utilizzano la sintassi dello schema JSON per specificare i vincoli, tra cui valori minimi e massimi, lunghezze delle stringhe e modelli consentiti.

Per ulteriori informazioni, consulta Schema per le definizioni dei tipi.

Schema per le definizioni delle funzionalità

Una funzionalità è documentata utilizzando un documento JSON dichiarativo che fornisce un contratto chiaro su come la funzionalità dovrebbe funzionare all'interno del sistema.

Per una funzionalità, gli elementi obbligatori sono `$idname`,`extrinsicId`, `extrinsicVersion` e almeno un elemento in almeno una delle seguenti sezioni:

- `properties`
- `actions`
- `events`

Gli elementi opzionali di una funzionalità sono `$reftitle`,`description`,`version`, `$defs` e`extrinsicProperties`. Per una funzionalità, `$ref` deve fare riferimento a`aws.capability`.

Le sezioni seguenti descrivono in dettaglio lo schema utilizzato per le definizioni delle funzionalità.

$id (obbligatorio)

L'elemento $id identifica la definizione dello schema. Deve seguire questa struttura:

- Inizia con il prefisso `/schema-versions/` URI
- Includi il tipo di `capability` schema
- Usa una barra (`/`) come separatore di percorso URI
- Includi l'identità dello schema, con frammenti separati da punti () `.`
- Usa il `@` carattere per separare l'ID e la versione dello schema
- Termina con la versione semver, usando period (`.`) per separare i frammenti di versione

L'identità dello schema deve iniziare con uno spazio dei nomi radice lungo 3-12 caratteri, seguito da uno spazio dei nomi secondario e da un nome opzionali.

La versione semver include una versione MAJOR (fino a 3 cifre), una versione MINOR (fino a 3 cifre) e una versione PATCH opzionale (fino a 4 cifre).

> **Note**
>
> Non è possibile utilizzare i namespace riservati o `aws` `matter`

Example Esempio $id

```
/schema-version/capability/aws.Recording@1.0
```

$ref

L'`$ref`elemento fa riferimento a una funzionalità esistente all'interno del sistema. Segue gli stessi vincoli dell'`$id`elemento.

> **Note**
>
> È necessario che esista una definizione o una funzionalità del tipo con il valore fornito nel `$ref` file.

Example Esempio $ref

```
/schema-version/definition/aws.capability@1.0
```

nome (obbligatorio)

L'elemento name è una stringa che rappresenta il nome dell'entità nel documento dello schema. Spesso contiene abbreviazioni e deve seguire queste regole:

- Contiene solo caratteri alfanumerici, punti (.), barre (/), trattini (-) e spazi
- Inizia con una lettera
- Massimo 64 caratteri

L'elemento name viene utilizzato nell'interfaccia utente e nella documentazione della console di Amazon Web Services.

Example Nomi di esempio

```
Door Lock
On/Off
Wi-Fi Network Management
PM2.5 Concentration Measurement
RTCSessionController
Energy EVSE
```

titolo

L'elemento title è una stringa descrittiva per l'entità rappresentata dal documento dello schema. Può contenere qualsiasi carattere e viene utilizzato nella documentazione. La lunghezza massima per un titolo di funzionalità è di 256 caratteri.

Example Titoli di esempio

```
Real-time Communication (RTC) Session Controller
Energy EVSE Capability
```

description

L'`description`elemento fornisce una spiegazione dettagliata dell'entità rappresentata dal documento dello schema. Può contenere qualsiasi carattere e viene utilizzato nella documentazione. La lunghezza massima per una descrizione delle funzionalità è di 2048 caratteri

Example Descrizione di esempio

```
Electric Vehicle Supply Equipment (EVSE) is equipment used to charge an Electric
 Vehicle (EV) or Plug-In Hybrid Electric Vehicle.
           This capability provides an interface to the functionality of Electric
 Vehicle Supply Equipment (EVSE) management.
```

version

L'elemento `version` è facoltativo. È una stringa che rappresenta la versione del documento dello schema. Vengono applicati i seguenti vincoli:

- Utilizza il formato semver, con i seguenti frammenti di versione separati da `.` (punti).
  - `MAJOR`versione, massimo 3 cifre
  - `MINOR`versione, massimo 3 cifre
  - `PATCH`versione (opzionale), massimo 4 cifre
- La lunghezza può essere compresa tra 3 e 12 caratteri.

Example Versioni di esempio

```
1.0
```

```
1.12
```

```
1.4.1
```

Utilizzo delle versioni con funzionalità

Una funzionalità è un'entità con versioni immutabili. Qualsiasi modifica dovrebbe creare una nuova versione. Il sistema utilizza il controllo delle versioni semantiche con il formato MAJOR.MINOR.PATCH, dove:

- La versione MAJOR aumenta quando si apportano modifiche all'API incompatibili con le versioni precedenti
- La versione MINOR aumenta quando si aggiungono funzionalità in modo retrocompatibile
- La versione PATCH aumenta quando si apportano aggiunte minori e senza impatto sulla funzionalità.

Le funzionalità derivate dai cluster Matter sono basate sulla versione 1.4 e ogni versione di Matter dovrebbe essere importata nel sistema. Poiché la versione Matter utilizza i livelli MAJOR e MINOR di semver, le integrazioni gestite possono utilizzare solo le versioni PATCH.

Quando aggiungi versioni PATCH per Matter, assicurati di tenere conto del fatto che Matter utilizza revisioni sequenziali. Tutte le versioni PATCH devono essere conformi alla revisione documentata nella specifica Matter e devono essere retrocompatibili.

Per risolvere eventuali problemi di incompatibilità con le versioni precedenti, è necessario collaborare con la Connectivity Standards Alliance (CSA) per risolverli nelle specifiche e rilasciare una nuova revisione.

AWS-le funzionalità gestite sono state rilasciate con una versione iniziale di. `1.0` Con queste, è possibile utilizzare tutti e tre i livelli di versione.

Extrinsic Version (obbligatoria)

Questa è una stringa che rappresenta una versione gestita all'esterno del sistema. AWS IoT Per le funzionalità di Matter, esegue il `extrinsicVersion` mapping a `revision`

È rappresentato come un valore intero con stringhe e la lunghezza può essere compresa tra 1 e 10 cifre numeriche.

Example Versioni di esempio

```
7
```

```
1567
```

ExtrinsICID (obbligatorio)

L'`extrinsicId`elemento rappresenta un identificatore gestito all'esterno del sistema IoT di Amazon Web Services. Per le funzionalità di Matter, viene mappato a `clusterId` `attributeId`,`commandId`,`eventId`, o `fieldId`, a seconda del contesto.

`extrinsicId`Può essere un numero intero decimale con stringhe (1-10 cifre) o un intero esadecimale con stringhe (prefisso 0x o 0X, seguito da 1-8 cifre esadecimali).

> **Note**
>
> Per, l' AWS ID fornitore (VID) è 0x1577 e per Matter è 0. Il sistema garantisce che gli schemi personalizzati non utilizzino queste funzionalità riservate. VIDs

Example Esempio: ExtrinsICIDS

```
0018
0x001A
0x15771002
```

$defs

La `$defs` sezione è una mappa di sottoschemi a cui è possibile fare riferimento all'interno del documento dello schema, come consentito dallo schema JSON. In questa mappa, la chiave viene utilizzata nelle definizioni di riferimento locali e il valore fornisce lo schema JSON.

> **Note**
>
> Il sistema impone solo che `$defs` sia una mappa valida e che ogni sottoschema sia uno schema JSON valido. Non viene applicata alcuna regola aggiuntiva.

Segui questi vincoli quando lavori con le definizioni:

- Usa solo caratteri compatibili con gli URI nei nomi delle definizioni
- Assicurati che ogni valore sia un sottoschema valido
- Includi un numero qualsiasi di sottoschemi che rientrano nei limiti di dimensione del documento dello schema

Proprietà estrinseche

L'`extrinsicProperties`elemento contiene un insieme di proprietà definite in un sistema esterno ma mantenute all'interno del modello di dati. Per quanto riguarda le funzionalità di Matter, esegue il mapping a diversi elementi non modellati o parzialmente modellati all'interno del cluster, dell'attributo, del comando o dell'evento ZCL.

Le proprietà estrinseche devono rispettare questi vincoli:

- I nomi delle proprietà devono essere alfanumerici senza spazi o caratteri speciali
- I valori delle proprietà possono essere qualsiasi valore dello schema JSON
- Massimo 20 proprietà

Il sistema supporta diverse funzionalità`extrinsicProperties`, tra cui `accessapiMaturity`,`cli`,`cliFunctionName`, e altre ancora. Queste proprietà facilitano le trasformazioni tra ACL e modelli di dati AWS (e viceversa).

> **Note**
>
> Le proprietà estrinseche sono supportate per gli elementi `actionevent`,`property`, e `struct` fields di una funzionalità, ma non per la capacità o il cluster stesso.

Proprietà estrinseche supportate dal sistema

Il sistema tiene traccia dei seguenti attributi di cluster, attributo, comando o evento non modellati o parzialmente modellati, come `extrinsicProperties` durante le trasformazioni da o verso ZCL:

`access`

Ogni oggetto di accesso contiene quanto segue:

- op- Operazione modellata come e `enum` con valori:`read`,`write`, o `invoke`
- `privilege`- Privilegio modellato come e `enum` con valori:`view`,,,`proxy_view`, o `operate` `manage` `administer`
- `role`- Stringa illimitata che rappresenta un ruolo di operatore

`apiMaturity`

Una stringa semplice illimitata che rappresenta il livello di maturità. Questo è modellato in ZCL come e `enum` con valori:,,, o `stable` `provisional` `internal` `deprecated`

`side`

Modellato come enum con valori:,, e `either` `server` `client`

Proprietà booleane

Le seguenti proprietà sono bandiere booleane:

- `isFabricScoped`
- `isFabricSensitive`
- `mustUseAtomicWrite`
- `mustUseTimedInvoke`

Proprietà delle stringhe

Le seguenti proprietà sono rappresentate come stringhe illimitate:

- `cli`
- `cliFunctionName`
- `functionName`
- `group`
- `introducedIn`
- `manufacturerCode`
- `noDefaultImplementation`
- `presentIf`
- `priority`
- `removedIn`
- `reportableChange`
- `reportMinInterval`
- `reportMaxInterval`
- `restriction`
- `storage`

Considerazioni sulla trasformazione

Per le trasformazioni ZCL, `extrinsicProperties` vengono memorizzate in una mappa senza elaborazione. Gli schemi personalizzati che utilizzano il rilevamento non subiscono la trasformazione ZCL. Tuttavia, se si prevede di implementare trasformazioni ZCL per schemi personalizzati in futuro, è necessario modellare tutti i tipi di stringa semplice illimitati `extrinsicProperties` e definire vincoli come enumerazioni, pattern (regex) e lunghezza. Questa preparazione garantisce la corretta gestione di queste proprietà durante la trasformazione.

Al contrario, le trasformazioni AWS da connettore a connettore non `extrinsicProperties` sono affatto incluse, poiché questi dettagli non sono richiesti nel formato del connettore.

Proprietà

Le proprietà rappresentano gli stati della funzionalità gestiti dal dispositivo. Ogni stato è definito come una coppia chiave-valore, in cui la chiave descrive il nome dello stato e il valore descrive la definizione dello stato.

Quando lavorate con le proprietà, seguite questi vincoli:

- Utilizzate solo caratteri alfanumerici nei nomi delle proprietà, senza spazi o caratteri speciali
- Includi un numero qualsiasi di proprietà che rientrano nei limiti di dimensione del documento dello schema

Utilizzo delle proprietà

Una proprietà all'interno di una funzionalità è un elemento fondamentale che rappresenta uno stato specifico di un dispositivo basato su integrazioni gestite. Rappresenta la condizione o la configurazione corrente del dispositivo. Standardizzando il modo in cui queste proprietà sono definite e strutturate, i sistemi di casa intelligente garantiscono che i dispositivi di diversi produttori possano comunicare in modo efficace, creando un'esperienza fluida e interoperabile.

Per una proprietà di capacità, gli elementi obbligatori sono e. `extrinsicId` `value` Gli elementi opzionali di una proprietà di capacità sono `descriptionretrievable`,`mutable`, `reportable` e`extrinsicProperties`.

Valore

Una struttura illimitata che consente ai costruttori di inserire qualsiasi vincolo conforme allo schema JSON per definire il tipo di dati di questa proprietà.

Quando definisci i valori, segui questi vincoli:

- Per i tipi semplici, usa `type` e qualsiasi altro vincolo nativo dello schema JSON come o `maxLength` `maximum`
- Per i tipi compositi, usa, o. `oneOf` `allOf` `anyOf` Il sistema non supporta la `not` parola chiave
- Per fare riferimento a qualsiasi tipo globale, usala `$ref` con un riferimento rilevabile valido
- Per l'annullabilità, segui la definizione dello schema del tipo OpenAPI fornendo all'attributo nullable un flag booleano (se null è un valore consentito) `true`

Esempio:

```
{
    "$ref": "/schema-versions/definition/matter.uint16@1.4",
    "nullable": true,
    "maximum": 4096
}
```

Recuperabile

Un booleano che descrive se lo stato è leggibile o meno.

L'aspetto della leggibilità dello stato è rimandato all'implementazione della funzionalità da parte del dispositivo. Il dispositivo decide se un determinato stato è leggibile o meno. Questo aspetto dello stato non è ancora supportato per essere riportato nel rapporto sulla capacità e quindi non è applicato all'interno del sistema.

Esempio: `true` o `false`

Mutable

Un booleano che descrive se lo stato è scrivibile o meno.

L'aspetto della scrivibilità dello stato è rimandato all'implementazione della funzionalità da parte del dispositivo. Il dispositivo decide se un determinato stato è scrivibile o meno. Questo aspetto dello stato non è ancora supportato per essere riportato nel rapporto sulla capacità e quindi non è applicato all'interno del sistema.

Esempio: `true` o `false`

Segnalabile

Un booleano che descrive se lo stato viene segnalato dal dispositivo quando c'è un cambiamento nello stato.

L'aspetto relativo alla rendicontabilità dello stato è rimandato all'implementazione della funzionalità da parte del dispositivo. Il dispositivo decide se un determinato stato è segnalabile o meno. Questo aspetto dello stato non è ancora supportato per essere riportato nel rapporto sulla capacità e quindi non è applicato all'interno del sistema.

Esempio: `true` o `false`

Azioni

Le azioni sono operazioni gestite dallo schema che seguono un modello di richiesta-risposta. Ogni azione rappresenta un'operazione implementata dal dispositivo.

Segui questi vincoli durante l'implementazione delle azioni:

- Includi solo azioni univoche nell'array delle azioni
- Includi un numero qualsiasi di azioni che rientrano nei limiti di dimensione del documento dello schema

Utilizzo di operazioni

Un'azione è un modo standardizzato per interagire e controllare le funzionalità dei dispositivi in un sistema di integrazione gestito. Rappresenta un comando o un'operazione specifico che può essere eseguito su un dispositivo, completo di un formato strutturato per modellare qualsiasi parametro di richiesta o risposta necessario. Queste azioni fungono da ponte tra le intenzioni degli utenti e le operazioni del dispositivo, consentendo un controllo coerente e affidabile su diversi tipi di dispositivi intelligenti.

Per un'azione, gli elementi obbligatori sono `name` e`extrinsicId`. Gli elementi opzionali sono `description`e`extrinsicProperties`, `request` e`response`.

Descrizione

La descrizione ha un limite di lunghezza massimo di 1536 caratteri.

Richiesta

La sezione di richiesta è facoltativa e può essere omessa se non ci sono parametri di richiesta. Se omessa, il sistema supporta l'invio di una richiesta senza alcun payload utilizzando semplicemente il nome di. `Action` Viene utilizzato per azioni semplici, come accendere o spegnere una luce.

Le azioni complesse richiedono parametri aggiuntivi. Ad esempio, una richiesta di streaming delle riprese della telecamera potrebbe includere parametri relativi al protocollo di streaming da utilizzare o se inviare lo streaming a un dispositivo di visualizzazione specifico.

Per una richiesta di azione, l'elemento obbligatorio è`parameters`. Gli elementi opzionali sono `descriptionextrinsicId`, e`extrinsicProperties`.

Descrizione della richiesta

La descrizione segue lo stesso formato della sezione 3.5, con una lunghezza massima di 2048 caratteri.

Risposta

Nelle integrazioni gestite, per ogni richiesta di azione inviata tramite l'SendManagedThingCommandAPI, la richiesta raggiunge il dispositivo e prevede una risposta asincrona. La risposta all'azione definisce la struttura di questa risposta.

Per una richiesta di azione, l'elemento obbligatorio è`parameters`. Gli elementi opzionali sono `namedescription`,`extrinsicId`,`extrinsicProperties`, `errors` e`responseCode`.

Descrizione della risposta

La descrizione segue lo stesso description formato e ha una lunghezza massima di 2048 caratteri.

Nome della risposta

Il nome segue lo stesso formatonome (obbligatorio), con questi dettagli aggiuntivi:

- Il nome convenzionale di una risposta è derivato dall'aggiunta `Response` al nome dell'azione.
- Se desideri utilizzare un nome diverso, puoi fornirlo in questo `name` elemento. Se nella risposta `name` viene fornito a, questo valore ha la precedenza maggiore rispetto al nome convenzionale.

Errori

Una matrice illimitata di messaggi univoci fornita nella risposta, in caso di errori durante l'elaborazione della richiesta.

Vincoli:

- Un elemento del messaggio viene dichiarato come oggetto JSON con i seguenti campi:
  - `code`: Una stringa contenente caratteri alfanumerici e _ (caratteri di sottolineatura), con una lunghezza compresa tra 1 e 64 caratteri
  - `message`: valore di stringa illimitato

Example Esempio di messaggio di errore

```
"errors": [
   {
      "code": "AD_001",
      "message": "Unable to receive signal from the sensor. Please check connection
 with the sensor."
   }
]
```

Codice di risposta

Un codice intero che mostra come è stata gestita la richiesta. È consigliabile che il codice del dispositivo restituisca un codice utilizzando la specifica del codice di stato della risposta del server HTTP per consentire l'uniformità all'interno del sistema.

Vincolo: un valore intero compreso tra 100 e 599.

Parametri di richiesta o risposta

La sezione dei parametri è definita come una mappa di coppie di nomi e sottoschemi. È possibile definire un numero qualsiasi di parametri all'interno dei parametri della richiesta, se rientrano nel documento dello schema.

I nomi dei parametri possono contenere solo caratteri alfanumerici. Gli spazi o altri caratteri non sono consentiti.

campo dei parametri

Gli elementi obbligatori in a `parameter` sono `extrinsicId` `evalue`. Gli elementi opzionali sono `description` `eextrinsicProperties`.

L'elemento di descrizione segue lo stesso formato didescription, con una lunghezza massima di 1024 caratteri.

`extrinsicId`e sostituisce **`extrinsicProperties`**

`extrinsicId`de `extrinsicProperties` seguono lo stesso formato ExtrinsICID (obbligatorio) eProprietà estrinseche, con questi dettagli aggiuntivi:

- Se nella richiesta o nella risposta `extrinsicId` viene fornito un, questo valore ha la precedenza maggiore rispetto al valore fornito a livello di azione. Il sistema deve `extrinsicId` prima utilizzare request/response il livello, se manca utilizzare il livello di azione `extrinsicId`
- `extrinsicProperties`Se fornite nella richiesta o nella risposta, queste proprietà hanno la precedenza maggiore rispetto al valore va fornito a livello di azione. Il sistema deve passare al livello di azione `extrinsicProperties` e sostituire le coppie chiave-valore fornite a livello request/response `extrinsicProperties`

Example Esempio di override di ExtrinsicId ed ExtrinsicProperties

```
{
   "name": "ToggleWithEffect",
   "extrinsicId": "0x0001",

   "extrinsicProperties": {
      "apiMaturity": "provisional",
      "introducedIn": "1.2"
   },
   "request": {
      "extrinsicProperties": {
         "apiMaturity": "stable",
         "manufacturerCode": "XYZ"
      },
      "parameters": {
         ...
      }
   },
   "response": {
```

```
      "extrinsicProperties": {
         "noDefaultImplementation": true
      },
      "parameters": {
 {
         ...
      }
   }
}
```

Nell'esempio precedente, i valori effettivi per la richiesta di azione sarebbero:

```
# effective request
"name": "ToggleWithEffect",
"extrinsicId": "0x0001",
"extrinsicProperties": {
   "apiMaturity": "stable",
   "introducedIn": "1.2"
   "manufacturerCode": "XYZ"
},
"parameters": {
   ...
}

# effective response
"name": "ToggleWithEffectResponse",
"extrinsicId": "0x0001",
"extrinsicProperties": {
   "apiMaturity": "provisional",
   "introducedIn": "1.2"
   "noDefaultImplementation": true
},
"parameters": {
   ...
}
```

Azioni integrate

Per tutte le funzionalità, puoi eseguire azioni personalizzate utilizzando le parole chiave `ReadState` e`UpdateState`. Queste due parole chiave di azione agiranno sulle proprietà della funzionalità definite nel modello di dati.

ReadState

Invia un comando a `managedThing` per leggere i valori delle sue proprietà di stato. Viene utilizzato `ReadState` come metodo per forzare l'aggiornamento dello stato del dispositivo.

UpdateState

Invia un comando per aggiornare alcune proprietà.

Forzare la sincronizzazione dello stato del dispositivo può essere utile nei seguenti scenari:

1. Il dispositivo è rimasto offline per un periodo di tempo e non emetteva eventi.
2. Il dispositivo è stato appena fornito e non dispone ancora di alcuno stato mantenuto nel cloud.
3. Lo stato del dispositivo non è sincronizzato con lo stato reale del dispositivo.

ReadState esempi

Controlla se la luce è accesa o spenta utilizzando l'SendManagedThingCommandAPI:

```
{
  "Endpoints": [
    {
      "endpointId": "1",
      "capabilities": [
        {
          "id": "aws.OnOff",
          "name": "On/Off",
          "version": "1",
          "actions": [
            {
              "name": "ReadState",
              "parameters": {
                "propertiesToRead": [ "OnOff" ]
              }
            }
          ]
        }
      ]
    }
  ]
}
```

Leggi tutte le proprietà dello stato relative alla `matter.OnOff` funzionalità:

```
{
  "Endpoints": [
    {
      "endpointId": "1",
      "capabilities": [
        {
          "id": "aws.OnOff",
          "name": "On/Off",
          "version": "1",
          "actions": [
            {
              "name": "ReadState",
              "parameters": {
                "propertiesToRead": [ "*" ]
                // Use the wildcard operator to read ALL state properties for a
 capability
              }
            }
          ]
        }
      ]
    }
  ]
}
```

UpdateState esempio

Cambia `OnTime` for a light usando l'SendManagedThingCommandAPI:

```
{
  "Endpoints": [
    {
      "endpointId": "1",
      "capabilities": [
        {
          "id": "matter.OnOff",
          "name": "On/Off",
          "version": "1",
          "actions": [
            {
              "name": "UpdateState",
```

```
              "parameters": {
                "OnTime": 5
              }
            }
          ]
        }
      ]
    }
  ]
}
```

Eventi

Gli eventi sono segnali unidirezionali gestiti dallo schema e implementati dal dispositivo.

Implementa gli eventi in base a questi vincoli:

- Includi solo eventi unici nell'array degli eventi
- Includi un numero qualsiasi di eventi che rientrano nei limiti di dimensione del documento dello schema

Eventi nei sistemi di integrazione gestiti

Lavorare con gli eventi

Un evento è un modo standardizzato per conoscere in modo proattivo le modifiche apportate a un dispositivo o all'ambiente circostante. Rappresenta un evento modellato che il dispositivo invierà al cloud per fornire informazioni su qualcosa che è stato modificato sul dispositivo o rilevato nel suo ambiente. Poiché questi eventi sono modellati, i clienti possono utilizzarli in un flusso di controllo per reagire a eventi specifici e ai dettagli forniti al loro interno.

Per un evento, gli elementi obbligatori sono `name` e`extrinsicId`. Gli elementi opzionali sono `description`e`extrinsicProperties`, `request`.

Descrizione

La descrizione segue lo stesso formato descritto indescription, con una lunghezza massima di 512 caratteri.

Richiesta

La `request` sezione è facoltativa e può essere omessa se non sono presenti parametri di richiesta. Se omesso, il sistema supporta un dispositivo che invia una richiesta di evento senza alcun payload utilizzando semplicemente il nome dell'evento. Viene utilizzato in eventi semplici, come un guasto del sensore di una pompa o se l'allarme viene disattivato su un rilevatore di fumo o monossido di carbonio.

Le azioni complesse richiedono parametri aggiuntivi. Ad esempio, una richiesta di streaming delle riprese della telecamera potrebbe includere parametri relativi al protocollo di streaming da utilizzare o se inviare lo streaming a un dispositivo di visualizzazione specifico.

Per una richiesta di evento, l'elemento obbligatorio è`parameters`. Non ci sono elementi opzionali.

Risposta

Le risposte agli eventi non sono attualmente supportate.

Schema per le definizioni dei tipi

Le seguenti sezioni descrivono in dettaglio lo schema utilizzato per le definizioni dei tipi.

$id

L'elemento $id identifica la definizione dello schema. Deve seguire questa struttura:

- Inizia con il prefisso `/schema-versions/` URI
- Includi il tipo di `definition` schema
- Usa una barra (`/`) come separatore di percorso URI
- Includi l'identità dello schema, con frammenti separati da punti () `.`
- Usa il `@` carattere per separare l'ID e la versione dello schema
- Termina con la versione semver, usando period (`.`) per separare i frammenti di versione

L'identità dello schema deve iniziare con uno spazio dei nomi radice lungo 3-12 caratteri, seguito da uno spazio dei nomi secondario e da un nome opzionali.

La versione semver include una versione MAJOR (fino a 3 cifre), una versione MINOR (fino a 3 cifre) e una versione PATCH opzionale (fino a 4 cifre).

> **Note**
>
> Non è possibile utilizzare i namespace riservati o `aws matter`

Example Esempio $id

```
/schema-version/capability/aws.Recording@1.0
```

$ref

L'elemento $ref fa riferimento a una definizione di tipo esistente all'interno del sistema. Segue gli stessi vincoli dell'elemento. `$id`

> **Note**
>
> È necessario che esista una definizione o una funzionalità del tipo con il valore fornito nel `$ref` file.

Example Esempio $ref

```
/schema-version/definition/aws.capability@1.0
```

nome

L'elemento name è una stringa che rappresenta il nome dell'entità nel documento dello schema. Spesso contiene abbreviazioni e deve seguire queste regole:

- Contiene solo caratteri alfanumerici, punti (.), barre (/), trattini (-) e spazi
- Inizia con una lettera
- Massimo 192 caratteri

L'elemento name viene utilizzato nell'interfaccia utente e nella documentazione della console di Amazon Web Services.

Example Nomi di esempio

```
Door Lock
```

```
On/Off
Wi-Fi Network Management
PM2.5 Concentration Measurement
RTCSessionController
Energy EVSE
```

titolo

L'elemento title è una stringa descrittiva per l'entità rappresentata dal documento dello schema. Può contenere qualsiasi carattere e viene utilizzato nella documentazione.

Example Titoli di esempio

```
Real-time Communication (RTC) Session Controller
Energy EVSE Capability
```

description

L'`description`elemento fornisce una spiegazione dettagliata dell'entità rappresentata dal documento dello schema. Può contenere qualsiasi carattere e viene utilizzato nella documentazione.

Example Descrizione di esempio

```
Electric Vehicle Supply Equipment (EVSE) is equipment used to charge an Electric
 Vehicle (EV) or Plug-In Hybrid Electric Vehicle.
           This capability provides an interface to the functionality of Electric
 Vehicle Supply Equipment (EVSE) management.
```

ExtrinsICID

L'`extrinsicId`elemento rappresenta un identificatore gestito all'esterno del sistema IoT di Amazon Web Services. Per le funzionalità di Matter, viene mappato a `clusterId` `attributeId`,`commandId`,`eventId`, o `fieldId`, a seconda del contesto.

`extrinsicId`Può essere un numero intero decimale con stringhe (1-10 cifre) o un intero esadecimale con stringhe (prefisso 0x o 0X, seguito da 1-8 cifre esadecimali).

> **Note**
>
> Per, l' AWS ID fornitore (VID) è 0x1577 e per Matter è 0. Il sistema garantisce che gli schemi personalizzati non utilizzino queste funzionalità riservate. VIDs

Example Esempio: ExtrinsICIDS

```
0018
0x001A
0x15771002
```

Proprietà estrinseche

L'`extrinsicProperties`elemento contiene un insieme di proprietà definite in un sistema esterno ma mantenute all'interno del modello di dati. Per quanto riguarda le funzionalità di Matter, esegue il mapping a diversi elementi non modellati o parzialmente modellati all'interno del cluster, dell'attributo, del comando o dell'evento ZCL.

Le proprietà estrinseche devono rispettare questi vincoli:

- I nomi delle proprietà devono essere alfanumerici senza spazi o caratteri speciali
- I valori delle proprietà possono essere qualsiasi valore dello schema JSON
- Massimo 20 proprietà

Il sistema supporta diverse funzionalità`extrinsicProperties`, tra cui `accessapiMaturity`,`cli`,`cliFunctionName`, e altre ancora. Queste proprietà facilitano le trasformazioni tra ACL e modelli di dati AWS (e viceversa).

> **Note**
>
> Le proprietà estrinseche sono supportate per gli elementi `actionevent`,`property`, e `struct` fields di una funzionalità, ma non per la capacità o il cluster stesso.

Proprietà estrinseche supportate dal sistema

Il sistema tiene traccia dei seguenti attributi di cluster, attributo, comando o evento non modellati o parzialmente modellati, come `extrinsicProperties` durante le trasformazioni da o verso ZCL:

`access`

Ogni oggetto di accesso contiene quanto segue:

- op- Operazione modellata come e `enum` con valori:`read`,`write`, o `invoke`

- `privilege`- Privilegio modellato come e `enum` con valori:`view`,,,`proxy_view`, o `operate` `manage` `administer`
- `role`- Stringa illimitata che rappresenta un ruolo di operatore

`apiMaturity`

Una stringa semplice illimitata che rappresenta il livello di maturità. Questo è modellato in ZCL come e `enum` con valori:,,, o `stable` `provisional` `internal` `deprecated`

`side`

Modellato come enum con valori:,, e `either` `server` `client`

Proprietà booleane

Le seguenti proprietà sono bandiere booleane:

- `isFabricScoped`
- `isFabricSensitive`
- `mustUseAtomicWrite`
- `mustUseTimedInvoke`

Proprietà delle stringhe

Le seguenti proprietà sono rappresentate come stringhe illimitate:

- `cli`
- `cliFunctionName`
- `functionName`
- `group`
- `introducedIn`
- `manufacturerCode`
- `noDefaultImplementation`
- `presentIf`
- `priority`
- `removedIn`
- `reportableChange`
- `reportMinInterval`

- `reportMaxInterval`
- `restriction`
- `storage`

Considerazioni sulla trasformazione

Per le trasformazioni ZCL, `extrinsicProperties` vengono memorizzate in una mappa senza elaborazione. Gli schemi personalizzati che utilizzano il rilevamento non subiscono la trasformazione ZCL. Tuttavia, se si prevede di implementare trasformazioni ZCL per schemi personalizzati in futuro, è necessario modellare tutti i tipi di stringa semplice illimitati `extrinsicProperties` e definire vincoli come enumerazioni, pattern (regex) e lunghezza. Questa preparazione garantisce la corretta gestione di queste proprietà durante la trasformazione.

Al contrario, le trasformazioni AWS da connettore a connettore non `extrinsicProperties` sono affatto incluse, poiché questi dettagli non sono richiesti nel formato del connettore.

Creazione e utilizzo delle definizioni dei tipi nei documenti dello schema delle capacità

Tutti gli elementi degli schemi si risolvono in definizioni di tipo. Queste definizioni di tipo sono definizioni di tipo primitive (come booleani, stringhe, numeri) o definizioni di tipo con namespace (definizioni di tipo create a partire da definizioni di tipo primitive per comodità).

Quando si definisce uno schema personalizzato, è possibile utilizzare sia definizioni primitive che definizioni di tipi di namespace.

Indice

- Tipi di matter
- Tipi di aws
  - Definizione del tipo di bitmap
  - Definizione del tipo Enum

Definizioni di tipi primitivi

Le definizioni di tipo primitive sono gli elementi costitutivi di tutte le definizioni di tipo definite nelle integrazioni gestite. Tutte le definizioni dello spazio dei nomi, incluse le definizioni dei tipi personalizzate, si risolvono in una definizione di tipo primitiva tramite la parola chiave o la `$ref` parola chiave. `type`

Tutti i tipi primitivi sono annullabili utilizzando la `nullable` parola chiave ed è possibile identificare tutti i tipi primitivi utilizzando la parola chiave. `type`

Booleani

I tipi booleani supportano i valori predefiniti.

Definizione di esempio:

```
{
    "type" : "boolean",
    "default" : "false",
    "nullable" : true
}
```

Supporto per tipi di numeri interi

I tipi interi supportano quanto segue:

- `default` valori
- `maximum` valori
- `minimum` valori
- `exclusiveMaximum` valori
- `exclusiveMinimum` valori

- `multipleOf` valori

Se x è il valore da convalidare, deve essere vero quanto segue:

- x ≥ `minimum`
- x > `exclusiveMinimum`
- x < `exclusiveMaximum`

> **Note**
>
> I numeri con una parte frazionaria zero sono considerati numeri interi, ma i numeri in virgola mobile vengono rifiutati.
>
> ```
> 1.0 // Schema-Compliant
> 3.1415926 // NOT Schema-Compliant
> ```

Sebbene sia possibile specificare entrambi `minimum` e `exclusiveMinimum` o entrambi `maximum` e`exclusiveMaximum`, non è consigliabile utilizzarli entrambi contemporaneamente.

Definizioni di esempio:

```
{
    "type" : "integer",
    "default" : 2,
    "nullable" : true,
    "maximum" : 10,
    "minimum" : 0,
    "multipleOf": 2
}
```

Definizione alternativa:

```
{
    "type" : "integer",
    "default" : 2,
    "nullable" : true,
    "exclusiveMaximum" : 11,
```

```
    "exclusiveMinimum" : -1,
    "multipleOf": 2
}
```

Numeri

Utilizza il tipo numerico per qualsiasi tipo numerico, inclusi numeri interi e numeri in virgola mobile.

I tipi di numeri supportano quanto segue:

- `default` valori
- `maximum` valori
- `minimum` valori
- `exclusiveMaximum` valori
- `exclusiveMinimum` valori
- `multipleOf`valori. Il multiplo può essere un numero in virgola mobile.

Se `x` è il valore da convalidare, deve essere vero quanto segue:

- `x` **≥** `minimum`
- `x` **>** `exclusiveMinimum`
- `x` **<** `exclusiveMaximum`

Sebbene sia possibile specificare entrambi `minimum` e `exclusiveMinimum` o entrambi `maximum` e`exclusiveMaximum`, non è consigliabile utilizzarli entrambi contemporaneamente.

Definizioni di esempio:

```
{
    "type" : "number",
    "default" : 0.4,
    "nullable" : true,
    "maximum" : 10.2,
    "minimum" : 0.2,
    "multipleOf": 0.2
}
```

Definizione alternativa:

```
{
    "type" : "number",
    "default" : 0.4,
    "nullable" : true,
    "exclusiveMaximum" : 10.2,
    "exclusiveMinimum" : 0.2,
    "multipleOf": 0.2
}
```

Stringhe

I tipi di stringa supportano quanto segue:

- `default` valori
- vincoli di lunghezza (devono essere numeri non negativi) inclusi e valori `maxLength` `minLength`
- `pattern`valori per le espressioni regolari

Quando si definiscono espressioni regolari, la stringa è valida se l'espressione corrisponde a un punto qualsiasi all'interno della stringa. Ad esempio, l'espressione regolare p corrisponde a qualsiasi stringa contenente una p, come «apple», non solo alla stringa «p». Per motivi di chiarezza, consigliamo di racchiudere le espressioni regolari con `^...$` (ad esempio,^p$), a meno che non abbiate un motivo specifico per non farlo.

Definizione di esempio:

```
{
    "type" : "string",
    "default" : "defaultString",
    "nullable" : true,
    "maxLength": 10,
    "minLength": 1,
    "pattern" : "^([0-9a-fA-F]{2})+$"
}
```

Nulls

I tipi nulli accettano solo un singolo valore:`null`.

Definizione di esempio:

```
{ "type": "null" }
```

Matrici

I tipi di array supportano quanto segue:

- `default`— un elenco che verrà utilizzato come valore predefinito.
- `items`— Definizione del tipo JSON imposta a tutti gli elementi dell'array.
- Vincoli di lunghezza (deve essere un numero non negativo)
  - `minItems`
  - `maxItems`
- `pattern`valori per Regex
- `uniqueItems`— un booleano che indica se gli elementi dell'array devono essere unici
- `prefixItems`— un array in cui ogni elemento è uno schema che corrisponde a ciascun indice dell'array del documento. Cioè, un array in cui il primo elemento convalida il primo elemento dell'array di input, il secondo elemento convalida il secondo elemento dell'array di input e così via.

Definizione di esempio:

```
{
   "type": "array",
   "default": ["1", "2"],
   "items" : {
        "type": "string",
        "pattern": "^([a-zA-Z0-9_ -/]+)$"
    },
    "minItems" : 1,
    "maxItems": 4,
    "uniqueItems" : true,
}
```

Esempi di convalida degli array:

```
//Examples:
["1", "2", "3", "4"] // Schema-Compliant
[]                   // NOT Schema-Compliant: minItems=1
["1", "1"]           // NOT Schema-Compliant: uniqueItems=true
```

```
["{"]               // NOT Schema-Compliant: Does not match the RegEx pattern.
```

Definizione alternativa utilizzando la convalida delle tuple:

```
{
    "type": "array",
    "prefixItems": [
        { "type": "number" },
        { "type": "string" },
        { "enum": ["Street", "Avenue", "Boulevard"] },
        { "enum": ["NW", "NE", "SW", "SE"] }
    ]
}

//Examples:
[1600, "Pennsylvania", "Avenue", "NW"]              // Schema-Compliant

// And, by default, it's also okay to add additional items to end:
[1600, "Pennsylvania", "Avenue", "NW", "Washington"] // Schema-Compliant
```

Oggetti

I tipi di oggetti supportano quanto segue:

- Vincoli di proprietà
  - `properties`— Definire le proprietà (coppie chiave-valore) di un oggetto utilizzando la parola chiave. `properties` Il valore di `properties` è un oggetto, dove ogni chiave è il nome di una proprietà e ogni valore è uno schema utilizzato per convalidare quella proprietà. Qualsiasi proprietà che non corrisponde a nessuno dei nomi di proprietà nella `properties` parola chiave viene ignorata da questa parola chiave.
  - `required`— Per impostazione predefinita, le proprietà definite dalla `properties` parola chiave non sono obbligatorie. Tuttavia, è possibile fornire un elenco di proprietà obbligatorie utilizzando la `required` parola chiave. La `required` parola chiave accetta una matrice di zero o più stringhe. Ognuna di queste stringhe deve essere unica.
  - `propertyNames`— Questa parola chiave consente di controllare lo RegEx schema dei nomi delle proprietà. Ad esempio, potreste voler imporre che tutte le proprietà di un oggetto abbiano nomi che seguano una convenzione specifica.
  - `patternProperties`— Questo associa le espressioni regolari agli schemi. Se il nome di una proprietà corrisponde all'espressione regolare specificata, il valore della proprietà deve essere

convalidato rispetto allo schema corrispondente. Ad esempio, utilizzare `patternProperties` per specificare che un valore deve corrispondere a uno schema particolare, dato un particolare tipo di nome di proprietà.

- `additionalProperties`— Questa parola chiave controlla come vengono gestite le proprietà aggiuntive. Le proprietà aggiuntive sono proprietà i cui nomi non sono elencati nella parola chiave properties o che corrispondono a `patternProperties` nessuna delle espressioni regolari di. Per impostazione predefinita, sono consentite proprietà aggiuntive. L'impostazione di questo campo su `false` significa che non sono consentite proprietà aggiuntive.
- `unevaluatedProperties`— Questa parola chiave è simile a, `additionalProperties` tranne per il fatto che può riconoscere le proprietà dichiarate nei sottoschemi. `unevaluatedProperties`funziona raccogliendo tutte le proprietà che vengono convalidate con successo durante l'elaborazione degli schemi e utilizzandole come elenco di proprietà consentite. Ciò consente di eseguire operazioni più complesse come l'aggiunta di proprietà in modo condizionale. Fate riferimento all'esempio seguente per maggiori dettagli.

- `anyOf`— Il valore di questa parola chiave DEVE essere un array non vuoto. Ogni elemento dell'array DEVE essere uno schema JSON valido. Un'istanza viene convalidata correttamente rispetto a questa parola chiave se viene convalidata con successo su almeno uno schema definito dal valore di questa parola chiave.
- `oneOf`— Il valore di questa parola chiave DEVE essere un array non vuoto. Ogni elemento dell'array DEVE essere uno schema JSON valido. Un'istanza viene convalidata correttamente rispetto a questa parola chiave se viene convalidata con successo esattamente su uno schema definito dal valore di questa parola chiave.

Esempio di obbligatorio:

```
{
  "type": "object",
  "required": ["test"]
}

// Schema Compliant
{
  "test": 4
}

// NOT Schema Compliant
{}
```

PropertyNames esempio:

```
{
  "type": "object",
  "propertyNames": {
    "pattern": "^[A-Za-z_][A-Za-z0-9_]*$"
  }
}

// Schema Compliant
{
  "_a_valid_property_name_001": "value"
}

// NOT Schema Compliant
{
  "001 invalid": "value"
}
```

PatternProperties esempio:

```
{
  "type": "object",
  "patternProperties": {
    "^S_": { "type": "string" },
    "^I_": { "type": "integer" }
  }
}

// Schema Compliant
{ "S_25": "This is a string" }
{ "I_0": 42 }

// NOT Schema Compliant
{ "S_0": 42 } // Value must be a string
{ "I_42": "This is a string" } // Value must be an integer
```

AdditionalProperties esempio:

```
{
  "type": "object",
  "properties": {
```

```
    "test": {
        "type": "string"
    }
  },
  "additionalProperties": false
}

// Schema Compliant
{
  "test": "value"
}
OR
{}

// NOT Schema Compliant
{
    "notAllowed": false
}
```

UnevaluatedProperties esempio:

```
{
  "type": "object",
  "properties": {
    "standard_field": { "type": "string" }
  },
  "patternProperties": {
    "^@": { "type": "integer" } // Allows properties starting with '@'
  },
  "unevaluatedProperties": false // No other properties allowed
}

// Schema Compliant
{
  "standard_field": "some value",
  "@id": 123,
  "@timestamp": 1678886400
}
// This passes because "standard_field" is evaluated by properties,
// "@id" and "@timestamp" are evaluated by patternProperties,
// and no other properties remain unevaluated.

// NOT Schema Compliant
```

```
{
  "standard_field": "some value",
  "another_field": "unallowed"
}
// This fails because "another_field" is unevaluated and doesn't match
// the @ pattern, leading to a violation of unevaluatedProperties: false
```

AnyOf esempio:

```
{
  "anyOf": [
    { "type": "string", "maxLength": 5 },
    { "type": "number", "minimum": 0 }
  ]
}

// Schema Compliant
"short"
12

// NOT Schema Compliant
"too long"
-5
```

OneOf esempio:

```
{
  "oneOf": [
    { "type": "number", "multipleOf": 5 },
    { "type": "number", "multipleOf": 3 }
  ]
}

// Schema Compliant
10
9

// NOT Schema compliant
2  // Not a multiple of either 5 or 3
15 // Multiple of both 5 and 3 is rejected.
```

definizioni dei tipi con namespace

Le definizioni dei tipi con namespace sono tipi creati a partire da tipi primitivi. Questi tipi devono seguire il formato Le integrazioni *`namespace`*`.`*`typename`*`.` gestite forniscono tipi predefiniti nei namespace e. `aws` `matter` Puoi utilizzare qualsiasi spazio dei nomi per i tipi personalizzati ad eccezione dei namespace riservati e dei nomi. `aws` `matter`

Per trovare le definizioni dei tipi con namespace disponibili, utilizza l'API con il filtro impostato su. ListSchemaVersions`Typedefinition`

Tipi di **`matter`**

Trova i tipi di dati nello spazio dei `matter` nomi utilizzando l'ListSchemaVersionsAPI con il filtro impostato su `matter` e il `Namespace` filtro impostato su. `Type` `definition`

Tipi di **`aws`**

Trova i tipi di dati nel `aws` namespace utilizzando l'ListSchemaVersionsAPI con il filtro impostato su `aws` e il `Namespace` `Type` filtro impostato su. `definition`

Definizione del tipo di bitmap

Le bitmap hanno due proprietà obbligatorie:

- `type`deve essere un oggetto
- `properties`deve essere un oggetto contenente ogni definizione di bit. Ogni bit è un oggetto con proprietà `extrinsicId` `evalue`. Il valore di ogni bit deve essere un numero intero con un valore minimo di 0 e un valore massimo di almeno 1.

Esempio di definizione bitmap:

```
{
    "title" : "Sample Bitmap Type",
    "description" : "Type definition for SampleBitmap.",
    "$ref" : "/schema-versions/definition/aws.bitmap@1.0 ",
    "type" : "object",
    "additionalProperties" : false,
    "properties" : {
        "Bit1" : {
            "extrinsicId" : "0x0000",
            "value" : {
```

```
                "type" : "integer",
                "maximum" : 1,
                "minimum" : 0
            }
        },
        "Bit2" : {
            "extrinsicId" : "0x0001",
            "value" : {
                "type" : "integer",
                "maximum" : 1,
                "minimum" : 0
            }
        }
    }
}

// Schema Compliant
{
    "Bit1": 1,
    "Bit1": 0
}

// NOT Schema Compliant
{
    "Bit1": -1,
    "Bit1": 0
}
```

Definizione del tipo Enum

Le enumerazioni richiedono tre proprietà:

- `type`deve essere un oggetto
- `enum`deve essere un array di stringhe uniche, con almeno un elemento
- `extrinsicIdMap`è un oggetto con proprietà che sono i valori enum. Il valore di ciascuna proprietà deve essere l'identificatore estrinseco che corrisponde al valore enum.

Esempio di definizione enum:

```
{
    "title" : "SampleEnum Type",
    "description" : "Type definition for SampleEnum.",
```

```
    "$ref" : "/schema-versions/definition/aws.enum@1.0",
    "type" : "string",
    "enum" : [
        "EnumValue0",
        "EnumValue1",
        "EnumValue2"
    ],
    "extrinsicIdMap" : {
        "EnumValue0" : "0",
        "EnumValue1" : "1",
        "EnumValue2" : "2"
    }
}

// Schema Compliant
"EnumValue0"
"EnumValue1"
"EnumValue2"

// NOT Schema Compliant
"NotAnEnumValue"
```

Gestisci i comandi e gli eventi dei dispositivi IoT

I comandi del dispositivo offrono la possibilità di gestire in remoto un dispositivo fisico garantendo il controllo completo sul dispositivo, oltre a eseguire aggiornamenti critici di sicurezza, software e hardware. Con un'ampia flotta di dispositivi, sapere quando un dispositivo esegue un comando consente di supervisionare l'intera implementazione del dispositivo. Un comando del dispositivo o un aggiornamento automatico attiverà una modifica dello stato del dispositivo, che a sua volta creerà un nuovo evento del dispositivo. Questo evento del dispositivo attiverà una notifica inviata automaticamente a una destinazione gestita dal cliente.

Argomenti

Comandi per i dispositivi

Una richiesta di comando è il comando inviato al dispositivo. Una richiesta di comando include un payload che specifica l'azione da intraprendere, ad esempio accendere la lampadina. Per inviare un comando al dispositivo, l'`SendManagedThingCommand`API viene chiamata per conto dell'utente finale tramite integrazioni gestite e la richiesta di comando viene inviata al dispositivo.

La risposta a a `SendManagedThingCommand` è a `traceId` ed è possibile utilizzarla `traceId` per tenere traccia della consegna dei comandi e dei relativi flussi di lavoro di risposta ai comandi, ove possibile.

Per ulteriori informazioni sul funzionamento dell'`SendManagedThingCommand`API, consulta SendManagedThingCommand.

Operazione **`UpdateState`**

Per aggiornare lo stato di un dispositivo, ad esempio l'ora in cui si accende una luce, utilizza l'`UpdateState`azione quando si chiama l'`SendManagedThingCommand`API. Fornisci la proprietà del modello di dati e il nuovo valore in cui esegui l'aggiornamento`parameters`. L'esempio seguente illustra una richiesta `SendManagedThingCommand` API che aggiorna una lampadina a`5`. `OnTime`

```
{
  "Endpoints": [
    {
```

```
      "endpointId": "1",
      "capabilities": [
        {
          "id": "matter.OnOff",
          "name": "On/Off",
          "version": "1",
          "actions": [
            {
              "name": "UpdateState",
              "parameters": {
                "OnTime": 5
              }
            }
          ]
        }
      ]
    }
  ]
}
```

Operazione **`ReadState`**

Per ottenere lo stato più recente di un dispositivo, inclusi i valori correnti di tutte le proprietà del modello di dati, utilizzate l'`ReadState`azione quando chiamate l'`SendManagedThingCommand`API. In`propertiesToRead`, puoi utilizzare le seguenti opzioni:

- Fornisci una proprietà specifica del modello di dati per ottenere il valore più recente, ad esempio `OnOff` per determinare se una luce è accesa o spenta.
- Utilizzate l'operatore wildcard (`*`) per leggere tutte le proprietà dello stato del dispositivo relative a una funzionalità.

Gli esempi seguenti illustrano entrambi gli scenari per una richiesta `SendManagedThingCommand` API che utilizza l'azione`ReadState`:

```
{
  "Endpoints": [
    {
      "endpointId": "1",
      "capabilities": [
        {
          "id": "aws.OnOff",
```

```
          "name": "On/Off",
          "version": "1",
          "actions": [
            {
              "name": "ReadState",
              "parameters": {
                "propertiesToRead": [ "OnOff" ]
              }
            }
          ]
        }
      ]
    }
  ]
}
```

```
{
  "Endpoints": [
    {
      "endpointId": "1",
      "capabilities": [
        {
          "id": "aws.OnOff",
          "name": "On/Off",
          "version": "1",
          "actions": [
            {
              "name": "ReadState",
              "parameters": {
                "propertiesToRead": [ "*" ]
              }
            }
          ]
        }
      ]
    }
  ]}
```

Eventi del dispositivo

Un evento del dispositivo include lo stato corrente del dispositivo. Ciò può significare che il dispositivo ha cambiato stato o sta segnalando il suo stato anche se lo stato non è cambiato. Include report sulle

proprietà ed eventi definiti nel modello di dati. Un evento può essere il completamento del ciclo della lavatrice o il termostato ha raggiunto la temperatura desiderata impostata dall'utente finale.

Notifiche degli eventi del dispositivo

Un utente finale può abbonarsi a destinazioni specifiche gestite dal cliente che crea per gli aggiornamenti su eventi specifici del dispositivo. Per creare una destinazione gestita dal cliente, chiama l'API. `CreateDestination` Quando un evento del dispositivo viene segnalato alle integrazioni gestite dal dispositivo, la destinazione gestita dal cliente riceve una notifica se ne esiste una.

Etichettatura delle risorse gestite per le integrazioni

Per aiutarti a gestire e organizzare le tue risorse, puoi facoltativamente assegnare i tuoi metadati a ciascuna di queste risorse sotto forma di tag. Questa sezione descrive i tag e mostra come crearli.

Nozioni di base sui tag

Puoi utilizzare i tag per classificare le risorse di integrazione gestite 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. 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'API AWS CLI e le integrazioni gestite. Puoi associare i tag a oggetti gestiti, profili di provisioning, locker di credenziali e attività over-the-air (OTA) quando li crei utilizzando il `Tags` campo nei seguenti comandi:

- CreateManagedThing
- CreateProvisioningProfile
- CreateCredentialLocker
- CreateOtaTask
- CreateAccountAssociation

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 aggiungete 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

Puoi applicare autorizzazioni a livello di risorsa basate su tag nelle policy IAM che utilizzi per le azioni API di integrazione gestita. 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 azioni di integrazione gestite in cui un identificatore per una risorsa che può essere taggata è un parametro obbligatorio. Ad esempio, l'uso di non GetCustomEndpointè consentito o negato sulla base delle chiavi e dei valori del contesto delle condizioni perché in questa richiesta non viene fatto riferimento a nessuna risorsa etichettabile (oggetti gestiti, profili di provisioning, credenziali locker, over-the-air attività). Per ulteriori informazioni sulle integrazioni gestite, sulle risorse taggabili e sulle chiavi di condizione supportate, leggi la funzionalità Azioni, risorse e chiavi di condizione per le integrazioni gestite di. AWS IoT AWS IoT Device Management

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 seguente politica di esempio applica due restrizioni basate su tag per l'azione. `CreateManagedThing` Un utente IAM limitato da questa policy:

- Impossibile creare un oggetto gestito con il tag «env=prod» (nell'esempio, vedi la riga). `"aws:RequestTag/env" : "prod"`
- Impossibile modificare o accedere a un oggetto gestito che ha un tag esistente «env=prod» (nell'esempio, vedi la riga). `"aws:ResourceTag/env" : "prod"`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
```

```
      "Action": "iotmanagedintegrations:CreateManagedThing",
      "Resource": "arn:aws:iotmanagedintegrations:us-east-1:123456789012:managed-
thing/*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "iotmanagedintegrations:CreateManagedThing",
        "iotmanagedintegrations:DeleteManagedThing",
        "iotmanagedintegrations:GetManagedThing",
        "iotmanagedintegrations:UpdateManagedThing"
      ],
      "Resource": "arn:aws:iotmanagedintegrations:us-east-1:123456789012:managed-
thing/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotmanagedintegrations:CreateManagedThing",
        "iotmanagedintegrations:DeleteManagedThing",
        "iotmanagedintegrations:GetManagedThing",
        "iotmanagedintegrations:UpdateManagedThing"
      ],
      "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.

Notifiche di integrazioni gestite

Le notifiche di integrazioni gestite forniscono aggiornamenti e informazioni chiave dai dispositivi. Le notifiche includono eventi relativi ai connettori, comandi dei dispositivi, eventi del ciclo di vita, aggiornamenti OTA (Over-the-Air) e segnalazioni di errori. Queste informazioni forniscono informazioni utili per creare flussi di lavoro automatizzati, intraprendere azioni immediate o archiviare dati sugli eventi per la risoluzione dei problemi.

Attualmente, solo i flussi di dati di Amazon Kinesis sono supportati come destinazione per le notifiche di integrazioni gestite. Prima di configurare le notifiche, devi prima configurare un flusso di dati Amazon Kinesis e consentire alle integrazioni gestite di accedere al flusso di dati.

Configurare Amazon Kinesis per le notifiche

Fasi di configurazione di Amazon Kinesis

- Fase 1: creare un flusso di dati Amazon Kinesis
- Fase 2: Creare una politica di autorizzazioni
- Passaggio 3: accedi alla dashboard IAM e seleziona Ruoli
- Fase 4: Utilizzare una politica di fiducia personalizzata
- Passaggio 5: Applica la tua politica sulle autorizzazioni
- Fase 6: Inserire il nome del ruolo

Per configurare le notifiche di Amazon Kinesis per le integrazioni gestite, segui questi passaggi:

Fase 1: creare un flusso di dati Amazon Kinesis

Un Amazon Kinesis Data Stream può importare una grande quantità di dati in tempo reale, archiviarli in modo duraturo e renderli disponibili per l'utilizzo da parte delle applicazioni.

Per creare un flusso di dati Amazon Kinesis

- Per creare un flusso di dati Kinesis, segui i passaggi descritti in Creare e gestire flussi di dati Kinesis.

Fase 2: Creare una politica di autorizzazioni

Crea una politica di autorizzazioni che consenta alle integrazioni gestite di accedere al flusso di dati Kinesis.

Per creare una politica di autorizzazioni

- Per creare una politica di autorizzazioni, copia la politica seguente e segui i passaggi descritti in Creare politiche utilizzando l'editor JSON

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": "kinesis:PutRecord",
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Passaggio 3: accedi alla dashboard IAM e seleziona Ruoli

Apri la dashboard IAM e fai clic su Ruoli.

Per accedere alla dashboard IAM

- Apri la dashboard IAM e fai clic su Ruoli.

  Per ulteriori informazioni, consulta la creazione di ruoli IAM nella Guida AWS Identity and Access Managementper l'utente.

Fase 4: Utilizzare una politica di fiducia personalizzata

Puoi utilizzare una policy di fiducia personalizzata per concedere alle integrazioni gestite l'accesso al flusso di dati Kinesis.

Per utilizzare una politica di fiducia personalizzata

- Crea un nuovo ruolo e scegli una politica di fiducia personalizzata. Fai clic su Avanti.

  La seguente politica consente alle integrazioni gestite di assumere il ruolo e la `Condition` dichiarazione aiuta a prevenire problemi confusi tra i deputati.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iotmanagedintegrations.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:iotmanagedintegrations:ca-
central-1:123456789012:*"
                }
            }
        }
    ]
}
```

Passaggio 5: Applica la tua politica sulle autorizzazioni

Aggiungi al ruolo la politica di autorizzazione creata nel passaggio 2.

Per aggiungere una politica di autorizzazioni

- Nella pagina Aggiungi autorizzazioni, cerca e aggiungi la politica di autorizzazione creata nel passaggio 2. Fai clic su Avanti.

Fase 6: Inserire il nome del ruolo

- Inserisci il nome del ruolo e fai clic su Crea ruolo.

Configura le notifiche relative alle integrazioni gestite

Fasi di configurazione delle notifiche

Per configurare le notifiche relative alle integrazioni gestite, segui questi passaggi:

Passaggio 1: concedere all'utente le autorizzazioni per chiamare l'API CreateDestination

- Concedi all'utente le autorizzazioni per chiamare l'API **`CreateDestination`**

  La seguente politica definisce i requisiti per l'utente per chiamare l'CreateDestinationAPI.

  Consulta Concedere a un utente le autorizzazioni per passare un ruolo a un AWS servizio nella Guida per l'AWS Identity and Access Managementutente per ottenere le autorizzazioni passrole per le integrazioni gestite.

  JSON

```
{
      "Version":"2012-10-17",
      "Statement":[
         {
            "Effect":"Allow",
            "Action":"iam:PassRole",

 "Resource":"arn:aws:iam::123456789012:role/ROLE_CREATED_IN_PREVIOUS_STEP",
            "Condition":{
               "StringEquals":{
```

```
 "iam:PassedToService":"iotmanagedintegrations.amazonaws.com"
                }
             }
          },
          {
             "Effect":"Allow",
             "Action":"iotmanagedintegrations:CreateDestination",
             "Resource":"*"
          }
       ]
}
```

Fase 2: Chiama l'API CreateDestination

- Chiama l'**`CreateDestination`**API

  Dopo aver creato il flusso di dati di Amazon Kinesis e il ruolo di accesso allo stream, chiama l'CreateDestinationAPI per creare la destinazione delle notifiche a cui verranno indirizzate le notifiche. Per il `DeliveryDestinationArn` parametro, usa il nuovo `arn` flusso di dati di Amazon Kinesis.

  ```
  {
      "DeliveryDestinationArn": "Your Kinesis arn"
      "DeliveryDestinationType": "KINESIS"
      "Name": "DestinationName"
      "ClientToken": "string"
      "RoleArn": "arn:aws:iam::accountID:role/ROLE_CREATED_IN_PREVIOUS_STEP"
  }
  ```

  > **Note**
  >
  > `ClientToken`è un token di idempotenza. Se riprovi una richiesta completata correttamente inizialmente utilizzando lo stesso token client e gli stessi parametri, il nuovo tentativo avrà esito positivo senza eseguire ulteriori azioni.

Fase 3: Chiama l'API CreateNotificationConfiguration

- Chiama l'**`CreateNotificationConfiguration`**API

  Infine, usa l'CreateNotificationConfigurationAPI per creare la configurazione di notifica che indirizza i tipi di eventi scelti verso la tua destinazione rappresentata dal flusso di dati Kinesis. Nel `DestinationName` parametro, utilizza lo stesso nome di destinazione utilizzato per la chiamata iniziale all'`CreateDestination`API.

```
{
    "EventType": "DEVICE_EVENT"
    "DestinationName" // This name has to be identical to the name in
 createDestination API
    "ClientToken": "string"
}
```

Tipi di eventi monitorati con integrazioni gestite

Di seguito sono riportati i tipi di eventi monitorati con notifiche di integrazioni gestite:

- `DEVICE_COMMAND`
  - Lo stato del comando SendManagedThingCommandAPI. I valori validi sono `succeeded` o `failed`.

```
{
              "version":"0",
              "messageId":"6a7e8feb-b491-4cf7-a9f1-bf3703467718",
              "messageType":"DEVICE_COMMAND",
              "source":"aws.iotmanagedintegrations",
              "customerAccountId":"123456789012",
              "timestamp":"2017-12-22T18:43:48Z",
              "region":"ca-central-1",
              "resources":[
                    "arn:aws:iotmanagedintegrations:ca-
central-1:123456789012:managed-thing/6a7e8feb-b491-4cf7-a9f1-bf3703467718"
              ],
              "payload":{
                "traceId":"1234567890abcdef0",
                "receivedAt":"2017-12-22T18:43:48Z",
                "executedAt":"2017-12-22T18:43:48Z",
```

```
                "result":"failed"
              }
}
```

- `DEVICE_COMMAND_REQUEST`
  - La richiesta di comando da Web Real-Time Communication (WebRTC).

    Lo standard WebRTC consente la comunicazione tra due peer. Questi peer possono trasmettere video, audio e dati arbitrari in tempo reale. Le integrazioni gestite supportano WebRTC per abilitare questi tipi di streaming tra un'applicazione mobile del cliente e il dispositivo di un utente finale. Per ulteriori informazioni sullo standard WebRTC, vedere WebRTC.

```
{
              "version":"0",
              "messageId":"6a7e8feb-b491-4cf7-a9f1-bf3703467718",
              "messageType":"DEVICE_COMMAND_REQUEST",
              "source":"aws.iotmanagedintegrations",
              "customerAccountId":"123456789012",
              "timestamp":"2017-12-22T18:43:48Z",
              "region":"ca-central-1",
              "resources":[
                  "arn:aws:iotmanagedintegrations:ca-
central-1:123456789012:managed-thing/6a7e8feb-b491-4cf7-a9f1-bf3703467718"
              ],
              "payload":{
                "endpoints":[{
                  "endpointId":"1",
                  "capabilities":[{
                    "id":"aws.DoorLock",
                    "name":"Door Lock",
                    "version":"1.0"
                  }]
                }]
              }
}
```

- `DEVICE_DISCOVERY_STATUS`
  - Lo stato di rilevamento del dispositivo.

```
{
      "version":"0",
      "messageId":"6a7e8feb-b491-4cf7-a9f1-bf3703467718",
```

```
      "messageType":"DEVICE_DISCOVERY_STATUS",
      "source":"aws.iotmanagedintegrations",
      "customerAccountId":"123456789012",
      "timestamp":"2017-12-22T18:43:48Z",
      "region":"ca-central-1",
      "resources":[
        "arn:aws:iotmanagedintegrations:ca-central-1:123456789012:managed-
thing/6a7e8feb-b491-4cf7-a9f1-bf3703467718"
      ],
      "payload":{
        "deviceCount": 1,
        "deviceDiscoveryId": "123",
        "status": "SUCCEEDED"
      }
}
```

- DEVICE_EVENT
  - Una notifica del verificarsi di un evento relativo al dispositivo.

```
{
      "version":"1.0",
      "messageId":"2ed545027bd347a2b855d28f94559940",
      "messageType":"DEVICE_EVENT",
      "source":"aws.iotmanagedintegrations",
      "customerAccountId":"123456789012",
      "timestamp":"1731630247280",
      "resources":[
        "/quit/1b15b39992f9460ba82c6c04595d1f4f"
      ],
      "payload":{
        "endpoints":[{
          "endpointId":"1",
          "capabilities":[{
            "id":"aws.DoorLock",
            "name":"Door Lock",
            "version":"1.0",
            "properties":[{
              "name":"ActuatorEnabled",
              "value":"true"
            }]
          }]
        }]
      }
```

```
}
```

- DEVICE_LIFE_CYCLE
  - Lo stato del ciclo di vita del dispositivo.

```
{
      "version": "1.0.0",
      "messageId": "8d1e311a473f44f89d821531a0907b05",
      "messageType": "DEVICE_LIFE_CYCLE",
      "source": "aws.iotmanagedintegrations",
      "customerAccountId": "123456789012",
      "timestamp": "2024-11-14T19:55:57.568284645Z",
      "region": "ca-central-1",
      "resources": [
        "arn:aws:iotmanagedintegrations:ca-central-1:123456789012:managed-thing/
d5c280b423a042f3933eed09cf408657"
        ],
      "payload": {
        "deviceDetails": {
          "id": "d5c280b423a042f3933eed09cf408657",
          "arn": "arn:aws:iotmanagedintegrations:ca-central-1:123456789012:managed-
thing/d5c280b423a042f3933eed09cf408657",
          "createdAt": "2024-11-14T19:55:57.515841147Z",
          "updatedAt": "2024-11-14T19:55:57.515841559Z"
        },
        "status": "UNCLAIMED"
      }
}
```

- DEVICE_OTA
  - Una notifica OTA del dispositivo.
- DEVICE_STATE
  - Una notifica quando lo stato di un dispositivo è stato aggiornato.

```
{
      "messageType": "DEVICE_STATE",
      "source": "aws.iotmanagedintegrations",
      "customerAccountId": "123456789012",
      "timestamp": "1731623291671",
      "resources": [
        "arn:aws:iotmanagedintegrations:ca-central-1:123456789012:managed-
thing/61889008880012345678"
```

```
      ],
      "payload": {
        "addedStates": {
          "endpoints": [{
            "endpointId": "nonEndpointId",
            "capabilities": [{
              "id": "aws.OnOff",
              "name": "On/Off",
              "version": "1.0",
              "properties": [{
                "name": "OnOff",
                "value": {
                  "propertyValue": "\"onoff\"",
                  "lastChangedAt": "2024-06-11T01:38:09.000414Z"
                }
              }
            ]}
          ]}
        ]}
      }
}
```

Cloud-to-Cloud Connettori (C2C)

Un cloud-to-cloud connettore consente di creare e facilitare la comunicazione bidirezionale tra dispositivi di terze parti e. AWS

Argomenti

Cos'è un connettore cloud-to-cloud (C2C)?

Un cloud-to-cloud connettore è un pacchetto software predefinito che lo collega in modo sicuro Cloud AWS all'endpoint di un provider cloud di terze parti. Utilizzando il connettore C2C, i fornitori di soluzioni possono sfruttare le integrazioni gestite per AWS IoT Device Management per controllare i dispositivi collegati a cloud di terze parti.

Le integrazioni gestite includono un catalogo di connettori in cui AWS i clienti possono visualizzare e selezionare i connettori con cui desiderano integrarsi. Per ulteriori informazioni, consulta Cos'è il catalogo dei connettori C2C?

Le integrazioni gestite richiedono che ogni connettore sia implementato come funzione. AWS Lambda

Cos'è il catalogo dei connettori C2C?

Il catalogo di integrazioni gestite per AWS IoT Device Management connector è una raccolta di connettori C2C che facilitano la comunicazione bidirezionale tra le integrazioni gestite per AWS IoT Device Management e un provider cloud di terze parti. È possibile visualizzare i connettori in o in. Console di gestione AWS AWS CLI

Per utilizzare la console per visualizzare il catalogo dei connettori di integrazioni gestite

1. Apri la console di integrazioni gestite
2. Nel riquadro di navigazione a sinistra, scegli Integrazioni gestite
3. Nel riquadro di navigazione a sinistra della console delle integrazioni gestite, scegli Catalogo.

AWS Lambda funziona come connettori C2C

Ogni funzione Lambda del connettore C2C traduce e trasporta comandi ed eventi tra le integrazioni gestite e le azioni corrispondenti su piattaforme di terze parti. Per ulteriori informazioni su Lambda, consulta What is. AWS Lambda

Ad esempio, supponiamo che un utente finale possieda una lampadina intelligente prodotta da un OEM terzo. Con un connettore C2C, un utente finale può emettere un comando per accendere o spegnere questa luce tramite una piattaforma di integrazioni gestita. Questo comando verrà quindi inoltrato alla funzione Lambda ospitata nel connettore, che tradurrà la richiesta in una chiamata API verso la piattaforma di terze parti per accendere o spegnere il dispositivo.

La funzione Lambda è richiesta quando si chiama l'`CreateCloudConnector`API. Il codice distribuito nella funzione Lambda deve implementare tutte le interfacce e le funzionalità menzionate in. Crea un connettore C2C (Cloud-to-Cloud)

Workflow del connettore per le integrazioni gestite

Gli sviluppatori devono registrare i connettori C2C con integrazioni gestite per. AWS IoT Device Management Questo processo di registrazione crea una risorsa di connettori logici a cui i clienti possono accedere per utilizzare il connettore.

> **Note**
>
> Un connettore C2C è un insieme di metadati creati all'interno di integrazioni gestite per AWS IoT Device Management per descrivere il connettore.

Il diagramma seguente illustra il ruolo di un connettore C2C nell'invio di un comando dall'applicazione mobile a un dispositivo connesso al cloud. Il connettore C2C funge da livello di traduzione tra le integrazioni gestite per AWS IoT Device Management e una piattaforma cloud di terze parti.

Linee guida per l'utilizzo di un connettore C2C () cloud-to-cloud

Qualsiasi connettore C2C che crei è un tuo contenuto e qualsiasi connettore C2C creato da un altro cliente a cui accedi è contenuto di terze parti. AWS non crea o gestisce alcun connettore C2C come parte di integrazioni gestite.

Puoi condividere i tuoi connettori C2C con altri clienti di integrazioni gestite. In tal caso, autorizzi in AWS qualità di fornitore di servizi a elencare tali connettori C2C e le relative informazioni di contatto sulla AWS console e comprendi che altri clienti potrebbero contattarti. AWS Sei l'unico responsabile della concessione ai clienti dell'accesso ai tuoi connettori C2C e di tutte le condizioni che regolano l'accesso di un altro AWS cliente ai tuoi connettori C2C.

Crea un connettore C2C (Cloud-to-Cloud)

Le seguenti sezioni illustrano i passaggi per creare un connettore C2C (Cloud-to-Cloud) per integrazioni gestite per AWS IoT Device Management.

Argomenti

Prerequisiti

Prima di creare un connettore C2C (Cloud-to-Cloud), è necessario quanto segue:

- E Account AWS per ospitare il connettore C2C e registrarlo tramite integrazioni gestite. Per ulteriori informazioni, consulta Creare un. Account AWS
- Quando crei il connettore, hai bisogno di determinate autorizzazioni IAM. Per utilizzare nuovamente il plugin

- Assicurati che i provider cloud di terze parti a cui è destinato il connettore supportino l'autorizzazione OAuth 2.0. Per ulteriori informazioni, consulta OAuth Requisiti 2.0 per il collegamento degli account.

  Inoltre, per testare il connettore, lo sviluppatore del connettore deve disporre di quanto segue:

  - Un ID client dal cloud di terze parti da associare al connettore C2C
  - Un client segreto proveniente dal cloud di terze parti da associare al connettore C2C
  - Un URL di OAuth autorizzazione 2.0
  - Un URL con token OAuth 2.0
- Qualsiasi chiave API richiesta dall'API di terze parti
- Qualsiasi chiave API richiesta dalla registrazione o dall'elenco di autorizzazioni API di terze parti per l'URL di OAuth callback ospitato da. AWS Alcune terze parti consentono esplicitamente di inserire un URL di OAuth reindirizzamento nell'elenco, mentre altre dispongono di un flusso di lavoro in cui gli utenti possono accedere e registrare l'URL. OAuth Rivolgiti alla terza parte specifica per capire cosa è necessario per consentire l'inserimento nell'elenco degli endpoint di reindirizzamento delle integrazioni gestite OAuth

Autorizzazioni richieste

Quando crei il connettore, hai bisogno di determinate autorizzazioni IAM. Oltre alle `iotmanagedintegrations:` autorizzazioni per le azioni, sono necessarie le seguenti autorizzazioni:

- CreateAccountAssociation, CreateConnectorDestination, e GetAccountAssociation, richiede StartAccountAssociationRefresh`secretsmanager:GetSecretValue`
- CreateCloudConnector richiede `lambda:Invoke`

Per ulteriori informazioni su `iotmanagedintegrations:` autorizzazioni e azioni, consulta Azioni definite dalle integrazioni AWS gestite

Requisiti del connettore C2C

Il connettore C2C che sviluppi facilita la comunicazione bidirezionale tra le integrazioni gestite per AWS IoT Device Management e un cloud di fornitori terzi. Il connettore deve implementare interfacce per integrazioni gestite per AWS IoT Device Management per eseguire azioni per conto degli utenti finali. Queste interfacce forniscono la funzionalità per scoprire i dispositivi degli utenti finali, avviare i

comandi dei dispositivi inviati da integrazioni gestite per AWS IoT Device Management e identificare gli utenti sulla base di un token di accesso. Per supportare le operazioni del dispositivo, il connettore deve gestire la traduzione dei messaggi di richiesta e risposta tra le integrazioni gestite per AWS IoT Device Management e la relativa piattaforma di terze parti.

Di seguito sono riportati i requisiti per il connettore C2C:

- Il server di autorizzazione di terze parti deve essere conforme agli standard OAuth 2.0 e alle configurazioni elencate in. OAuth requisiti di configurazione
- Sarà necessario un connettore C2C per interpretare gli identificatori delle AWS implementazioni del Matter Data Model e deve emettere le risposte e gli eventi conformi alle implementazioni del Matter Data Model. AWS Per ulteriori informazioni, consulta AWS implementazione del modello di dati Matter
- Un connettore C2C deve essere in grado di chiamare le integrazioni gestite per AWS IoT Device Management APIs con autenticazione. `SigV4` Per gli eventi asincroni inviati con l' SendConnectorEvent API, le stesse Account AWS credenziali utilizzate per registrare il connettore devono essere utilizzate per firmare la richiesta correlata. SendConnectorEvent
- Il connettore deve implementare le operazioni`AWS.ActivateUser`,`AWS.DiscoverDevices`, `AWS.SendCommand` e. `AWS.DeactivateUser`
- Quando il connettore C2C riceve eventi di terze parti relativi alle risposte ai comandi del dispositivo o all'individuazione del dispositivo, deve inoltrarli alle integrazioni gestite con l'API. `SendConnectorEvent` Per ulteriori informazioni su questi eventi e sull'`SendConnectorEvent`API, consulta. SendConnectorEvent

> **Note**
>
> L'`SendConnectorEvent`API fa parte dell'SDK per le integrazioni gestite e viene utilizzata al posto della creazione e della firma manuali delle richieste.

OAuth Requisiti 2.0 per il collegamento degli account

Ogni connettore C2C si basa su un server di autorizzazione OAuth 2.0 per autenticare gli utenti finali. Tramite questo server, gli utenti finali collegano i propri account di terze parti alla piattaforma del dispositivo del cliente. Il collegamento dell'account è il primo passaggio richiesto da un utente finale

per utilizzare i dispositivi supportati dal connettore C2C. Per ulteriori informazioni sui diversi ruoli nel collegamento degli account e OAuth nella versione 2.0, consulta. Ruoli di collegamento degli account

Sebbene il connettore C2C non richieda l'implementazione di una logica aziendale specifica per supportare il flusso di autorizzazione, il server di autorizzazione OAuth2 .0 associato al connettore C2C deve soddisfare i. OAuth requisiti di configurazione

> **Note**
>
> Le integrazioni gestite supportano AWS IoT Device Management solo la OAuth versione 2.0 con un flusso di codice di autorizzazione. Per ulteriori informazioni, consulta RFC 6749.

Il collegamento degli account è un processo che consente alle integrazioni gestite e al connettore di accedere ai dispositivi dell'utente finale utilizzando un token di accesso. Questo token fornisce integrazioni gestite per AWS IoT Device Management con l'autorizzazione dell'utente finale, in modo che il connettore possa interagire con i dati dell'utente finale tramite chiamate API. Per ulteriori informazioni, consulta Flusso di lavoro per il collegamento degli account.

Ti consigliamo di non registrare questi token sensibili in nessun registro. Se tuttavia sono archiviati nei log, ti consigliamo di utilizzare le politiche di protezione dei dati di CloudWatch Logs per mascherare i token contenuti nei log. Per ulteriori informazioni, consulta Incremento della protezione dei dati di log sensibili con il mascheramento.

Le integrazioni gestite per AWS IoT Device Management non ottengono direttamente un token di accesso, ma lo fanno tramite l'Authorization Code Grant Type. Innanzitutto, le integrazioni gestite per AWS IoT Device Management devono ottenere un codice di autorizzazione. Quindi scambia il codice con un token di accesso e un token di aggiornamento. Il token di aggiornamento viene utilizzato per richiedere un nuovo token di accesso alla scadenza del vecchio token di accesso. Se sia il token di accesso che il token di aggiornamento sono scaduti, è necessario eseguire nuovamente il flusso di collegamento degli account. È possibile farlo con l'operazione API. `StartAccountAssociationRefresh`

> **Important**
>
> Il token di accesso emesso deve essere definito per utente, ma non per OAuth client. Il token non deve fornire l'accesso a tutti i dispositivi di tutti gli utenti del client.
> Il server di autorizzazione deve eseguire una delle seguenti operazioni:

- Emetti token di accesso che contengono un ID estraibile dell'utente finale (proprietario della risorsa), come un token JWT.
- Restituisci l'ID dell'utente finale per ogni token di accesso emesso.

OAuth requisiti di configurazione

La tabella seguente illustra i parametri richiesti dal server di OAuth autorizzazione per le integrazioni gestite per AWS IoT Device Management per eseguire il collegamento degli account:

OAuth Parametri del server

Campo	Campo obbligatorio	Commento
`clientId`	Sì	Un identificatore pubblico per l'applicazione. Viene utilizzato per avviare flussi di autentica zione e può essere condiviso pubblicamente.
`clientSecret`	Sì	Una chiave segreta utilizzata per autenticare l'applicazione con il server di autorizzazione, soprattutto quando si scambia un codice di autorizzazione con un token di accesso. Deve essere mantenuta riservata e non condivisa pubblicamente.
`authorizationType`	Sì	Il tipo di autorizzazione supportato da questa configurazione di autorizza zione. Attualmente, "OAuth 2.0" è l'unico valore supportat o.
`authUrl`	Sì	L'URL di autorizzazione per il provider cloud di terze parti.

`tokenUrl`	Sì	L'URL del token per il provider di servizi cloud di terze parti.
`tokenEndpointAuthenticationScheme`	Sì	Schema di autenticazione di «HTTP_BASIC» o «REQUEST_BODY_CREDENTIALS». HTTP_BASIC segnala che le credenziali del client sono incluse nell'intestazione di autorizzazione, mentre il ladder segnala che sono incluse nel corpo della richiesta.

Il OAuth server utilizzato deve essere configurato in modo che i valori delle stringhe dei token di accesso siano codificati in Base64 con il set di caratteri UTF-8.

Ruoli di collegamento degli account

Per creare un connettore C2C, avrai bisogno di un server di autorizzazione OAuth 2.0 e di un collegamento all'account. Per ulteriori informazioni, consulta Flusso di lavoro per il collegamento degli account.

OAuth 2.0 definisce i seguenti quattro ruoli nell'implementazione del collegamento degli account:

1. Server di autorizzazione
2. Proprietario della risorsa (utente finale)
3. Server di risorse
4. Client

Di seguito vengono definiti ciascuno di questi OAuth ruoli:

Server di autorizzazione

Il server di autorizzazione è il server che identifica e autentica l'identità di un utente finale in un cloud di terze parti. I token di accesso forniti da questo server possono collegare l'account della

piattaforma cliente dell'utente AWS finale e l'account della piattaforma di terze parti. Questo processo è denominato collegamento dell'account.

Il server di autorizzazione supporta il collegamento degli account fornendo quanto segue:

- Visualizza una pagina di accesso per consentire all'utente finale di accedere al sistema. Questo viene in genere definito endpoint di autorizzazione.
- Autentica l'utente finale nel sistema.
- Genera un codice di autorizzazione che identifica l'utente finale.
- Trasmette il codice di autorizzazione alle integrazioni gestite per AWS IoT Device Management.
- Accetta il codice di autorizzazione dalle integrazioni gestite per AWS IoT Device Management e restituisce un token di accesso che le integrazioni gestite per AWS IoT Device Management possono utilizzare per accedere ai dati dell'utente finale nel tuo sistema. Questa operazione viene in genere completata tramite un URI separato, chiamato URI token o endpoint.

> **Important**
>
> Il server di autorizzazione deve supportare il flusso del codice di autorizzazione OAuth 2.0 da utilizzare con integrazioni gestite per AWS IoT Device Management Connector. le integrazioni gestite per AWS IoT Device Management supportano anche il flusso del codice di autorizzazione con Proof Key for Code Exchange (PKCE).
> Il server di autorizzazione deve:
>
> - Emetti token di accesso che contengono l'ID estraibile dell'utente finale o del proprietario della risorsa, ad esempio token JWT
> - Puoi restituire l'ID dell'utente finale per ogni token di accesso emesso
>
> In caso contrario, il connettore non sarà in grado di supportare l'operazione richiesta. `AWS.ActivateUser` Ciò impedirà l'utilizzo dei connettori con le integrazioni gestite.

Se lo sviluppatore o il proprietario del connettore non gestisce il proprio server di autorizzazione, il server di autorizzazione utilizzato deve fornire l'autorizzazione per le risorse gestite dalla piattaforma di terze parti dello sviluppatore del connettore. Ciò significa che tutti i token ricevuti dalle integrazioni gestite dal server di autorizzazione devono fornire limiti di sicurezza significativi sui dispositivi (la risorsa). Ad esempio, un token per utenti finali non consente comandi su un altro dispositivo dell'utente finale; le autorizzazioni fornite dal token sono mappate alle risorse all'interno della piattaforma. Considerate l'esempio di Lights Incorporated. Quando un utente finale avvia il flusso di collegamento dell'account con il proprio connettore, viene reindirizzato alla

pagina di accesso di Lights Incorporated che si trova davanti al server di autorizzazione. Dopo aver effettuato l'accesso e concesso le autorizzazioni al client, forniscono un token che consente al connettore di accedere alle risorse all'interno del proprio account Lights Incorporated.

Proprietario della risorsa (utente finale)

In qualità di proprietario della risorsa, consenti a integrazioni gestite per i clienti di AWS IoT Device Management di accedere alle risorse associate al tuo account eseguendo il collegamento dell'account. Ad esempio, si consideri la lampadina intelligente che un utente finale ha inserito nell'applicazione mobile Lights Incorporated. Il proprietario della risorsa si riferisce all'account dell'utente finale che ha acquistato e effettuato l'onboarding del dispositivo. Nel nostro esempio, il proprietario della risorsa è modellato come un account Lights Incorporated OAuth2 2.0. In qualità di proprietario della risorsa, questo account fornisce le autorizzazioni per emettere comandi e gestire il dispositivo.

Server di risorse

Questo è il server che ospita le risorse protette che richiedono l'autorizzazione all'accesso (dati del dispositivo). Il AWS cliente deve accedere alle risorse protette per conto di un utente finale e lo fa tramite integrazioni gestite per i connettori AWS IoT Device Management dopo il collegamento dell'account. Considerando ad esempio la lampadina intelligente di prima, il server di risorse è un servizio basato su cloud di proprietà di Lights Incorporated che gestisce la lampadina dopo l'installazione. Tramite il server di risorse, il proprietario della risorsa può impartire comandi alla lampadina intelligente, come accenderla e spegnerla. La risorsa protetta fornisce solo le autorizzazioni per l'account dell'utente finale e per altri account a accounts/entities cui potrebbe aver fornito l'autorizzazione.

Cliente

In questo contesto, il client è il tuo connettore C2C. Un client è definito come un'applicazione a cui viene concesso l'accesso alle risorse all'interno di un server di risorse per conto dell'utente finale. Il processo di collegamento dell'account rappresenta il connettore, il client, che richiede l'accesso alle risorse di un utente finale all'interno del cloud di terze parti.

Sebbene il connettore sia il OAuth client, le integrazioni gestite per AWS IoT Device Management eseguono operazioni per conto del connettore. Ad esempio, le integrazioni gestite per AWS IoT Device Management inoltrano richieste al server di autorizzazione per ottenere un token di accesso. Il connettore è ancora considerato il client perché è l'unico componente che accede alla risorsa protetta (dati del dispositivo) nel server di risorse.

Considerate la lampadina intelligente che è stata integrata da un utente finale. Una volta completato il collegamento dell'account tra la piattaforma del cliente e il server di autorizzazione di Lights Incorporated, il connettore stesso comunicherà con il server di risorse per recuperare informazioni sulla lampadina intelligente dell'utente finale. Il connettore può quindi ricevere comandi dall'utente finale. Ciò include l'accensione o lo spegnimento della luce per loro conto tramite il server di risorse Lights Incorporated. Pertanto, designiamo il connettore come client.

Flusso di lavoro per il collegamento degli account

Affinché le integrazioni gestite di un cliente per la piattaforma AWS IoT Device Management interagiscano con i dispositivi di un utente finale sulla piattaforma di terze parti tramite il connettore C2C, ottiene il token di accesso tramite il seguente flusso di lavoro:

1. Quando un utente avvia l'onboarding di dispositivi di terze parti tramite l'applicazione del cliente, le integrazioni gestite per AWS IoT Device Management restituiscono l'URI di autorizzazione e il. AssociationId
2. Il front-end dell'applicazione memorizza AssociationId e reindirizza l'utente finale alla pagina di accesso della piattaforma di terze parti.
    - L'utente finale accede. L'utente finale concede al client l'accesso ai dati del proprio dispositivo.
3. La piattaforma di terze parti crea un codice di autorizzazione. L'utente finale viene reindirizzato alle integrazioni gestite per l'URI di callback della piattaforma AWS IoT Device Management, incluso il codice allegato alla richiesta di reindirizzamento.
4. Le integrazioni gestite scambia questo codice con l'URI del token della piattaforma di terze parti.
5. L'URI del token convalida il codice di autorizzazione e restituisce un token di accesso OAuth2 .0 e un token di aggiornamento, associati all'utente finale.
6. Le integrazioni gestite richiamano il connettore C2C per completare il flusso di collegamento `AWS.ActivateUser` dell'account e ottenere. UserId
7. Le integrazioni gestite restituiscono OAuth RedirectUrl (dalla configurazione Connector Policy) la pagina di autenticazione riuscita all'applicazione del cliente.

> **Note**
>
> In caso di errori, le integrazioni gestite per AWS IoT Device Management aggiungono i parametri di query error ed error_description all'URL fornendo i dettagli dell'errore all'applicazione del cliente.

8. L'applicazione del cliente reindirizza l'utente finale a. OAuth RedirectUrl A questo punto il front-end dell'applicazione conosce AssociationId l'associazione sin dal primo passaggio.

   Tutte le richieste successive effettuate da integrazioni gestite per AWS IoT Device Management tramite il connettore C2C alla piattaforma cloud di terze parti, come i comandi per scoprire dispositivi e inviare comandi, includeranno il token di accesso OAuth2 .0.

Il diagramma seguente mostra la relazione tra i componenti chiave del collegamento degli account:

Implementa le operazioni di interfaccia del connettore C2C

Managed integrations for AWS IoT Device Management definisce quattro operazioni da gestire per qualificarsi come connettore. AWS Lambda Il connettore C2C deve implementare ognuna delle seguenti operazioni:

1. `AWS.ActivateUser`- Le integrazioni gestite per il AWS IoT Device Management servizio richiamano questa API per recuperare un identificatore utente univoco a livello globale associato al token .0 fornito. OAuth2 Questa operazione può essere utilizzata opzionalmente per eseguire eventuali requisiti aggiuntivi per il processo di collegamento dell'account.
2. `AWS.DiscoverDevices`- le integrazioni gestite per il servizio AWS IoT Device Management richiamano questa API al connettore per scoprire i dispositivi dell'utente
3. `AWS.SendCommand`- le integrazioni gestite per il servizio AWS IoT Device Management richiamano questa API al connettore per l'invio di comandi per i dispositivi dell'utente
4. `AWS.DeactivateUser`- le integrazioni gestite per il servizio AWS IoT Device Management richiamano questa API al connettore per disattivare il token di accesso dell'utente da decollegare nel server di autorizzazione.

Le integrazioni gestite per AWS IoT Device Management sempre richiamano la funzione Lambda con un payload di stringa JSON tramite l'azione. AWS Lambda `invokeFunction` Le operazioni di

richiesta devono includere un `operationName` campo in ogni payload della richiesta. Per ulteriori informazioni, consulta Invoke in AWS Lambda API Reference.

Ogni timeout di chiamata è impostato su due secondi e, se la chiamata fallisce, verrà ritentata cinque volte.

La Lambda implementata per il connettore analizzerà an `operationName` dal payload della richiesta e implementerà la funzionalità corrispondente da mappare al cloud di terze parti:

```
public ConnectorResponse handleRequest(final ConnectorRequest request)
        throws OperationFailedException {
    Operation operation;
    try {
        operation = Operation.valueOf(request.payload().operationName());
    } catch (IllegalArgumentException ex) {
        throw new ValidationException(
           "Unknown operation '%s'".formatted(request.payload().operationName()),
           ex
        );
    }

    return switch (operation) {
        case ActivateUser -> activateUserManager.activateUser(request);
        case DiscoverDevices -> deviceDiscoveryManager.listDevices(request);
        case SendCommand -> sendCommandManager.sendCommand(request);
        case DeactivateUser -> deactivateUser.deactivateUser(request);
    };
}
```

> **Note**
>
> Lo sviluppatore del connettore deve implementare le `deactivateUser.deactivateUser` operazioni`activateUserManager.activateUser(request)`, `deviceDiscoveryManager.listDevices(request)sendCommandManager.sendCommand(re` e elencate nell'esempio precedente.

L'esempio seguente descrive in dettaglio una richiesta di connettore generica proveniente da integrazioni gestite, in cui sono presenti campi comuni a tutte le interfacce richieste. Dall'esempio, puoi vedere che sono presenti sia un'intestazione di richiesta che un payload della richiesta. Le intestazioni di richiesta sono comuni a tutte le interfacce operative.

```
{
  "header": {
   "auth": {
    "token": “ashriu32yr97feqy7afsaf”,
    "type": “OAuth2.0"
   }
  },
  "payload":{
   "operationName": "AWS.SendCommand",
   "operationVersion": "1.0",
   "connectorId": “exampleId”,
  …
  }
}
```

Intestazioni di richiesta predefinite

I campi di intestazione predefiniti sono i seguenti.

```
{
    "header": {
        "auth": {
            "token": string,    // end user's Access Token
            "type": ENUM ["OAuth2.0"],
        }
    }
}
```

Qualsiasi API ospitata da un connettore deve elaborare i seguenti parametri di intestazione:

Intestazioni e campi predefiniti

Campo	Obbligatorio/facoltativo	Descrizione
`header:auth`	Sì	Informazioni di autorizza zione fornite dal costruttore di connettori C2C durante la registrazione dei connettori.
`header:auth:token`	Sì	Token di autorizzazione dell'utente generato dal provider di servizi cloud di

		terze parti e collegato a. `connectorAssociationID`
`header:auth:type`	Sì	Il tipo di autorizzazione necessaria.

> **Note**
>
> A tutte le richieste al connettore verrà allegato il token di accesso dell'utente finale. Si può presumere che il collegamento dell'account tra l'utente finale e il cliente delle integrazioni gestite sia già avvenuto.

Payload della richiesta

Oltre alle intestazioni comuni, ogni richiesta avrà un payload. Sebbene questo payload abbia campi unici per ogni tipo di operazione, ogni payload ha una serie di campi predefiniti che saranno sempre presenti.

Richiedi i campi del payload:

- `operationName`: L'operazione di una determinata richiesta, pari a uno dei seguenti valori:`AWS.ActivateUser`,, `AWS.SendCommandAWS.DiscoverDevices`,`AWS.DeactivateUser`.
- `operationVersion`: Ogni operazione è suddivisa in versioni per consentirne l'evoluzione nel tempo e fornire una definizione stabile dell'interfaccia per i connettori di terze parti. Le integrazioni gestite inseriscono un campo di versione nel payload di tutte le richieste.
- `connectorId`: L'ID del connettore a cui è stata inviata la richiesta.

Intestazioni di risposta predefinite

Ogni operazione risponderà con una risposta `ACK` alle integrazioni gestite per AWS IoT Device Management che confermerà che il connettore C2C ha ricevuto la richiesta e ha iniziato a elaborarla. Di seguito è riportato un esempio generico di tale risposta:

```
{
```

```
  "header":{
   "responseCode": 200
  },
  "payload":{
   "responseMessage": “Example response!”
  }
}
```

Ogni risposta operativa deve avere la seguente intestazione comune:

```
{
    "header": {
        "responseCode": Integer
    }
}
```

La tabella seguente elenca l'intestazione di risposta predefinita:

Intestazione e campo di risposta predefiniti

Campo	Obbligatorio/facoltativo	Commento
`header:responseCode`	Sì	ENUM di valori che indicano lo stato di esecuzione della richiesta.

Nelle varie interfacce di connettore e schemi API descritti in questo documento è presente un `responseMessage` campo or. `Message` Questo è un campo opzionale utilizzato dal connettore C2C Lambda per rispondere con qualsiasi contesto relativo alla richiesta e alla sua esecuzione. Preferibilmente, qualsiasi errore che dia origine a un codice di stato diverso da `200` dovrebbe includere un valore di messaggio che descriva l'errore.

Rispondi alle richieste di funzionamento del connettore C2C con l'API SendConnectorEvent

Le integrazioni gestite AWS IoT Device Management prevedono che il connettore si comporti in modo asincrono per ogni operazione. `AWS.SendCommand` `AWS.DiscoverDevices` Ciò significa che la risposta iniziale a queste operazioni «riconosce» semplicemente che il connettore C2C ha ricevuto la richiesta.

Utilizzando l'`SendConnectorEvent`API, il connettore dovrebbe inviare i tipi di eventi dall'elenco seguente a for `AWS.DiscoverDevices` and `AWS.SendCommand` operations, oltre a eventi proattivi del dispositivo (come l'accensione e lo spegnimento manuale di una luce). Per una spiegazione dettagliata di questi tipi di eventi e dei relativi casi d'uso, consulta Implementa AWS. DiscoverDevices operazioneImplementa AWS. SendCommand operazione, eInvia gli eventi del dispositivo con l'API SendConnectorEvent.

Ad esempio, se il connettore C2C riceve una `DiscoverDevices` richiesta, le integrazioni gestite per AWS IoT Device Management si aspettano che risponda in modo sincrono con il formato di risposta definito sopra. Quindi, devi richiamare l'`SendConnectorEvent`API con la struttura della richiesta definita in, per un evento DEVICE_DISCOVERY. Implementa AWS. DiscoverDevices operazione La chiamata `SendConnectorEvent` on API può avvenire ovunque tu abbia accesso alle credenziali Lambda del tuo connettore C2C. Account AWS Il flusso di rilevamento dei dispositivi non ha esito positivo finché le integrazioni gestite per AWS IoT Device Management non ricevono questo evento.

> **Note**
>
> In alternativa, la chiamata `SendConnectorEvent` API può avvenire prima della risposta alla chiamata Lambda del connettore C2C, se necessario. Tuttavia, questo flusso contraddice il modello asincrono per lo sviluppo del software.

- SendConnectorEvent- Il connettore chiama questa API managed integrations for AWS IoT Device Management per inviare eventi del dispositivo a integrazioni gestite per AWS IoT Device Management. Solo 3 tipi di eventi accettati dalle integrazioni gestite:
  - "DEVICE_DISCOVERY" — Questa operazione relativa all'evento deve essere utilizzata per inviare un elenco di dispositivi rilevati all'interno del cloud di terze parti per un token di accesso specifico.
  - «DEVICE_COMMAND_RESPONSE" — Questa operazione evento deve essere utilizzata per inviare un evento specifico del dispositivo come risultato dell'esecuzione del comando.
  - «DEVICE_EVENT" — Questa operazione evento deve essere utilizzata per qualsiasi evento proveniente dal dispositivo che non sia il risultato diretto di un comando basato sull'utente. Questo può servire come tipo di evento generale per segnalare in modo proattivo le modifiche o le notifiche dello stato del dispositivo.

Implementa AWS. ActivateUser operazione

L'`AWS.ActivateUser`operazione è necessaria per le integrazioni gestite per AWS IoT Device Management per recuperare un identificatore utente dal token .0 di OAuth2 un utente finale. Le integrazioni gestite per AWS IoT Device Management passeranno il OAuth token all'interno dell'intestazione della richiesta e si aspetta che il connettore includa l'identificatore utente univoco a livello globale nel payload di risposta. Questa operazione viene eseguita dopo un flusso di collegamento dell'account riuscito.

L'elenco seguente descrive i requisiti del connettore per facilitare il corretto flusso di `AWS.Activate` utenti.

- Il tuo connettore C2C Lambda può elaborare un messaggio di richiesta `AWS.ActivateUser` operativa proveniente da integrazioni gestite per AWS IoT Device Management.
- Il connettore C2C Lambda può determinare un identificatore utente univoco da un token .0 fornito. OAuth2 Normalmente, può essere estratto dal token stesso, se si tratta di un token JWT, o richiesto dal token al server di autorizzazione.

Flusso di lavoro di **`AWS.ActivateUser`**

1. Integrazioni gestite per AWS IoT Device Management richiamare il connettore C2C Lambda con il seguente payload:

```
{
   "header": {
        "auth": {
            "token": "ashriu32yr97feqy7afsaf",
            "type": "OAuth2.0"
        }
   },
   "payload": {
        "operationName": "AWS.ActivateUser",
        "operationVersion": "1.0.0",
        "connectorId": "Your-Connector-ID",
}
}
```

2. Il connettore C2C determina l'ID utente, dal token o interrogando il server di risorse di terze parti, da includere nella risposta. `AWS.ActivateUser`

3. Il connettore C2C risponde alla chiamata dell'operazione `AWS.ActivateUser` Lambda, incluso il payload predefinito e l'identificatore utente corrispondente all'interno del campo. `userId`

```
{
     "header": {
          "responseCode":200
     },
     "payload": {
          "responseMessage": "Successfully activated user with connector-id `Your-Connector-Id.”,
          "userId": "123456"
     }
}
```

Implementa AWS. DiscoverDevices operazione

Device Discovery allinea l'elenco dei dispositivi fisici di proprietà dell'utente finale con le rappresentazioni digitali di tali dispositivi degli utenti finali gestite nelle integrazioni gestite per AWS IoT Device Management. Viene eseguita da un AWS cliente su dispositivi di proprietà dell'utente finale solo dopo aver completato il collegamento dell'account tra l'utente e le integrazioni gestite per AWS IoT Device Management. Il rilevamento dei dispositivi è un processo asincrono in cui le integrazioni gestite per AWS IoT Device Management richiamano un connettore per avviare la richiesta di rilevamento dei dispositivi. Un connettore C2C restituisce un elenco di dispositivi degli utenti finali rilevati in modo asincrono con un identificatore di riferimento (denominato «) generato da integrazioni gestite. `deviceDiscoveryId`

Il diagramma seguente illustra il flusso di lavoro di rilevamento dei dispositivi tra l'utente finale e le integrazioni gestite per AWS IoT Device Management:

AWS. DiscoverDevices flusso di lavoro

1. Il cliente avvia il processo di scoperta del dispositivo per conto dell'utente finale.
2. Le integrazioni gestite per la AWS IoT Device Management generazione di un identificatore di riferimento richiesto `deviceDiscoveryId` per la richiesta di rilevamento del dispositivo generata dal Cliente. AWS

3. Integrazioni gestite per l' AWS IoT Device Management invio di una richiesta di rilevamento del dispositivo al connettore C2C utilizzando l'interfaccia `AWS.DiscoverDevices` operativa, inclusa una richiesta valida per OAuth `accessToken` l'utente finale e la. `deviceDiscoveryId`
4. Gli archivi del connettore `deviceDiscoveryId` devono essere inclusi nell'evento. `DEVICE_DISCOVERY` Questo evento conterrà anche un elenco dei dispositivi degli utenti finali scoperti e dovrà essere inviato alle integrazioni gestite per AWS IoT Device Management con l'`SendConnectorEvent`API come `DEVICE_DISCOVERY` evento.
5. Il connettore C2C chiamerà il server di risorse per recuperare tutti i dispositivi di proprietà dell'utente finale.
6. Il connettore C2C Lambda risponde all'invocazione Lambda `invokeFunction` () con la risposta ACK alle integrazioni gestite per AWS IoT Device Management, fungendo da risposta iniziale per l'operazione. `AWS.DiscoverDevices` Managed Integrations notifica al cliente con un ACK il processo di scoperta dei dispositivi avviato.
7. Il server di risorse ti invia un elenco di dispositivi posseduti e gestiti dall'utente finale.
8. Il connettore converte ogni dispositivo dell'utente finale nelle integrazioni gestite per il formato del dispositivo richiesto da AWS IoT Device Management`ConnectorDeviceId`, `ConnectorDeviceName` incluso un report sulle capacità per ogni dispositivo.
9. Il connettore C2C fornisce anche informazioni sul proprietario `UserId` dei dispositivi rilevati. Può essere recuperato dal server di risorse come parte dell'elenco dei dispositivi o in una chiamata separata a seconda dell'implementazione del server di risorse.
10. Successivamente, il connettore C2C chiamerà le integrazioni gestite per l'API AWS IoT Device Management`SendConnectorEvent`, tramite SigV4 utilizzando Account AWS credenziali e con il parametro operativo impostato come «DEVICE_DISCOVERY». Ogni dispositivo nell'elenco dei dispositivi inviati alle integrazioni gestite per AWS IoT Device Management sarà rappresentato da parametri specifici del dispositivo come`connectorDeviceId`, `connectorDeviceName` e a. `capabilityReport`

    - In base alla risposta del tuo server di risorse, devi notificare di conseguenza le integrazioni gestite per AWS IoT Device Management.

      Ad esempio, se il tuo server di risorse riceve una risposta in pagine all'elenco dei dispositivi rilevati per un utente finale, per ogni sondaggio puoi inviare un singolo evento `DEVICE_DISCOVERY` operativo, con un parametro di. `statusCode` `3xx` Se il rilevamento dei dispositivi è ancora in corso, ripeti i passaggi 5, 6 e 7.

11. Managed Integrations for AWS IoT Device Management invia una notifica al cliente in merito alla scoperta dei dispositivi dell'utente finale.
12. Se il connettore C2C invia un evento `DEVICE_DISCOVERY` operativo con il `statusCode` parametro aggiornato con un valore di 200, le integrazioni gestite notificheranno al cliente il completamento del flusso di lavoro di individuazione dei dispositivi.

> **Important**
>
> Se lo si desidera, i passaggi da 7 a 11 possono verificarsi prima del passaggio 6. Ad esempio, se la piattaforma di terze parti dispone di un'API per elencare i dispositivi di un utente finale, l'evento DEVICE_DISCOVERY può essere inviato `SendConnectorEvent` prima che il connettore C2C Lambda risponda con il tipico ACK.

Requisiti del connettore C2C per il rilevamento dei dispositivi

L'elenco seguente descrive i requisiti del connettore C2C per facilitare la corretta individuazione dei dispositivi.

- Il connettore C2C Lambda a può elaborare un messaggio di richiesta di rilevamento del dispositivo proveniente da integrazioni gestite per AWS IoT Device Management e gestirne l'operazione. `AWS.DiscoverDevices`
- Il connettore C2C può chiamare le integrazioni gestite per AWS IoT Device Management APIs tramite SigV4 utilizzando le credenziali utilizzate per la registrazione del Account AWS connettore.

Processo di scoperta dei dispositivi

I passaggi seguenti descrivono il processo di scoperta dei dispositivi con il connettore C2C e le integrazioni gestite per AWS IoT Device Management.

Processo di scoperta dei dispositivi

1. Le integrazioni gestite attivano il rilevamento dei dispositivi:
   - Invia una richiesta POST a `DiscoverDevices` con il seguente payload JSON:

```
/DiscoverDevices
{
   "header": {
```

```
        "auth": {
            "token": "ashriu32yr97feqy7afsaf",
            "type": "OAuth2.0"
        }
    },
    "payload": {
        "operationName": "AWS.DiscoverDevices",
        "operationVersion": "1.0",
        "connectorId": "Your-Connector-Id",
        "deviceDiscoveryId": "12345678"
    }
}
```

2. Il connettore conferma la scoperta:

   - Il connettore invia una conferma con la seguente risposta JSON:

```
{
     "header": {
          "responseCode":200
     },
     "payload": {
          "responseMessage": "Discovering devices for discovery-job-id
 '12345678' with connector-id `Your-Connector-Id`"
     }
}
```

3. Il connettore invia l'evento Device Discovery:

   - Invia una richiesta POST a `/connector-event/{your_connector_id}` con il seguente payload JSON:

```
AWS API - /SendConnectorEvent
URI - POST /connector-event/{your_connector_id}
{
   "UserId": "6109342",
   "Operation": "DEVICE_DISCOVERY",
   "OperationVersion": "1.0",
   "StatusCode": 200,
   "DeviceDiscoveryId": "12345678",
   "ConnectorId": "Your_connector_Id",
   "Message": "Device discovery for discovery-job-id '12345678' successful",
   "Devices": [
```

```
        {
            "ConnectorDeviceId": "Your_Device_Id_1",
            "ConnectorDeviceName": "Your-Device-Name",
            "CapabilityReport": {
          "nodeId":"1",
    "version":"1.0.0",
     "endpoints":[{
     "id":"1",
     "deviceTypes":["Camera"],
     "clusters":[{
      "id":"0x0006",
      "revision":1,
      "attributes":[{
       "id":"0x0000",
      }],
      "commands":["0x00","0x01"],
      "events":["0x00"]
     }]
    }]
   }
        }
    ]
}
```

Costruisci un CapabilityReport per l'evento DISCOVER_DEVICES

Come illustrato nella struttura degli eventi sopra definita, ogni dispositivo segnalato in un evento DISCOVER_DEVICES, che funge da risposta a un'`AWS.DiscoverDevices`operazione, richiederà una CapbilityReport descrizione delle funzionalità del dispositivo corrispondente. Un `CapabilityReport` indica le integrazioni gestite per le funzionalità dei dispositivi AWS IoT Device Management in un formato conforme a Matter. I seguenti campi devono essere forniti nel ``: CapabilityReport

- `nodeId`, String: identificatore per il nodo dei dispositivi contenente quanto segue `endpoints`
- `version`, String: versione di questo nodo del dispositivo, impostata dallo sviluppatore del connettore
- `endpoints`, Elenco<Cluster>: Elenco delle AWS implementazioni del Matter Data Model supportate da questo endpoint del dispositivo.
  - `id`, String: identificatore dell'endpoint impostato dallo sviluppatore del connettore

- `deviceTypes`, Elenco<String>: elenco dei tipi di dispositivi acquisiti da questo endpoint, ad esempio «Fotocamera».
- `clusters`, Elenco<Cluster>: Elenco delle AWS implementazioni del Matter Data Model supportate da questo endpoint.
  - `id`, String: identificatore del cluster come definito dallo standard Matter.
  - `revision`, Numero intero: numero di revisione del cluster definito dallo standard Matter.
  - `attributes`, Mappa<String, Object>: mappa degli identificatori degli attributi e dei corrispondenti valori di stato correnti del dispositivo, con identificatori e valori validi definiti dallo standard Matter.
    - `id`, String: ID dell'attributo definito dalle AWS implementazioni del Matter Data Model.
    - `value`, Oggetto: il valore corrente dell'attributo definito dall'attributo ID. Il tipo di «valore» può cambiare a seconda dell'attributo. Il `value` campo è facoltativo per ogni attributo e deve essere incluso solo se il connettore lambda è in grado di determinare lo stato corrente durante il rilevamento.
  - `commands`, Elenco<String>: Elenco dei comandi IDs supportati da questo cluster come definito dallo standard Matter.
  - `events`, Elenco<String>: Elenco degli eventi IDs supportati da questo cluster come definito dallo standard Matter.

Per l'elenco corrente delle funzionalità supportate e le relative AWS implementazioni del Matter Data Model, fate riferimento all'ultima versione della documentazione del modello di dati.

Implementa AWS. SendCommand operazione

L'`AWS.SendCommand`operazione consente alle integrazioni gestite per AWS IoT Device Management di inviare comandi avviati dall'utente finale tramite il AWS cliente al server di risorse. Il tuo server di risorse può supportare diversi tipi di dispositivi, ognuno dei quali ha il proprio modello di risposta. L'esecuzione dei comandi è un processo asincrono in cui le integrazioni gestite per AWS IoT Device Management inviano una richiesta di esecuzione del comando con un `TraceID`, che il connettore includerà in una risposta al comando inviata alle integrazioni gestite tramite la `` SendConnectorEvent API. Le integrazioni gestite per AWS IoT Device Management si aspettano che il server di risorse restituisca una risposta che confermi che il comando è stato ricevuto, ma non indica necessariamente che il comando è stato eseguito.

Il diagramma seguente illustra il flusso di esecuzione del comando con un esempio in cui l'utente finale tenta di accendere le luci della propria casa:

Workflow di esecuzione dei comandi del dispositivo

1. Un utente finale invia un comando per accendere una luce utilizzando l'applicazione del AWS cliente.
2. Il cliente inoltra le informazioni sui comandi alle integrazioni gestite per AWS IoT Device Management con le informazioni sul dispositivo dell'utente finale.
3. Le integrazioni gestite generano un «traceID» che il connettore utilizzerà per inviare le risposte ai comandi al servizio.
4. le integrazioni gestite per AWS IoT Device Management inviano la richiesta di comando al connettore, utilizzando l'interfaccia `AWS.SendCommand` operativa.
    - Il payload definito da questa interfaccia è costituito dall'identificatore del dispositivo, dai comandi del dispositivo formulati come Matterendpoints/clusters/commands, dal token di accesso dell'utente finale e da altri parametri richiesti.
5. Il connettore `traceId` memorizza i dati da includere nella risposta al comando.
    - Il connettore traduce la richiesta di comando di integrazioni gestite nel formato appropriato del server di risorse.
6. Il connettore riceve `UserId` dal token di accesso fornito dall'utente finale e lo associa al comando.
    a. `UserId`Può essere recuperato dal server di risorse utilizzando una chiamata separata o estratto dal token di accesso nel caso di JWT e token simili.
    b. L'implementazione dipende dal server di risorse e dai dettagli del token di accesso.
7. Il connettore richiama il server di risorse per «accendere» la luce dell'utente finale.
8. Il server di risorse interagisce con il dispositivo.
    a. Il connettore comunica alle integrazioni gestite per AWS IoT Device Management che il server di risorse ha fornito il comando, rispondendo con un ACK come risposta iniziale sincrona al comando.
    b. Le integrazioni gestite le inoltrano quindi all'applicazione del cliente.

9. Dopo che il dispositivo ha acceso la luce, l'evento del dispositivo viene catturato dal server di risorse.
10. Il server di risorse invia l'evento del dispositivo al connettore.
11. Il connettore trasforma l'evento del dispositivo generato dal server di risorse in un tipo di operazione di evento DEVICE_COMMAND_RESPONSE di integrazioni gestite.
12. Il connettore chiama l'API con l'operazione «DEVICE_COMMAND_RESPONSE`SendConnectorEvent`».
    - Allega le integrazioni `traceId` fornite da managed integrations per AWS IoT Device Management nella richiesta iniziale.
13. Le integrazioni gestite notificano al cliente la modifica dello stato del dispositivo dell'utente finale.
14. Il cliente notifica all'utente finale che la luce del dispositivo si è accesa.

> **Note**
>
> La configurazione del server di risorse determina la logica per la gestione dei messaggi di richiesta e risposta dei comandi del dispositivo non riusciti. Ciò include i tentativi di riprovare i messaggi utilizzando lo stesso ReferenceID per il comando.

Requisiti del connettore C2C per l'esecuzione dei comandi del dispositivo

L'elenco seguente descrive i requisiti del connettore C2C per facilitare la corretta esecuzione dei comandi del dispositivo.

- Il connettore C2C Lambda può elaborare i messaggi di richiesta `AWS.SendCommand` operativa provenienti da integrazioni gestite per AWS IoT Device Management.
- Il connettore C2C deve tenere traccia dei comandi inviati al server di risorse e mapparli con il `traceID` appropriato.
- Puoi chiamare integrazioni gestite per le API del servizio AWS IoT Device Management tramite SigV4 utilizzando AWS le credenziali Account AWS utilizzate per la registrazione del connettore C2C.

1. Le integrazioni gestite inviano il comando al connettore (fare riferimento al passaggio 4 nel diagramma precedente).

- ```
  /Send-Command
  {
       "header": {
            "auth": {
                "token": "ashriu32yr97feqy7afsaf",
                "type": "OAuth2.0"
            }
       },
       "payload": {
            "operationName": "AWS.SendCommand",
            "operationVersion": "1.0",
            "connectorId": "Your-Connector-Id",
            "connectorDeviceId": "Your_Device_Id",
            "traceId": "traceId-3241u78123419",
            "endpoints": [{
                "id": "1",
                "clusters": [{
                    "id": "0x0202",
                    "commands": [{
                        "0xff01":
                            {
                                "0x0000": "3”
                      }
                    }]
                }]
            }]
       }
    }
  ```

2. Comando ACK del connettore C2C (fare riferimento al passaggio 7 del diagramma precedente in cui il connettore invia ACK alle integrazioni gestite per AWS IoT Device Management Service).

- ```
  {
       "header":{
            "responseCode":200
       },
       "payload":{
            "responseMessage": "Successfully received send-command request for
   connector 'Your-Connector-Id' and connector-device-id 'Your_Device_Id'"
       }
    }
  ```

3. Il connettore invia l'evento Device Command Response (fare riferimento al passaggio 11 nel diagramma precedente).

   - ```
     AWS-API: /SendConnectorEvent
     URI: POST /connector-event/{Your-Connector-Id}

     {
        "UserId": "End-User-Id",
        "Operation": "DEVICE_COMMAND_RESPONSE",
        "OperationVersion": "1.0",
        "StatusCode": 200,
        "Message": “Example message”,
        "ConnectorDeviceId": "Your_Device_Id",
        "TraceId": "traceId-3241u78123419",
        "MatterEndpoint": {
             "id": "1",
             "clusters": [{
                 "id": "0x0202",
                 "attributes": [
                     {
                         "0x0000": “3”
                     }
                 ],
                 "commands": [
                     "0xff01":
                     {
                         "0x0000": "3”
                     }
        ]
             }]
         }
     }
     ```

   > **Note**
   >
   > Le modifiche allo stato del dispositivo a seguito dell'esecuzione di un comando non si rifletteranno nelle integrazioni gestite per AWS IoT Device Management fino a quando l'evento DEVICE_COMMAND_RESPONSE corrispondente non sarà stato ricevuto tramite l'API. SendConnectorEvent Ciò significa che fino a quando le integrazioni gestite non riceveranno l'evento della fase 3 precedente, indipendentemente dal fatto
     ```

che la risposta alla chiamata del connettore indichi l'esito positivo o meno, lo stato del dispositivo non verrà aggiornato.

Interpretazione degli «endpoint» della materia inclusa in AWS. SendCommand richiesta

Le integrazioni gestite utilizzeranno le funzionalità del dispositivo riportate durante l'individuazione del dispositivo per determinare quali comandi un dispositivo può accettare. Ogni funzionalità del dispositivo è modellata attraverso AWS implementazioni del Matter Data Model; pertanto, tutti i comandi in entrata verranno derivati dal campo `commands` all'interno di un determinato cluster. È responsabilità del connettore analizzare il campo `endpoints`, determinare il comando Matter corrispondente e tradurlo in modo che il comando corretto raggiunga il dispositivo. In genere, ciò significa tradurre il modello di dati Matter nelle relative richieste API.

Dopo l'esecuzione del comando, il connettore determina quali `attributi` definiti dalle AWS implementazioni del Matter Data Model sono cambiati di conseguenza. Queste modifiche vengono quindi segnalate alle integrazioni gestite per AWS IoT Device Management tramite eventi API DEVICE_COMMAND_RESPONSE inviati con l'API. `SendConnectorEvent`

Considera il campo `endpoints` incluso nel payload di esempio seguente: `AWS.SendCommand`

```
"endpoints": [{
    "id": "1",
    "clusters": [{
        "id": "0x0202",
        "commands": [{
            "0xff01":
                {
                    "0x0000": "3”
          }
        }]
    }]
}]
```

Da questo oggetto, il connettore può determinare quanto segue:

1. Imposta le informazioni sull'endpoint e sul cluster:
    a. Imposta l'endpoint `id` su «1".

> **Note**
>
> Se un dispositivo definisce più endpoint, ad esempio un singolo cluster, questo id viene utilizzato per indirizzare il comando alla funzionalità corretta. On/Off) can control multiple capabilities (i.e. turn a light on/off as well as turning a strobe on/off

   b. Imposta il cluster `id` su «0x0202" (cluster Fan Control).

2. Imposta le informazioni sul comando:

   a. Imposta l'identificatore del comando su «0xff01" (comando Update State definito da). AWS

   b. Aggiorna gli identificatori degli attributi inclusi con i valori forniti nella richiesta.

3. Aggiorna l'attributo:

   a. Imposta l'identificatore dell'attributo su «0x0000" (FanMode attributo del Fan Control Cluster).

   b. Imposta il valore dell'attributo su «3" (alta velocità della ventola).

Managed integrations ha definito due tipi di comandi «personalizzati» che non sono strettamente definiti dalle AWS implementazioni del Matter Data Model: i ReadState comandi and. UpdateState Per ottenere e impostare gli attributi del cluster definiti da Matter, le integrazioni gestite invieranno al connettore una `AWS.SendCommand` richiesta con il comando IDs relativo a UpdateState (id: 0xff01) o ReadState (id: 0xff02), con i parametri corrispondenti degli attributi che devono essere aggiornati o letti. Questi comandi possono essere richiamati per QUALSIASI tipo di dispositivo per attributi impostati come modificabili (aggiornabili) o recuperabili (leggibili) dalla corrispondente implementazione del Matter Data Model. AWS

Invia gli eventi del dispositivo con l'API SendConnectorEvent

Panoramica degli eventi avviati dal dispositivo

Sebbene l'`SendConnectorEvent`API venga utilizzata per rispondere in modo asincrono alle `AWS.DiscoverDevices` operazioni `AWS.SendCommand` e alle operazioni, viene anche utilizzata per notificare alle integrazioni gestite eventuali eventi avviati dal dispositivo. Gli eventi avviati dal dispositivo possono essere definiti come qualsiasi evento generato da un dispositivo senza un comando avviato dall'utente. Questi eventi del dispositivo possono includere, a titolo esemplificativo, modifiche dello stato del dispositivo, rilevamento del movimento, livelli della batteria e altro ancora.

Puoi inviare questi eventi alle integrazioni gestite utilizzando l'`SendConnectorEvent`API con l'operazione DEVICE_EVENT.

La sezione seguente utilizza una telecamera intelligente installata in una casa come esempio per spiegare ulteriormente il flusso di lavoro di questi eventi:

Workflow degli eventi del dispositivo

1. La videocamera rileva un movimento per il quale genera un evento che viene inviato al server di risorse.
2. Il server di risorse elabora l'evento e lo invia al connettore C2C.
3. Il connettore traduce questo evento nelle integrazioni gestite per l'interfaccia AWS IoT Device Management`DEVICE_EVENT`.
4. Il connettore C2C invia questo evento del dispositivo alle integrazioni gestite utilizzando l'`SendConnectorEvent`API con Operation impostata su «DEVICE_EVENT».
5. Le integrazioni gestite identificano il cliente interessato e trasmettono questo evento al cliente.
6. Il cliente riceve questo evento e lo visualizza all'utente tramite un identificatore utente.

Per ulteriori informazioni sul funzionamento delle `SendConnectorEvent` API, consulta `SendConnectorEvent` la Guida di riferimento alle integrazioni gestite per l'API AWS IoT Device Management.

Requisiti per gli eventi avviati dal dispositivo

Di seguito sono riportati alcuni requisiti per gli eventi avviati dal dispositivo.

- La risorsa del connettore C2C dovrebbe essere in grado di ricevere eventi asincroni del dispositivo dal server di risorse
- La risorsa del connettore C2C dovrebbe essere in grado di richiamare integrazioni gestite per le API del servizio AWS IoT Device Management tramite SigV4 utilizzando AWS le credenziali Account AWS utilizzate per la registrazione del connettore C2C.

L'esempio seguente mostra un connettore che invia un evento originato dal dispositivo tramite l'API: SendConnectorEvent

```
AWS-API: /SendConnectorEvent
```

```
URI: POST /connector-event/{Your-Connector-Id}

{
   "UserId": "Your-End-User-ID",
   "Operation": "DEVICE_EVENT",
   "OperationVersion": "1.0",
   "StatusCode": 200,
   "Message": None,
   "ConnectorDeviceId": "Your_Device_Id",
   "MatterEndpoint": {
        "id": "1",
        "clusters": [{
            "id": "0x0202",
            "attributes": [
                {
                    "0x0000": "3"
                }
            ]
        }]
    }]
}
```

Dall'esempio seguente, vediamo quanto segue:

- Proviene dall'endpoint del dispositivo con ID uguale a 1.
- La funzionalità del dispositivo a cui si riferisce questo evento ha un ID cluster di 0x0202, relativo al cluster Fan Control matter.
- L'attributo che è stato modificato ha l'ID 0x000, relativo al Fan Mode Enum all'interno del cluster. È stato aggiornato al valore 3, relativo al valore High.
- Poiché `connectorId` è un parametro restituito dal servizio cloud al momento della creazione, i connettori devono eseguire query utilizzando GetCloudConnector e filtrare per`lambdaARN`. Il file di lambda ARN viene interrogato tramite l'API. `Lambda.get_function_url_config` Ciò consente l'`CloudConnectorId`accesso dinamico in lambda e non la configurazione statica come in precedenza.

Implementa AWS. DeactivateUser operazione

panoramica sulla disattivazione degli utenti

La disattivazione dei token di accesso utente forniti è necessaria quando un cliente elimina il proprio account AWS cliente o quando un utente finale desidera scollegare il proprio account nel sistema dal sistema del cliente. AWS In entrambi i casi d'uso, le integrazioni gestite devono facilitare questo flusso di lavoro utilizzando il connettore C2C.

L'immagine seguente illustra la rimozione del collegamento di un account utente finale dal sistema

Flusso di lavoro di disattivazione degli utenti

1. L'utente avvia il processo di decollegamento tra l'account del AWS cliente e il server di autorizzazione di terze parti associato al connettore C2C.
2. Il cliente avvia l'eliminazione dell'associazione dell'utente tramite integrazioni gestite per AWS IoT Device Management.
3. Le integrazioni gestite avviano il processo di disattivazione tramite richiesta al connettore utilizzando l'interfaccia operativa. `AWS.DeactivateUser`
    - Il token di accesso /user è incluso nell'intestazione della richiesta.
4. Il connettore C2C accetta la richiesta e richiama il server di autorizzazione per revocare il token e qualsiasi accesso che fornisce.
    - Ad esempio, gli eventi di un account utente non collegato non devono più essere inviati alle integrazioni gestite dopo l'esecuzione. `AWS.DeactivateUser`
5. Il server di autorizzazione revoca l'accesso e invia una risposta al connettore C2C.
6. Il connettore C2C invia alle integrazioni gestite per AWS IoT Device Management un ACK che indica che il token di accesso dell'utente è stato revocato.
7. Le integrazioni gestite eliminano tutte le risorse di proprietà dell'utente finale che erano associate al tuo server di risorse.
8. Le integrazioni gestite inviano un ACK al cliente, indicando che tutte le associazioni relative al sistema sono state eliminate.
9. Il cliente notifica all'utente finale che il suo account è stato scollegato dalla piattaforma.

AWS. DeactivateUser requisiti

- La funzione Lambda del connettore C2C riceve un messaggio di richiesta dalle integrazioni gestite per gestire l'operazione. `AWS.DeactivateUser`
- Il connettore C2C deve revocare il token OAuth2.0 fornito e il token di aggiornamento corrispondente dell'utente all'interno del server di autorizzazione.

Di seguito è riportato un esempio di `AWS.DeactivateUser` richiesta che il connettore riceverà:

```
{
     "header": {
         "auth": {
             "token": "ashriu32yr97feqy7afsaf",
             "type": "OAuth2.0"

         }
     },
     "payload":{
         "operationName": "AWS.DeactivateUser"
         "operationVersion": "1.0"
         "connectorId": "Your-connector-Id"
     }
 }
```

Invoca il tuo connettore C2C

AWS Lambda consente politiche basate sulle risorse per autorizzare chi può richiamare una Lambda. Poiché le integrazioni gestite per AWS IoT Device Management sono una Servizio AWS, è necessario consentire alle integrazioni gestite di richiamare il connettore C2C Lambda tramite la policy delle risorse.

Allega una politica delle risorse con almeno le seguenti autorizzazioni minime al tuo connettore C2C Lambda. Ciò fornisce integrazioni gestite con i privilegi di invoke della funzione Lambda. Questa politica include una `Condition` chiave per aiutarti a limitare l'usabilità dei tuoi `connectorId` dati ai soli utenti previsti.

JSON

```
{
  "Version":"2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "Your-Desired-Policy-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "iotmanagedintegrations.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:ca-central-1:444455556666:function:connector-
lambda-name",
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:iotmanagedintegrations:ca-
central-1:444455556666:account-association/account-association-id"
        }
      }
    }
  ]
}
```

Aggiungi le autorizzazioni al tuo ruolo IAM

Tutte le integrazioni gestite APIs richiedono l'autenticazione AWS SigV4 per essere invocate. SigV4 è un protocollo di firma per autenticare AWS le richieste API utilizzando le tue credenziali. Account AWS Il ruolo IAM utilizzato per richiamare le integrazioni gestite APIs deve disporre delle seguenti autorizzazioni per poter richiamare correttamente: APIs

```
"Version": "2012-10-17",
"Statement": [
{
 "Sid": "Statement1",
 "Effect": "Allow",
 "Action": [
  "iotmanagedintegrations:Your-Required-Actions"
 ],
 "Resource": [
```

```
    "Your-Resource"
   ]
  }]
}
```

Per ulteriori informazioni sull'aggiunta di queste autorizzazioni, contatta. Supporto

Risorse aggiuntive

Per registrare il connettore C2C, è necessario quanto segue:

- L'ARN Lambda che designa il connettore che desideri registrare.

Testa manualmente il tuo connettore C2C

Per testare manualmente il connettore C2C end-to-end, è necessario simulare sia il cliente che l'utente finale.

Avrai bisogno delle seguenti risorse:

- Un AWS Lambda ARN che indica il connettore da testare.
- Un account utente di testing OAuth 2.0 dalla tua piattaforma cloud.
- Un connettore registrato con integrazioni gestite per AWS IoT Device Management. Per ulteriori informazioni, consulta Usa un connettore C2C (Cloud-to-Cloud).

Usa un connettore C2C (Cloud-to-Cloud)

Un connettore C2C gestisce la traduzione dei messaggi di richiesta e risposta e consente la comunicazione tra le integrazioni gestite e il cloud di un fornitore terzo. Facilita il controllo unificato su diversi tipi di dispositivi, piattaforme e protocolli, consentendo l'onboarding e la gestione di dispositivi di terze parti.

La procedura seguente elenca i passaggi per utilizzare il connettore C2C.

Passaggi per utilizzare il connettore C2C:

1. CreateCloudConnector

   Configura un connettore per abilitare la comunicazione bidirezionale tra le integrazioni gestite e i cloud di fornitori di terze parti.

Durante la configurazione del connettore, fornisci i seguenti dettagli:

- Nome: scegli un nome descrittivo per il connettore.
- Descrizione: Fornisci un breve riepilogo dello scopo e delle funzionalità del connettore.
- AWS Lambda ARN: specifica l'Amazon Resource Name (ARN) della AWS Lambda funzione che alimenterà il connettore.

Crea e distribuisci una AWS Lambda funzione che comunichi con un fornitore APIs terzo per creare un connettore. Successivamente, richiama l'CreateCloudConnectorAPI all'interno delle integrazioni gestite e fornisci la AWS Lambda funzione ARN per la registrazione. Assicurati che la AWS Lambda funzione sia implementata nello stesso AWS account in cui crei il connettore nelle integrazioni gestite. Ti verrà assegnato un Connector ID univoco per identificare l'integrazione.

Esempio di richiesta e risposta CreateCloudConnector API:

```
Request:

{
    "Name": "CreateCloudConnector",
    "Description": "Testing for C2C",
    "EndpointType": "LAMBDA",
    "EndpointConfig": {
        "lambda": {
            "arn": "arn:aws:lambda:us-east-1:xxxxxx:function:TestingConnector"
        }
    },
    "ClientToken": "abc"
}

Response:

{
    "Id": "string"
}
```

Flusso di creazione:

> **Note**
>
> Usa GetCloudConnector, UpdateCloudConnectorDeleteCloudConnector, e ListCloudConnectors APIs se necessario per questa procedura.

2. CreateConnectorDestination

Configura Destinazioni per fornire le impostazioni e le credenziali di autenticazione necessarie ai connettori per stabilire connessioni sicure con i cloud di fornitori di terze parti. Utilizza Destinations per registrare le credenziali di autenticazione di terze parti con integrazioni gestite, ad esempio i dettagli di autorizzazione OAuth 2.0, tra cui l'URL di autorizzazione, lo schema di autenticazione e la posizione delle credenziali all'interno. Gestione dei segreti AWS

Prerequisiti

Prima di creare un ConnectorDestination, devi:

- Chiama l'CreateCloudConnectorAPI per creare un connettore. L'ID restituito dalla funzione viene utilizzato nella chiamata CreateConnectorDestinationAPI API.
- Recupera il `tokenUrl` file per la piattaforma 3P del connettore. (Puoi scambiare un AuthCode con un AccessToken).
- Recupera l'AuthURL per la piattaforma 3P del connettore. (Gli utenti finali possono autenticarsi utilizzando nome utente e password).
- Usa `clientId` e `clientSecret` (dalla piattaforma 3P) nel gestore segreto del tuo account.

Esempio di richiesta e risposta CreateConnectorDestination API:

```
Request:

{
    "Name": "CreateConnectorDestination",
    "Description": "CreateConnectorDestination",
    "AuthType": "OAUTH",
    "AuthConfig": {
        "oAuth": {
            "authUrl": "https://xxxx.com/oauth2/authorize",
            "tokenUrl": "https://xxxx/oauth2/token",
            "scope": "testScope",
```

```
            "tokenEndpointAuthenticationScheme": "HTTP_BASIC",
            "oAuthCompleteRedirectUrl": "about:blank",
            "proactiveRefreshTokenRenewal": {
                "enabled": false,
                "DaysBeforeRenewal": 30
            }
        }
    },
    "CloudConnectorId": "<connectorId>", // The connectorID instance from response
 of Step 1.
    "SecretsManager": {
        "arn": "arn:aws:secretsmanager:*****:secret:*******",
        "versionId": "********"
    },
    "ClientToken": "***"
}

Response:

{
    "Id":"string"
}
```

Flusso di creazione di destinazioni cloud:

> **Note**
>
> Utilizzare GetCloudConnector, UpdateCloudConnectorDeleteCloudConnector, e ListCloudConnectors APIs se necessario per questa procedura.

3. CreateAccountAssociation

Le associazioni rappresentano le relazioni tra gli account cloud di terze parti degli utenti finali e una destinazione del connettore. Dopo aver creato un'associazione e aver collegato gli utenti finali alle integrazioni gestite, i loro dispositivi sono accessibili tramite un ID di associazione univoco. Questa integrazione abilita tre funzioni chiave: scoperta dei dispositivi, invio di comandi e ricezione di eventi.

Prerequisiti

Prima di creare un AccountAssociation, è necessario completare quanto segue:

- Chiama l'CreateConnectorDestinationAPI per creare una destinazione. L'ID restituito dalla funzione viene utilizzato nella chiamata CreateAccountAssociationAPI.
- Richiama l'CreateAccountAssociationAPI.

Esempio di richiesta e risposta CreateAccountAssociation API:

```
Request:

{
    "Name": "CreateAccountAssociation",
    "Description": "CreateAccountAssociation",
    "ConnectorDestinationId": "<destinationId>", //The destinationID from
 destination creation.
    "ClientToken": "***"
}

Response:

{
    "Id":"string"
}
```

> **Note**
>
> Utilizzare GetCloudConnector, UpdateCloudConnectorDeleteCloudConnector, e ListCloudConnectors APIs se necessario per questa procedura.

An AccountAssociationha uno stato che viene interrogato da GetAccountAssociatione. ListAccountAssociations APIs Questi APIs mostrano lo stato dell'Associazione. L'StartAccountAssociationRefreshAPI consente l'aggiornamento di uno AccountAssociationstato alla scadenza del relativo token di aggiornamento.

4. Individuazione dei dispositivi

Ogni elemento gestito è collegato a dettagli specifici del dispositivo, come il numero di serie e un modello di dati. Il modello di dati descrive la funzionalità del dispositivo, indicando se si tratta

di una lampadina, un interruttore, un termostato o un altro tipo di dispositivo. Per scoprire un dispositivo 3P e creare un ManagedThing per il dispositivo 3P, è necessario seguire i passaggi seguenti in sequenza.

a. Chiama StartDeviceDiscoveryl'API per avviare il processo di scoperta del dispositivo.

   Esempio di richiesta e risposta StartDeviceDiscovery API:

```
Request:

{
    "DiscoveryType": "CLOUD",
    "AccountAssociationId": "*****",
    "ClientToken": "abc"
}

Response:

{
    "Id": "string",
    "StartedAt": number
}
```

b. Richiama GetDeviceDiscoveryl'API per verificare lo stato del processo di scoperta.

c. Invoca ListDiscoveredDevicesl'API per elencare i dispositivi rilevati.

   Esempio di richiesta e risposta ListDiscoveredDevices API:

```
Request:

//Empty body

Response:

{
    "Items": [
    {
      "Brand": "string",
      "ConnectorDeviceId": "string",
      "ConnectorDeviceName": "string",
      "DeviceTypes": [ "string" ],
      "DiscoveredAt": number,
```

```
      "ManagedThingId": "string",
      "Model": "string",
      "Modification": "string"
    }
],
    "NextToken": "string"
}
```

d. Richiama l'CreateManagedThingAPI per selezionare i dispositivi dall'elenco di rilevamento da importare nelle integrazioni gestite.

   Esempio di richiesta e risposta CreateManagedThing API:

```
Request:

{
    "Role": "DEVICE",
    "AuthenticationMaterial": "CLOUD:XXXX:<connectorDeviceId1>",
    "AuthenticationMaterialType": "DISCOVERED_DEVICE",
    "Name": "sample-device-name"
    "ClientToken": "xxx"
}

Response:

{
   "Arn": "string", // This is the ARN of the managedThing
   "CreatedAt": number,
   "Id": "string"
}
```

e. Invoca GetManagedThingl'API per visualizzare questa nuova creazione`managedThing`. Lo stato sarà`UNASSOCIATED`.

f. Invoca RegisterAccountAssociationl'API per associarlo `managedThing` a uno specifico`accountAssociation`. Al termine di un'RegisterAccountAssociationAPI di successo, lo `ASSOCIATED` stato `managedThing` cambia.

   Esempio di richiesta e risposta RegisterAccountAssociation API:

```
Request:

{
```

```
    "AccountAssociationId": "string",
    "DeviceDiscoveryId": "string",
    "ManagedThingId": "string"
}

Response:

{
    "AccountAssociationId": "string",
    "DeviceDiscoveryId": "string",
    "ManagedThingId": "string"
}
```

5. Invia un comando al dispositivo 3P

   Per controllare un dispositivo appena installato, utilizza l'SendManagedThingCommandAPI, con l'Association ID creato in precedenza e un'azione di controllo basata sulla funzionalità supportata dal dispositivo. Il connettore utilizza le credenziali memorizzate dal processo di collegamento dell'account, per autenticarsi con il cloud di terze parti e richiamare la chiamata API pertinente per l'operazione.

   Esempio di richiesta e risposta SendManagedThingCommand API:

```
Request:

{
    "AccountAssociationId": "string",
    "ConnectorAssociationId": "string",
    "Endpoints": [
       {
          "capabilities": [
             {
                "actions": [
                   {
                      "actionTraceId": "string",
                      "name": "string",
                      "parameters": JSON value,
                      "ref": "string"
                   }
                ],
                "id": "string",
                "name": "string",
                "version": "string"
```

```
            }
         ],
         "endpointId": "string"
      }
   ]
}

Response:

{
   "TraceId": "string"
}
```

Invia il comando al flusso del dispositivo 3P:

6. Il connettore invia eventi alle integrazioni gestite

   L'SendConnectorEventAPI acquisisce quattro tipi di eventi dal connettore alle integrazioni gestite, rappresentati dai seguenti valori enum per il parametro Operation Type:

   - DEVICE_COMMAND_RESPONSE: la risposta asincrona che il connettore invia in risposta a un comando.
   - DEVICE_DISCOVERY: in risposta a un processo di rilevamento dei dispositivi, il connettore invia l'elenco dei dispositivi rilevati alle integrazioni gestite, utilizza l'API. SendConnectorEvent
   - DEVICE_EVENT: invia gli eventi del dispositivo ricevuti.
   - DEVICE_COMMAND_REQUEST: richieste di comando avviate dal dispositivo. Ad esempio, i flussi di lavoro WebRTC.

   Il connettore può anche inoltrare gli eventi del dispositivo utilizzando l'SendConnectorEventAPI, con un parametro opzionale. `userId`

   - Per gli eventi dei dispositivi con`userId`:

     Esempio di richiesta e risposta SendConnectorEvent API:

     ```
     Request:

     {
         "UserId": "*****",
     ```

```
    "Operation": "DEVICE_EVENT",
    "OperationVersion": "1.0",
    "StatusCode": 200,
    "ConnectorId": "****",
    "ConnectorDeviceId": "***",
    "TraceId": "***",
    "MatterEndpoint": {
        "id": "**",
        "clusters": [{
            .....
            }
        }]
    }
}

Response:

{
    "ConnectorId": "string"
}
```

- Per gli eventi relativi ai dispositivi senza`userId`:

  Esempio di richiesta e risposta SendConnectorEvent API:

```
Request:

{
    "Operation": "DEVICE_EVENT",
    "OperationVersion": "1.0",
    "StatusCode": 200,
    "ConnectorId": "*****",
    "ConnectorDeviceId": "****",
    "TraceId": "****",
    "MatterEndpoint": {
        "id": "**",
        "clusters": [{
            ....
        }]
    }
}

Response:
```

```
{
    "ConnectorId": "string"
}
```

Per rimuovere il collegamento tra una determinata associazione `managedThing` e un'associazione di account, utilizza il meccanismo di annullamento della registrazione:

Esempio di richiesta e risposta DeregisterAccountAssociation API:

```
Request:

{
    "AccountAssociationId": "****",
    "ManagedThingId": "****"
}

Response:

HTTP/1.1 200 // Empty body
```

Invia flusso di eventi:

7. Aggiorna lo stato del connettore su «In elenco» per renderlo visibile agli altri clienti delle integrazioni gestite

   Per impostazione predefinita, i connettori sono privati e visibili solo all' AWS account che li ha creati. Puoi scegliere di rendere visibile un connettore agli altri clienti di Managed Integrations.

   Per condividere il connettore con altri utenti, utilizza l'opzione Rendi visibile nella Console di gestione AWS pagina dei dettagli del connettore per inviare il tuo ID del connettore AWS per la revisione. Una volta approvato, il connettore è disponibile per tutti gli utenti delle integrazioni gestite all'interno dello stesso Regione AWS dispositivo. Inoltre, puoi limitare l'accesso a un AWS account specifico IDs modificando la politica di accesso sulla funzione associata AWS Lambda al connettore. Per garantire che il connettore sia utilizzabile da altri clienti, gestisci le autorizzazioni di accesso IAM sulla tua funzione Lambda da AWS altri account al tuo connettore visibile.

Consulta i Servizio AWS termini e le politiche della tua organizzazione che regolano la condivisione dei connettori e le autorizzazioni di accesso prima di rendere i connettori visibili agli altri clienti delle integrazioni gestite.

Integrazioni gestite Hub SDK

Utilizza gli argomenti di questa sezione per scoprire come effettuare l'onboarding e il controllo dei dispositivi hub IoT utilizzando l'SDK Hub per le integrazioni gestite. Per ulteriori informazioni sulle integrazioni gestite End device SDK, consulta il. Integrazioni gestite SDK per dispositivi finali

Architettura Hub SDK

Onboarding dei dispositivi

Scopri in che modo i componenti Hub SDK supportano l'onboarding dei dispositivi prima di iniziare a lavorare con le integrazioni gestite. Questa sezione illustra i componenti architettonici essenziali necessari per l'onboarding dei dispositivi, incluso il modo in cui il core provisioner e i plug-in specifici del protocollo interagiscono per gestire l'autenticazione, la comunicazione e la configurazione degli utenti dei dispositivi.

Componenti Hub SDK per l'onboarding dei dispositivi

Componenti SDK

- Fornitore principale
- Plugin di provisioning specifici per il protocollo
- Middleware specifico per il protocollo

Fornitore principale

Il core provisioner è il componente centrale che orchestra l'onboarding dei dispositivi nell'implementazione dell'hub IoT. Coordina tutte le comunicazioni tra le integrazioni gestite e i plug-in di provisioning specifici del protocollo, garantendo un onboarding sicuro e affidabile dei dispositivi. Quando si effettua l'onboard di un dispositivo, il core provisioner gestisce il flusso di autenticazione, gestisce la messaggistica MQTT ed elabora le richieste dei dispositivi tramite le seguenti funzioni:

Connessione MQTT

Crea connessioni con il broker MQTT per la pubblicazione e la sottoscrizione di argomenti cloud.

Coda e gestore dei messaggi

Elabora le richieste di aggiunta e rimozione dei dispositivi in arrivo in sequenza.

Interfaccia del plugin di protocollo

Funziona con plug-in di provisioning specifici del protocollo per l'onboarding dei dispositivi gestendo l'autenticazione e le modalità di collegamento radio.

Client Hub SDK APIs

Ricevi e inoltra report sulle funzionalità dei dispositivi dai plug-in CDMB specifici del protocollo alle integrazioni gestite.

Plugin di provisioning specifici per il protocollo

I plugin di provisioning specifici del protocollo sono librerie che gestiscono l'onboarding dei dispositivi per diversi protocolli di comunicazione. Ogni plugin traduce i comandi del core provisioner in azioni specifiche del protocollo per i tuoi dispositivi IoT. Questi plugin eseguono:

- Inizializzazione del middleware specifica per il protocollo
- Configurazione della modalità di collegamento radio basata sulle richieste del provider principale
- Rimozione del dispositivo tramite chiamate API middleware

Middleware specifico per il protocollo

Il middleware specifico del protocollo funge da livello di traduzione tra i protocolli del dispositivo e le integrazioni gestite. Questo componente elabora la comunicazione in entrambe le direzioni, ricevendo comandi dai plugin del provider e inviandoli agli stack di protocolli, raccogliendo anche le risposte dai dispositivi e indirizzandole nuovamente attraverso il sistema.

Flussi di onboarding dei dispositivi

Rivedi la sequenza di operazioni che si verificano quando esegui l'onboard dei dispositivi utilizzando Hub SDK. Questa sezione mostra come i componenti interagiscono durante il processo di onboarding e delinea i metodi di onboarding supportati.

Flussi di onboarding

- Configurazione semplice (SS)

- Configurazione Zero-touch (ZTS)
- Configurazione guidata dall'utente (UGS)

Configurazione semplice (SS)

L'utente finale accende il dispositivo IoT e ne scansiona il codice QR utilizzando l'applicazione del produttore del dispositivo. Il dispositivo viene quindi registrato nel cloud delle integrazioni gestite e si connette all'hub IoT.

Configurazione Zero-touch (ZTS)

La configurazione Zero-touch (ZTS) semplifica l'onboarding dei dispositivi preassociando il dispositivo a monte della catena di fornitura. Ad esempio, anziché essere gli utenti finali a scansionare il codice QR del dispositivo, questo passaggio viene completato prima per collegare in anticipo i dispositivi agli account dei clienti. Ad esempio, questo passaggio può essere completato presso il centro logistico.

Quando l'utente finale riceve e accende il dispositivo, si registra automaticamente nel cloud delle integrazioni gestite e si connette all'hub IoT senza richiedere ulteriori azioni di configurazione.

Configurazione guidata dall'utente (UGS)

L'utente finale accende il dispositivo e segue i passaggi interattivi per integrarlo nelle integrazioni gestite. Ciò potrebbe includere la pressione di un pulsante sull'hub IoT, l'utilizzo di un'app del produttore del dispositivo o la pressione di pulsanti sia sull'hub che sul dispositivo. È possibile utilizzare questo metodo se la configurazione semplice fallisce.

Controllo dei dispositivi

Le integrazioni gestite gestiscono la registrazione, l'esecuzione dei comandi e il controllo dei dispositivi. È possibile creare esperienze per gli utenti finali senza conoscere i protocolli specifici del dispositivo utilizzando una gestione dei dispositivi indipendente dal fornitore e dal protocollo.

Con il controllo dei dispositivi, è possibile visualizzare e modificare gli stati del dispositivo, come la luminosità della lampadina o la posizione della porta. La funzionalità emette eventi per i cambiamenti di stato, che puoi utilizzare per analisi, regole e monitoraggio.

Funzionalità principali

Modifica o leggi lo stato del dispositivo

Visualizza e modifica gli attributi del dispositivo in base ai tipi di dispositivo. Puoi accedere a:

- Stato del dispositivo: valori correnti degli attributi del dispositivo
- Stato di connettività: stato di raggiungibilità del dispositivo
- Health status: valori di sistema come il livello della batteria e la potenza del segnale (RSSI)

Notifica di modifica dello stato

Ricevi eventi quando gli attributi del dispositivo o lo stato di connettività cambiano, ad esempio le regolazioni della luminosità delle lampadine o le modifiche dello stato della serratura.

Modalità offline

I dispositivi comunicano con altri dispositivi sullo stesso hub IoT anche senza connettività Internet. Gli stati dei dispositivi si sincronizzano con il cloud quando viene ripristinata la connettività.

Sincronizzazione dello stato

Tieni traccia delle modifiche di stato provenienti da più fonti, dalle app dei produttori dei dispositivi e dalle regolazioni manuali dei dispositivi.

Esamina i componenti e i processi di Hub SDK necessari per controllare i dispositivi tramite integrazioni gestite. Questo argomento descrive come Edge Agent, Common Data Model Bridge (CDMB) e i plug-in specifici del protocollo interagiscono per gestire i comandi dei dispositivi, gestire gli stati dei dispositivi ed elaborare le risposte tra protocolli diversi.

Flussi di controllo dei dispositivi

Il diagramma seguente mostra il flusso di controllo del end-to-end dispositivo descrivendo come un utente finale accende una presa intelligente Zigbee.

Componenti Hub SDK per il controllo dei dispositivi

L'architettura Hub SDK utilizza i seguenti componenti per elaborare e instradare i comandi di controllo dei dispositivi nell'implementazione IoT. Ogni componente svolge un ruolo specifico nella traduzione

dei comandi cloud in azioni del dispositivo, nella gestione degli stati del dispositivo e nella gestione delle risposte. Le seguenti sezioni descrivono in dettaglio come questi componenti interagiscono nella distribuzione:

L'Hub SDK è composto dai seguenti componenti e facilita l'onboarding e il controllo dei dispositivi sugli hub IoT.

Componenti principali:

Agente Edge

Funge da gateway tra l'hub IoT e le integrazioni gestite.

Common Data Model Bridge (CDMB)

Traduce tra il modello di AWS dati e i modelli di dati del protocollo locale come Z-Wave e Zigbee. Include un CDMB di base e plugin CDMB specifici del protocollo.

Fornitore

Gestisce l'individuazione e l'onboarding dei dispositivi. Include un core provisioner e plug-in provisioner specifici del protocollo per attività di onboarding specifiche del protocollo.

Componenti secondari

Integrazione all'hub

Fornisce all'hub certificati e chiavi client per comunicazioni sicure sul cloud.

Proxy MQTT

Fornisce connessioni MQTT al cloud delle integrazioni gestite.

Logger

Scrive i log localmente o nel cloud delle integrazioni gestite.

Installa e convalida le integrazioni gestite Hub SDK

Scegli tra i seguenti metodi di distribuzione per installare le integrazioni gestite Hub SDK sui tuoi dispositivi,AWS IoT Greengrass per la distribuzione automatizzata o l'installazione manuale degli script. Questa sezione descrive le fasi di configurazione e convalida per entrambi gli approcci.

Metodi di distribuzione

- Installa l'Hub SDK con AWS IoT Greengrass
- Implementa l'Hub SDK con uno script
- Implementa Hub SDK con systemd

Installa l'Hub SDK con AWS IoT Greengrass

Implementa le integrazioni gestite che utilizzano i componenti Hub SDK per i tuoi dispositivi AWS IoT Greengrass (versione Java).

> **Note**
>
> È necessario averlo già configurato e conoscerlo. AWS IoT Greengrass Per ulteriori informazioni, consulta Cosa contiene la AWS IoT Greengrass documentazione della guida per gli AWS IoT Greengrass sviluppatori.

L' AWS IoT Greengrass utente deve disporre dell'autorizzazione per modificare le seguenti directory:

- `/dev/aipc`
- `/data/aws/iotmi/config`
- `/data/ace/kvstorage`

Argomenti

- Distribuisci i componenti localmente
- Distribuzione nel cloud
- Verifica il provisioning dell'hub
- Verifica il funzionamento di CDMB
- Verificare il funzionamento di LPW-Provisioner

Distribuisci i componenti localmente

Utilizza l'CreateDeployment AWS IoT Greengrass API sul tuo dispositivo per distribuire i componenti Hub SDK. I numeri di versione non sono statici e possono variare in base alla versione

utilizzata in quel momento. Utilizza il seguente formato per**`version`**: com.amazon.io. TManaged IntegrationsDevice AceCommon=. `0.2.0`

```
/greengrass/v2/bin/greengrass-cli deployment create \
--recipeDir recipes \
--artifactDir artifacts \
-m "com.amazon.IoTManagedIntegrationsDevice.AceCommon=version" \
-m "com.amazon.IoTManagedIntegrationsDevice.HubOnboarding=version" \
-m "com.amazon.IoTManagedIntegrationsDevice.AceZigbee=version" \
-m "com.amazon.IoTManagedIntegrationsDevice.LPW-Provisioner=version" \
-m "com.amazon.IoTManagedIntegrationsDevice.Agent=version" \
-m "com.amazon.IoTManagedIntegrationsDevice.MQTTProxy=version" \
-m "com.amazon.IoTManagedIntegrationsDevice.CDMB=version" \
-m "com.amazon.IoTManagedIntegrationsDevice.AceZwave=version"
```

Distribuzione nel cloud

Segui le istruzioni contenute nella guida per AWS IoT Greengrass sviluppatori per eseguire i seguenti passaggi:

1. Carica artefatti su Amazon S3.
2. Aggiorna le ricette per includere la posizione degli artefatti di Amazon S3.
3. Crea una distribuzione cloud sul dispositivo per i nuovi componenti.

Verifica il provisioning dell'hub

Conferma la corretta esecuzione del provisioning controllando il file di configurazione. Apri il `/data/aws/iotmi/config/iotmi_config.json` file e verifica che lo stato sia impostato su. `PROVISIONED`

Verifica il funzionamento di CDMB

Controlla il file di registro per i messaggi di avvio di CDMB e l'inizializzazione corretta. La *`logs file`* posizione può variare a seconda di dove AWS IoT Greengrass è installato.

```
tail -f -n 100 /greengrass/v2/logs/com.amazon.IoTManagedIntegrationsDevice.CDMB.log
```

Esempio

```
[2024-09-06 02:31:54.413758906][IoTManagedIntegrationsDevice_CDMB][info] Successfully
 subscribed to topic: south/bF|gi_044F8821D0193608C8D5BF80858E20A56E3A8490/control
[2024-09-06 02:31:54.513956059][IoTManagedIntegrationsDevice_CDMB][info] Successfully
 subscribed to topic: south/bF|gi_044F8821D0193608C8D5BF80858E20A56E3A8490/setup
```

Verificare il funzionamento di LPW-Provisioner

Controllate il file di registro per i messaggi di avvio di LPW-Provisioner e la corretta inizializzazione. La *`logs file`* posizione può variare a seconda di dove è installato. AWS IoT Greengrass

```
tail -f -n 100 /greengrass/v2/logs/com.amazon.IoTManagedIntegrationsDevice.LPW-
Provisioner.log
```

Esempio

```
[2024-09-06 02:33:22.068898877][LPWProvisionerCore][info] Successfully subscribed to
 topic: south/bF|gi_044F8821D0193608C8D5BF80858E20A56E3A8490/setup
```

Implementa l'Hub SDK con uno script

Distribuisci manualmente i componenti Hub SDK delle integrazioni gestite utilizzando gli script di installazione, quindi convalida la distribuzione. Questa sezione descrive le fasi di esecuzione degli script e il processo di verifica.

Argomenti

Prepara il tuo ambiente

Completa questi passaggi prima di eseguire lo script di installazione dell'SDK:

1. Crea una cartella denominata `middleware` all'interno della `artifacts` cartella.
2. Copia i file del middleware dell'hub nella `middleware` cartella.

3. Esegui i comandi di inizializzazione prima di avviare l'SDK.

> **Important**
>
> Ripeti i comandi di inizializzazione dopo ogni riavvio dell'hub.

```
#Get the current user
_user=$(whoami)

#Get the current group
_grp=$(id -gn)

#Display the user and group
echo "Current User: $_user"
echo "Current Group: $_grp"

sudo mkdir -p /dev/aipc/
sudo chown -R $_user:$_grp /dev/aipc
sudo mkdir -p /data/ace/kvstorage
sudo chown -R $_user:$_grp /data/ace/kvstorage
```

Esegui lo script Hub SDK

Vai alla directory degli artefatti ed esegui lo script. `start_iotmi_sdk.sh` Questo script avvia i componenti dell'hub SDK nella sequenza corretta. Esamina i seguenti log di esempio per verificare il successo dell'avvio:

> **Note**
>
> I registri di tutti i componenti in esecuzione si trovano all'interno della `artifacts/logs` cartella.

```
hub@hub-293ea release_Oct_17$ ./start_iotmi_sdk.sh
-------Stopping SDK running processes---
DeviceAgent: no process found
------Starting SDK-------
-------Creating logs directory----------
```

```
Logs directory created.
-------Verifying Middleware paths-------
All middleware libraries exist
-------Verifying Middleware pre reqs---
AIPC and KVstroage directories exist
-------Starting HubOnboarding-----------
-------Starting MQTT Proxy-----------
-------Starting Event Manager-----------
-------Starting Zigbee Service----------
-------Starting Zwave Service----------
/data/release_Oct_17/middleware/AceZwave/bin /data/release_Oct_17
/data/release_Oct_17
-------Starting CDMB--------------------
-------Starting Agent--------------------
-------Starting Provisioner--------------
-------Checking SDK status---------------
hub         6199  1.7  0.7 1004952 15568 pts/2   Sl+  21:41   0:00 ./iotmi_mqtt_proxy -
C /data/aws/iotmi/config/iotmi_config.json
Process 'iotmi_mqtt_proxy' is running.
hub         6225  0.0  0.1 301576  2056 pts/2    Sl+  21:41   0:00 ./middleware/
AceCommon/bin/ace_eventmgr
Process 'ace_eventmgr' is running.
hub         6234  104  0.2 238560  5036 pts/2    Sl+  21:41   0:38 ./middleware/
AceZigbee/bin/ace_zigbee_service
Process 'ace_zigbee_service' is running.
hub         6242  0.4  0.7 1569372 14236 pts/2   Sl+  21:41   0:00 ./zwave_svc
Process 'zwave_svc' is running.
hub         6275  0.0  0.2 1212744 5380 pts/2    Sl+  21:41   0:00 ./DeviceCdmb
Process 'DeviceCdmb' is running.
hub         6308  0.6  0.9 1076108 18204 pts/2   Sl+  21:41   0:00 ./
IoTManagedIntegrationsDeviceAgent
Process 'DeviceAgent' is running.
hub         6343  0.7  0.7 1388132 13812 pts/2   Sl+  21:42   0:00 ./
iotmi_lpw_provisioner
Process 'iotmi_lpw_provisioner' is running.
------Successfully Started SDK----
```

Verifica il provisioning dell'hub

Verifica che il `iot_provisioning_state` campo in `/data/aws/iotmi/config/iotmi_config.json` sia impostato su`PROVISIONED`.

Verifica il funzionamento dell'agente

Controlla il file di registro per i messaggi di avvio dell'agente e la corretta inizializzazione.

```
tail -f -n 100 logs/agent_logs.txt
```

Esempio

```
[2024-09-06 02:31:54.413758906][Device_Agent][info] Successfully subscribed to topic:
 south/bF|gi_044F8821D0193608C8D5BF80858E20A56E3A8490/control
[2024-09-06 02:31:54.513956059][Device_Agent][info] Successfully subscribed to topic:
 south/bF|gi_044F8821D0193608C8D5BF80858E20A56E3A8490/setup
```

> **Note**
>
> Verifica che il `iotmi.db` database esista nella tua `artifacts` directory.

Verifica il funzionamento di LPW-Provisioner

Controllate il file di registro per i messaggi di `LPW-Provisioner` avvio e l'inizializzazione corretta.

```
tail -f -n 100 logs/provisioner_logs.txt
```

Il codice seguente mostra un esempio.

```
[2024-09-06 02:33:22.068898877][LPWProvisionerCore][info] Successfully subscribed to
 topic: south/bF|gi_044F8821D0193608C8D5BF80858E20A56E3A8490/setup
```

Implementa Hub SDK con systemd

> **Important**
>
> Segui la `hubSystemdSetup` cartella del `readme.md` file release.tgz per gli ultimi aggiornamenti.

Questa sezione descrive gli script e i processi per la distribuzione e la configurazione dei servizi su un dispositivo hub basato su Linux.

Panoramica

Il processo di distribuzione è costituito da due script principali:

- `copy_to_hub.sh`: viene eseguito sul computer host per copiare i file necessari nell'hub
- `setup_hub.sh`: viene eseguito sull'hub per configurare l'ambiente e distribuire i servizi

Inoltre, `systemd/deploy_iotshd_services_on_hub.sh` gestisce la sequenza di avvio dei processi e la gestione delle autorizzazioni dei processi e viene attivato automaticamente da. `setup_hub.sh`

Prerequisiti

I prerequisiti elencati sono necessari per una corretta implementazione.

- il servizio systemd è disponibile sull'hub
- accesso SSH al dispositivo hub
- Privilegi Sudo sul dispositivo hub
- `scp`utilità installata sul computer host
- `sed`utilità installata sul computer host
- utilità di decompressione installata sul computer host

Struttura dei file

La struttura del file è progettata per facilitare l'organizzazione e la gestione dei suoi vari componenti, consentendo un accesso e una navigazione efficienti del contenuto.

```
hubSystemdSetup/
### README.md
### copy_to_hub.sh
### setup_hub.sh
### iotshd_config.json  # Sample configuration file
### local_certs/  # Directory for DHA certificates
### systemd/
    ### *.service.template  # Systemd service templates
    ### deploy_iotshd_services_on_hub.sh
```

Nel file tgz della versione SDK, la struttura complessiva del file è:

```
IoT-managed-integrations-Hub-SDK-aarch64-v1.0.0.tgz
###package/
    ###greengrass/
        ###artifacts/
        ###recipes/
    ###hubSystemdSetup/
        ### REAME.md
        ### copy_to_hub.sh
        ### setup_hub.sh
        ### iotshd_config.json  # Sample configuration file
        ### local_certs/  # Directory for DHA certificates
        ### systemd/
            ### *.service.template  # Systemd service templates
            ### deploy_iotshd_services_on_hub.sh
```

Configurazione iniziale

Estrai il pacchetto SDK

```
tar -xzf managed-integrations-Hub-SDK-v<Version>-linux-aarch64-<timestamp>.tgz
```

Vai alla directory estratta e prepara il pacchetto:

```
# Create package.zip containing required artifacts
zip -r package.zip package/greengrass/artifacts
# Move package.zip to the hubSystemdSetup directory
mv package.zip ../hubSystemdSetup/
```

Aggiungi i file di configurazione del dispositivo

Segui i due passaggi elencati per creare i file di configurazione del dispositivo e copiarli nell'hub.

1. Aggiungi i file di configurazione del dispositivo per creare i file di configurazione del dispositivo necessari. L'SDK utilizza questo file per le sue funzioni.
2. Copia i file di configurazione per copiare i file di configurazione creati nell'hub.

Copiare i file nell'hub

Esegui lo script di distribuzione dal tuo computer host:

```
chmod +x copy_to_hub.sh
./copy_to_hub.sh hub_ip_address package_file
```

Example Esempio

```
./copy_to_hub.sh 192.168.50.223 ~/Downloads/EAR3-package.zip
```

Questo copia:

- Il file del pacchetto (rinominato in package.zip sull'hub)
- File di configurazione
- Certificati
- file di servizio Systemd

Configura l'hub

Dopo aver copiato i file, inserisci SSH nell'hub ed esegui lo script di configurazione:

```
ssh hub@hub_ip
chmod +x setup_hub.sh
sudo ./setup_hub.sh
```

Configurazioni di utenti e gruppi

Per impostazione predefinita, utilizziamo l'hub utente e l'hub di gruppo per i componenti SDK. Esistono diversi modi per configurarli:

- Usa un utente/gruppo personalizzato:

  ```
  sudo ./setup_hub.sh --user=USERNAME --group=GROUPNAME
  ```

- Creali manualmente prima di eseguire lo script di installazione:

  ```
  sudo groupadd -f GROUPNAME
  sudo useradd -r -g GROUPNAME USERNAME
  ```

- Aggiungi i comandi in`setup_hub.sh`.

Gestisci i servizi

Per riavviare tutti i servizi, esegui il seguente script dall'hub:

```
sudo /usr/local/bin/deploy_iotshd_services_on_hub.sh
```

Lo script di installazione creerà le directory necessarie, imposterà le autorizzazioni appropriate e distribuirà automaticamente i servizi. Se non utilizzi SSH/SCP, devi modificarlo in base al tuo metodo di distribuzione specifico. `copy_to_hub.sh` Assicurati che tutti i file e le configurazioni dei certificati siano configurati correttamente prima della distribuzione.

Integra i tuoi hub verso integrazioni gestite

Configura i dispositivi hub per comunicare con le integrazioni gestite configurando la struttura di directory, i certificati e i file di configurazione dei dispositivi richiesti. Questa sezione descrive come interagiscono i componenti del sottosistema di onboarding dell'hub, dove archiviare certificati e file di configurazione, come creare e modificare il file di configurazione del dispositivo e i passaggi per completare il processo di provisioning dell'hub.

Sottosistema Hub Onboarding

Il sottosistema di onboarding dell'hub utilizza questi componenti principali per gestire il provisioning e la configurazione dei dispositivi:

Componente di onboarding dell'hub

Gestisce il processo di onboarding dell'hub coordinando lo stato dell'hub, l'approccio di approvvigionamento e i materiali di autenticazione.

File di configurazione del dispositivo

Memorizza i dati essenziali di configurazione dell'hub sul dispositivo, tra cui:

- Stato di fornitura del dispositivo (fornito o non fornito)
- Certificato e posizioni chiave
- Informazioni di autenticazione Altri processi SDK, come il proxy MQTT, fanno riferimento a questo file per determinare lo stato dell'hub e le impostazioni di connessione.

Interfaccia per il gestore dei certificati

Fornisce un'interfaccia di utilità per la lettura e la scrittura di certificati e chiavi dei dispositivi. È possibile implementare questa interfaccia per lavorare con:

- Archiviazione del file system
- Moduli di sicurezza hardware (HSM)
- Moduli Trusted Platform (TPM)
- Soluzioni di archiviazione sicure personalizzate

Componente proxy MQTT

Gestisce device-to-cloud la comunicazione utilizzando:

- Certificati e chiavi client forniti
- Informazioni sullo stato del dispositivo contenute nel file di configurazione
- Connessioni MQTT alle integrazioni gestite

Il diagramma seguente descrive l'architettura del sottosistema Hub Onboarding e i relativi componenti. Se non lo stai utilizzando AWS IoT Greengrass, puoi ignorare quel componente del diagramma.

Configurazione dell'hub onboarding

Completa questi passaggi di configurazione per ogni dispositivo hub prima di iniziare il processo di onboarding per il provisioning della flotta. Questa sezione descrive come creare oggetti gestiti, impostare strutture di directory e configurare i certificati richiesti.

Passaggi di impostazione

- Fase 1: Registrare un endpoint personalizzato
- Fase 2: Creare un profilo di provisioning
- Fase 3: Creare un oggetto gestito (approvvigionamento della flotta)
- Passaggio 4: Creare la struttura delle cartelle
- Fase 5: Aggiungere materiale di autenticazione al dispositivo hub
- Passaggio 6: Creare il file di configurazione del dispositivo
- Fase 7: Copiare il file di configurazione nell'hub

Fase 1: Registrare un endpoint personalizzato

Crea un endpoint di comunicazione dedicato che i tuoi dispositivi utilizzino per lo scambio di dati con integrazioni gestite. Questo endpoint stabilisce un punto di connessione sicuro per tutti i device-to-cloud messaggi, inclusi i comandi del dispositivo, gli aggiornamenti di stato e le notifiche.

Per registrare un endpoint

- Utilizza l'RegisterCustomEndpointAPI per creare un endpoint per la comunicazione delle device-to-managed integrazioni.

  RegisterCustomEndpointEsempio di richiesta

  ```
  aws iot-managed-integrations register-custom-endpoint
  ```

  Risposta:

  ```
  {
    [ACCOUNT-PREFIX]-ats.iot.AWS-REGION.amazonaws.com
  }
  ```

  > **Note**
  >
  > Memorizza l'indirizzo dell'endpoint. Ti servirà per le future comunicazioni tra dispositivi.

  Per restituire le informazioni sull'endpoint, utilizza l'`GetCustomEndpoint`API.

  Per ulteriori informazioni, consulta l'RegisterCustomEndpointAPI e l'API nella Guida di riferimento dell'GetCustomEndpointAPI per le integrazioni gestite.

Fase 2: Creare un profilo di provisioning

Un profilo di provisioning contiene le credenziali di sicurezza e le impostazioni di configurazione necessarie ai dispositivi per connettersi alle integrazioni gestite.

Per creare un profilo di approvvigionamento della flotta

- Chiama l'CreateProvisioningProfileAPI per generare quanto segue:

- Un modello di provisioning che definisce le impostazioni di connessione del dispositivo
- Un certificato di richiesta e una chiave privata per l'autenticazione del dispositivo

> **Important**
>
> Archivia il certificato di richiesta, la chiave privata e l'ID del modello in modo sicuro. Avrai bisogno di queste credenziali per integrare i dispositivi nelle integrazioni gestite. Se perdi queste credenziali, devi creare un nuovo profilo di provisioning.

`CreateProvisioningProfile`richiesta di esempio

```
aws iot-managed-integrations create-provisioning-profile \
    --provisioning-type FLEET_PROVISIONING \
    --name PROFILE_NAME
```

Risposta:

```
{
"Arn":"arn:aws:iotmanagedintegrations:AWS-REGION:ACCOUNT-ID:provisioning-
profile/PROFILE-ID",
    "ClaimCertificate":
  "-----BEGIN CERTIFICATE-----
  MIICiTCCAfICCQD6m7.....w3rrszlaEXAMPLE=
  -----END CERTIFICATE-----",
    "ClaimCertificatePrivateKey":
  "-----BEGIN RSA PRIVATE KEY-----
  MIICiTCCAfICCQ...3rrszlaEXAMPLE=
 -----END RSA PRIVATE KEY-----",
    "Id": "PROFILE-ID",
    "PROFILE-NAME",
         "ProvisioningType": "FLEET_PROVISIONING"
}
```

Fase 3: Creare un oggetto gestito (approvvigionamento della flotta)

Utilizza l'`CreateManagedThing`API per creare un oggetto gestito per il tuo dispositivo hub. Ogni hub richiede una propria funzionalità gestita con materiali di autenticazione unici. Per ulteriori

informazioni, consulta l'CreateManagedThingAPI nella Guida alle API di riferimento per le integrazioni gestite.

Quando crei un oggetto gestito, specifica questi parametri:

- `Role`: Imposta questo valore su `CONTROLLER` per gli hub che non supportano il comando e il controllo, altrimenti imposta su. `DEVICE`
- `AuthenticationMaterialType`: Imposta questo valore su. `WIFI_SETUP_QR_BAR_CODE`
- `AuthenticationMaterial`: include i seguenti campi. È possibile utilizzare uno dei due `UPC` o `EAN` non entrambi.
  - `SN`: Il numero di serie univoco di questo dispositivo
  - `UPC`: Il codice prodotto universale per questo dispositivo
  - `EAN`: Il numero di articolo internazionale di questo dispositivo

> **Important**
>
> Ogni dispositivo deve avere un numero di serie (SN) univoco nel materiale di autenticazione.

`CreateManagedThing`Esempio di richiesta:

```
{
 "Role": "CONTROLLER",
 "AuthenticationMaterialType": "WIFI_SETUP_QR_BAR_CODE",
 "AuthenticationMaterial": "SN:123456789524;UPC:829576019524"
}
```

Per ulteriori informazioni, consulta il riferimento CreateManagedThingall'API di riferimento per le integrazioni gestite.

(Facoltativo) Ottieni una cosa gestita

La `ProvisioningStatus` cosa che hai gestito deve essere quella che hai `PRE_ASSOCIATED` prima di poter procedere. Per ulteriori informazioni su ProvisioningStatus, consulta Device Provisioning. Utilizza l'`GetManagedThing`API per verificare che l'oggetto gestito esista e sia pronto per il provisioning. Per ulteriori informazioni, consulta il riferimento GetManagedThingall'API di riferimento per le integrazioni gestite.

Passaggio 4: Creare la struttura delle cartelle

Crea le directory per i tuoi file di configurazione e i tuoi certificati. Per impostazione predefinita, il processo di onboarding dell'hub utilizza il. `/data/aws/iotmi/config/iotmi_config.json`

È possibile specificare percorsi personalizzati per certificati e chiavi private nel file di configurazione. Questa guida utilizza il percorso predefinito`/data/aws/iotmi/certs`.

```
mkdir -p /data/aws/iotmi/config
mkdir -p /data/aws/iotmi/certs

/data/
    aws/
        iotmi/
            config/
            certs/
```

Fase 5: Aggiungere materiale di autenticazione al dispositivo hub

Copia i certificati e le chiavi sul tuo dispositivo hub, quindi crea un file di configurazione specifico del dispositivo. Questi file stabiliscono una comunicazione sicura tra l'hub e le integrazioni gestite durante il processo di provisioning.

Per copiare il certificato e la chiave della richiesta

- Copia questi file di autenticazione dalla tua risposta `CreateProvisioningProfile` API al tuo dispositivo hub:
  - `claim_cert.pem`: Il certificato di richiesta (comune a tutti i dispositivi)
  - `claim_pk.key`: La chiave privata per il certificato di richiesta

  Posiziona entrambi i file nella `/data/aws/iotmi/certs` directory.

  > **Important**
  >
  > Quando memorizzi certificati e chiavi private in formato PEM, assicurati che la formattazione sia corretta gestendo correttamente i caratteri di nuova riga. Per i file con codifica PEM, i caratteri di nuova riga (`\n`) devono essere sostituiti con veri e propri

separatori di riga, poiché la semplice memorizzazione delle nuove righe in escape non verrà recuperata correttamente in seguito.

> **Note**
>
> Se utilizzi l'archiviazione sicura, archivia queste credenziali nella tua posizione di archiviazione sicura anziché nel file system. Per ulteriori informazioni, consulta Crea un gestore di certificati personalizzato per l'archiviazione sicura.

Passaggio 6: Creare il file di configurazione del dispositivo

Crea un file di configurazione che contenga identificatori univoci del dispositivo, posizioni dei certificati e impostazioni di provisioning. L'SDK utilizza questo file durante l'onboarding dell'hub per autenticare il dispositivo, gestire lo stato del provisioning e memorizzare le impostazioni di connessione.

> **Note**
>
> Ogni dispositivo hub richiede il proprio file di configurazione con valori unici specifici del dispositivo.

Utilizzare la procedura seguente per creare o modificare il file di configurazione e copiarlo nell'hub.

- Creare o modificare il file di configurazione (fleet provisioning).

  Configura questi campi obbligatori nel file di configurazione del dispositivo:

  - Percorsi dei certificati
    1. `iot_claim_cert_path`: Ubicazione del certificato di reclamo (`claim_cert.pem`)
    2. `iot_claim_pk_path`: Ubicazione della chiave privata (`claim_pk.key`)
    3. `SECURE_STORAGE`Utilizzatelo per entrambi i campi quando implementate il Secure Storage Cert Handler
  - Impostazioni di connessione
    1. `fp_template_name`: Il `ProvisioningProfile` nome usato in precedenza.

2. `endpoint_url`: l'URL dell'endpoint delle integrazioni gestite ricavato dalla risposta dell'`RegisterCustomEndpoint`API (uguale per tutti i dispositivi in una regione).

- Identificatori di dispositivo
  1. SN: numero di serie del dispositivo che corrisponde alla chiamata `CreateManagedThing` API (unico per dispositivo)
  2. UPCCodice prodotto universale ottenuto dalla chiamata `CreateManagedThing` API (uguale per tutti i dispositivi di questo prodotto)

```
{
    "ro": {
        "iot_provisioning_method": "FLEET_PROVISIONING",
        "iot_claim_cert_path": "<SPECIFY_THIS_FIELD>",
        "iot_claim_pk_path": "<SPECIFY_THIS_FIELD>",
        "fp_template_name": "<SPECIFY_THIS_FIELD>",
        "endpoint_url": "<SPECIFY_THIS_FIELD>",
        "SN": "<SPECIFY_THIS_FIELD>",
        "UPC": "<SPECIFY_THIS_FIELD>"
    },
    "rw": {
        "iot_provisioning_state": "NOT_PROVISIONED"
    }
}
```

Contenuto del file di configurazione

Esamina il contenuto del `iotmi_config.json` file.

Indice

Chiave	Valori	Aggiunto dal cliente?	Note
`iot_provisioning_method`	`FLEET_PROVISIONING`	Sì	Specificate il metodo di provisioning che desiderate utilizzare.

Chiave	Valori	Aggiunto dal cliente?	Note
`iot_claim_cert_path`	Il percorso del file specificato o`SECURE_STORAGE` . Ad esempio, `/data/aws/iotmi/certs/claim_cert.pem`	Sì	Specificate il percorso del file che desiderate utilizzare o`SECURE_STORAGE` .
`iot_claim_pk_path`	Il percorso del file specificato o`SECURE_STORAGE` . Ad esempio, `/data/aws/iotmi/certs/claim_pk.pem`	Sì	Specificate il percorso del file che desiderate utilizzare o`SECURE_STORAGE` .
`fp_template_name`	Il nome del modello di fleet provisioning deve essere uguale al nome del `ProvisioningProfile` modello utilizzato in precedenza.	Sì	Uguale al nome usato `ProvisioningProfile` in precedenza
`endpoint_url`	L'URL dell'endpoint per le integrazioni gestite.	Sì	I tuoi dispositivi utilizzano questo URL per connetter si al cloud delle integrazioni gestite. Per ottenere queste informazioni, utilizza l'RegisterCustomEndpointAPI.
`SN`	Il numero di serie del dispositivo. Ad esempio, `AIDACKCEVSQ6C2EXAMPLE` .	Sì	È necessario fornire queste informazioni univoche per ogni dispositivo.

Chiave	Valori	Aggiunto dal cliente?	Note
UPC	Codice prodotto universale del dispositivo. Ad esempio, 841667145075 .	Sì	È necessario fornire queste informazioni per il dispositivo.
managed_thing_id	L'ID dell'oggetto gestito.	No	Queste informazioni vengono aggiunte successivamente durante il processo di onboarding dopo il provisioning dell'hub.
iot_provisioning_state	Lo stato di approvvigionamento.	Sì	Lo stato di approvvigionamento deve essere impostato come. NOT_PROVISIONED
iot_permanent_cert_path	Il percorso del certificato IoT. Ad esempio, /data/aws/iotmi/iot_cert.pem .	No	Queste informazioni vengono aggiunte successivamente durante il processo di onboarding dopo il provisioning dell'hub.
iot_permanent_pk_path	Il percorso del file della chiave privata IoT. Ad esempio, /data/aws/iotmi/iot_pk.pem .	No	Queste informazioni vengono aggiunte successivamente durante il processo di onboarding dopo il provisioning dell'hub.
client_id	L'ID client che verrà utilizzato per le connessioni MQTT.	No	Queste informazioni vengono aggiunte successivamente durante il processo di onboarding dopo il provisioning dell'hub, per consentire l'utilizzo di altri componenti.
mqtt_keep_alive_interval	L'intervallo è compreso tra 30 e 1200 e le unità sono espresse in secondi. Il valore predefinito è 300.	Sì	Utilizzatelo per impostare un intervallo di mantenimento attivo per le connessioni MQTT.

Chiave	Valori	Aggiunto dal cliente?	Note
`event_manager_upper_bound`	Il valore predefinito è 500.	No	Queste informazioni vengono aggiunte successivamente durante il processo di onboarding dopo il provisioning dell'hub, per essere utilizzate da altri componenti.

Fase 7: Copiare il file di configurazione nell'hub

Copia il file di configurazione `/data/aws/iotmi/config` o il percorso di directory personalizzato. Fornirai questo percorso al `HubOnboarding` file binario durante il processo di onboarding.

Per l'approvvigionamento della flotta

```
/data/
    aws/
        iotmi/
            config/
                iotmi_config.json
            certs/
                claim_cert.pem
                claim_pk.key
```

Incorpora i dispositivi e gestiscili nell'hub

Configura i tuoi dispositivi per l'onboarding nel tuo hub di integrazioni gestite creando un oggetto gestito e collegandolo al tuo hub. I dispositivi possono essere integrati in un hub tramite una configurazione semplice o una configurazione guidata dall'utente.

Argomenti

- Configurazione semplice per l'installazione e il funzionamento dei dispositivi
- Configurazione guidata dall'utente per l'installazione e il funzionamento dei dispositivi

Configurazione semplice per l'installazione e il funzionamento dei dispositivi

Configura i tuoi dispositivi per l'onboarding nel tuo hub di integrazioni gestite creando un oggetto gestito e collegandolo al tuo hub. Questa sezione descrive i passaggi per completare il processo di onboarding del dispositivo utilizzando una configurazione semplice.

Prerequisiti

Completa questi passaggi prima di tentare di effettuare l'onboard di un dispositivo:

- Effettua l'onboarding di un dispositivo hub nell'hub di integrazioni gestite.
- Installa la versione più recente di AWS CLI dal Managed AWS CLI Integrations Command Reference
- Iscriviti alle notifiche degli eventi DEVICE_LIFE_CYCLE.

Passaggi di impostazione

- Passaggio 1: creare un armadietto per le credenziali
- Passaggio 2: aggiungi l'armadietto delle credenziali al tuo hub
- Fase 3: Creare un oggetto gestito con credenziali.
- Passaggio 4: Collega il dispositivo e verificane lo stato.
- Passaggio 5: Ottieni le funzionalità del dispositivo
- Passaggio 6: inviare un comando all'oggetto gestito
- Passaggio 7: rimuovi l'elemento gestito dal tuo hub

Passaggio 1: creare un armadietto per le credenziali

Crea un blocco delle credenziali per il tuo dispositivo.

Per creare un armadietto per le credenziali

- Utilizza il comando `create-credential-locker`. L'esecuzione di questo comando attiverà la creazione di tutte le risorse di produzione, tra cui la key pair di configurazione Wi-Fi e il certificato del dispositivo.

  Esempio create-credential-locker

```
aws iot-managed-integrations create-credential-locker \
  --name "DEVICE_NAME"
```

Risposta:

```
{
  "Id": "LOCKER_ID"
  "Arn": "arn:aws:iotmanagedintegrations:AWS_REGION:AWS_ACCOUNT_ID:credential-
locker/LOCKER_ID
  "CreatedAt": "2025-06-09T13:58:52.977000+08:00"
}
```

Per ulteriori informazioni, consulta il create-credential-lockercomando nel Command Reference delle integrazioni AWS CLI gestite.

Passaggio 2: aggiungi l'armadietto delle credenziali al tuo hub

Aggiungi l'armadietto delle credenziali al tuo hub.

Per aggiungere un armadietto per le credenziali al tuo hub

- Usa il seguente comando per aggiungere un armadietto per le credenziali al tuo hub.

```
aws iotmi --region AWS_REGION --endpoint AWS_ENDPOINT update-managed-thing \
--identifier "HUB_MANAGED_THING_ID" --credential-locker-id "LOCKER_ID"
```

Fase 3: Creare un oggetto gestito con credenziali.

Crea un oggetto gestito con credenziali per il tuo dispositivo. Ogni dispositivo richiede la propria funzionalità gestita.

Per creare un oggetto gestito

- Usa il `create-managed-thing` comando per creare un oggetto gestito per il tuo dispositivo.

  Esempio create-managed-thing

```
#ZWAVE:
```

```
aws iot-managed-integrations create-managed-thing --role DEVICE \
--authentication-material '900137947003133...' \ #auth material from zwave qr code
--authentication-material-type ZWAVE_QR_BAR_CODE \
--credential-locker-id ${locker_id}

#ZIGBEE:
aws iot-managed-integrations create-managed-thing --role DEVICE \
--authentication-material 'Z:286...$I:A4DC00.' \ #auth material from zigbee qr code
--authentication-material-type ZIGBEE_QR_BAR_CODE \
--credential-locker-id ${locker_id}
```

> **Note**
>
> Esistono comandi separati per i dispositivi Z-wave e Zigbee.

Risposta:

```
{
  "Id": "DEVICE_MANAGED_THING_ID"
  "Arn": "arn:aws:iotmanagedintegrations:AWS_REGION:AWS_ACCOUNT_ID:managed-
thing/DEVICE_MANAGED_THING_ID"
  "CreatedAt": "2025-06-09T13:58:52.977000+08:00"
}
```

Per ulteriori informazioni, consulta il create-managed-thingcomando nel Command Reference delle integrazioni AWS CLI gestite.

Passaggio 4: Collega il dispositivo e verificane lo stato.

Collega il dispositivo e controllane lo stato.

- Usa il `get-managed-thing` comando per controllare lo stato del tuo dispositivo. La ProvisioningStatus parte della cosa gestita deve essere ATTIVATA. Per ulteriori informazioni su ProvisioningStatus, vedere Device Provisioning.

  Esempio get-managed-thing

```
#KINESIS NOTIFICATION:
{
```

```
    "version": "1.0.0",
    "messageId": "4ac684bb7f4c41adbb2eecc1e7991xxx",
    "messageType": "DEVICE_LIFE_CYCLE",
    "source": "aws.iotmanagedintegrations",
    "customerAccountId": "12345678901",
    "timestamp": "2025-06-10T05:30:59.852659650Z",
    "region": "us-east-1",
    "resources": ["XXX"],
    "payload": {
        "deviceDetails": {
            "id": "1e84f61fa79a41219534b6fd57052XXX",
            "arn": "XXX",
            "createdAt": "2025-06-09T06:24:34.336120179Z",
            "updatedAt": "2025-06-10T05:30:59.784157019Z"
        },
        "status": "ACTIVATED"
    }
}
aws iot-managed-integrations get-managed-thing \
--identifier :"DEVICE_MANAGED_THING_ID"
```

Risposta:

```
{
  "Id": "DEVICE_MANAGED_THING_ID"
  "Arn": "arn:aws:iotmanagedintegrations:AWS_REGION:AWS_ACCOUNT_ID:managed-
thing/MANAGED_THING_ID"
  "CreatedAt": "2025-06-09T13:58:52.977000+08:00"
}
```

Per ulteriori informazioni, consulta il get-managed-thingcomando nella Guida ai comandi delle integrazioni AWS CLI gestite.

Passaggio 5: Ottieni le funzionalità del dispositivo

Utilizza il `get-managed-thing-capabilities` comando per ottenere l'ID dell'endpoint e visualizzare un elenco di azioni possibili per il tuo dispositivo.

Per ottenere le funzionalità di un dispositivo

- Usa il `get-managed-thing-capabilities` comando e annota l'ID dell'endpoint.

Esempio get-managed-thing-capabilties

```
aws iotmi get-managed-thing-capabilities \
--identifier "DEVICE_MANAGED_THING_ID"
```

Risposta:

```
{
    "ManagedThingId": "1e84f61fa79a41219534b6fd57052cbc",
    "CapabilityReport": {
        "version": "1.0.0",
        "nodeId": "zw.FCB10009+06",
        "endpoints": [
            {
                "id": "ENDPOINT_ID"
                "deviceTypes": [
                    "On/Off Switch"
                ],
                "capabilities": [
                    {
                        "id": "matter.OnOff@1.4",
                        "name": "On/Off",
                        "version": "6",
                        "properties": [
                            "OnOff"
                        ],
                        "actions": [
                            "Off",
                            "On"
                        ],
                        "events": []
                    }
                    ...
}
```

Per ulteriori informazioni, consulta il get-managed-thing-capabilitiescomando nella Guida ai comandi delle integrazioni AWS CLI gestite.

Passaggio 6: inviare un comando all'oggetto gestito

Usa il `send-managed-thing-command` comando per inviare un comando di attivazione/disattivazione all'oggetto gestito.

Per inviare un comando alla cosa gestita

- Usa il `send-managed-thing-command` comando per inviare un comando alla tua cosa gestita.

  Esempio send-managed-thing-command

```
json=$(jq -cr '.|@json') <<EOF
[
  {
    "endpointId": "1",
    "capabilities": [
      {
        "id": "matter.OnOff@1.4",
        "name": "On/Off",
        "version": "1",
        "actions": [
          {
            "name": "Toggle",
            "parameters": {}
          }
        ]
      }
    ]
  }
]
EOF
aws iot-managed-integrations send-managed-thing-command \
--managed-thing-id "DEVICE_MANAGED_THING_ID" --endpoints "ENDPOINT_ID"
```

> **Note**
>
> Questo esempio usa jq cli to ma puoi anche passare l'intera stringa `endpointId`

  Risposta:

```
{
"TraceId": "TRACE_ID"
}
```

Per ulteriori informazioni, consulta il send-managed-thing-commandcomando nel Command Reference delle integrazioni AWS CLI gestite.

Passaggio 7: rimuovi l'elemento gestito dal tuo hub

Pulisci l'hub rimuovendo l'elemento gestito.

Per eliminare un oggetto gestito

- Usa il `delete-managed-thing` comando per rimuovere un elemento gestito dall'hub del dispositivo.

  Esempio delete-managed-thing

  ```
  aws iot-managed-integrations delete-managed-thing \
  --identifier "DEVICE_MANAGED_THING_ID"
  ```

  Per ulteriori informazioni, consulta il delete-managed-thingcomando nel Command Reference delle integrazioni AWS CLI gestite.

  > **Note**
  >
  > Se il dispositivo è bloccato in uno `DELETE_IN_PROGRESS` stato, aggiungi la `--force` bandiera al. `delete-managed-thing command`

  > **Note**
  >
  > Per i dispositivi Z-wave, è necessario mettere il dispositivo in modalità di associazione dopo aver eseguito il comando.

Configurazione guidata dall'utente per l'installazione e il funzionamento dei dispositivi

Configura i tuoi dispositivi per l'onboarding nel tuo hub di integrazioni gestite creando un oggetto gestito e collegandolo al tuo hub. Questa sezione descrive i passaggi per completare il processo di onboarding del dispositivo utilizzando la configurazione guidata dall'utente.

Prerequisiti

Completa questi passaggi prima di tentare l'onboard di un dispositivo:

- Effettua l'onboarding di un dispositivo hub nell'hub di integrazioni gestite.
- Installa la versione più recente di AWS CLI dal Managed AWS CLI Integrations Command Reference
- Iscriviti alle notifiche degli eventi DEVICE_DISCOVERY-STATUS.

Fasi di configurazione guidate dall'utente

- Prerequisito: abilitare la modalità di associazione sul dispositivo Z Wave
- Passaggio 1: avvia l'individuazione del dispositivo
- Passaggio 2: interrogare l'ID del processo di rilevamento
- Passaggio 3: crea un oggetto gestito per il tuo dispositivo
- Passaggio 4: interroga la cosa gestita
- Fase 5: Ottieni funzionalità gestite per gli oggetti
- Passaggio 6: inviare un comando all'oggetto gestito
- Passaggio 7: verifica lo stato dell'oggetto gestito
- Passaggio 8: rimuovi la cosa gestita dal tuo hub

Prerequisito: abilitare la modalità di associazione sul dispositivo Z Wave

Abilita la modalità di associazione sul dispositivo Z-wave. La modalità di associazione può variare per ogni dispositivo Z-Wave, quindi consulta le istruzioni del dispositivo per configurare correttamente la modalità di associazione. Di solito è un pulsante che l'utente deve premere.

Passaggio 1: avvia l'individuazione del dispositivo

Avvia il rilevamento dei dispositivi per l'hub per ottenere un Discovery Job ID da utilizzare per effettuare l'onboard del dispositivo.

Per avviare l'individuazione dei dispositivi

- Utilizzare il start-device-discoverycomando per ottenere l'ID del processo di rilevamento.

  Esempio start-device-discovery

```
#For Zigbee
  aws iot-managed-integrations start-device-discovery --discovery-type ZIGBEE \
  --controller-identifier HUB_MANAGED_THING_ID

  #For Zwave
  aws iot-managed-integrations start-device-discovery --discovery-type ZWAVE \
  --controller-identifier HUB_MANAGED_THING \
  --authentication-material-type ZWAVE_INSTALL_CODE \
  --authentication-material 13333

  #For Cloud
  aws iot-managed-integrations start-device-discovery --discovery-type CLOUD \
  --account-association-id C2C_ASSOCIATION_ID \

  #For Custom
  aws iot-managed-thing start-device-discovery --discovery-type CUSTOM \
  --controller-identifier HUB_MANAGED_THING_ID \
  --custom-protocol-detail NAME : NON_EMPTY_STRING \
```

  Risposta:

```
{
      "Id": DISCOVERY_JOB_ID,
      "StartedAt": "2025-06-03T14:43:12.726000-07:00"
  }
```

> **Note**
>
> Esistono comandi separati per i dispositivi Z-wave e Zigbee.

Per ulteriori informazioni, consulta l'start-device-discoveryAPI nella sezione Command Reference delle integrazioni AWS CLI gestite.

Passaggio 2: interrogare l'ID del processo di rilevamento

Usa il `list-discovered-devices` comando per ottenere il materiale di autenticazione del tuo dispositivo.

Per interrogare il tuo Discovery Job ID

- Utilizza il discovery job ID con il list-discovered-devices comando per ottenere il materiale di autenticazione del tuo dispositivo.

```
aws iot-managed-integrations list-discovered-devices --identifier DISCOVERY_JOB_ID
```

Risposta:

```
"Items": [
    {
        "DeviceTypes": [],
        "DiscoveredAt": "2025-06-03T14:43:37.619000-07:00",
        "AuthenticationMaterial": AUTHENTICATION_MATERIAL
    }
]
```

Passaggio 3: crea un oggetto gestito per il tuo dispositivo

Usa il `create-managed-thing` comando per creare un oggetto gestito per il tuo dispositivo. Ogni dispositivo richiede la propria funzionalità gestita.

Per creare un oggetto gestito

- Usa il `create-managed-thing` comando per creare un oggetto gestito per il tuo dispositivo.

  Esempio create-managed-thing

```
aws iot-managed-integrations create-managed-thing \
  --role  DEVICE --authentication-material-type DISCOVERED_DEVICE \
```

```
  --authentication-material "AUTHENTICATION_MATERIAL"
```

Risposta:

```
{
    "Id": "DEVICE_MANAGED_THING_ID"
    "Arn": "arn:aws:iotmanagedintegrations:AWS_REGION:AWS_ACCOUNT_ID:managed-
thing/DEVICE_MANAGED_THING_ID"
    "CreatedAt": "2025-06-09T13:58:52.977000+08:00"
  }
```

Per ulteriori informazioni, consulta il create-managed-thingcomando nella Guida ai comandi delle integrazioni AWS CLI gestite.

Passaggio 4: interroga la cosa gestita

È possibile verificare se una cosa gestita è attivata utilizzando il `get-managed-thing` comando.

Per interrogare una cosa gestita

- Usa il `get-managed-thing` comando per verificare se lo stato di provisioning dell'oggetto gestito è impostato `ACTIVATED` su. Per ulteriori informazioni sullo stato del provisioning, vedere Device Provisioning.

  Esempio get-managed-thing

  ```
  aws iot-managed-integrations get-managed-thing \
    --identifier "DEVICE_MANAGED_THING_ID"
  ```

  Risposta:

  ```
  {
        "Id": "DEVICE_MANAGED_THING_ID",
        "Arn": "arn:aws:iotmanagedintegrations:AWS_REGION:AWS_ACCOUNT_ID:managed-
  thing/DEVICE_MANAGED_THING_ID,
        "Role": "DEVICE",
        "ProvisioningStatus": "ACTIVATED",
        "MacAddress": "MAC_ADDRESS",
        "ParentControllerId": "PARENT_CONTROLLER_ID",
        "CreatedAt": "2025-06-03T14:46:35.149000-07:00",
  ```

```
      "UpdatedAt": "2025-06-03T14:46:37.500000-07:00",
      "Tags": {}
  }
```

Per ulteriori informazioni, consulta il get-managed-thingcomando nel Command Reference delle integrazioni AWS CLI gestite.

Fase 5: Ottieni funzionalità gestite per gli oggetti

È possibile visualizzare un elenco delle azioni disponibili di un oggetto gestito utilizzando`get-managed-thing-capabiltiies`.

Per sfruttare le funzionalità di un dispositivo

- Usa il `get-managed-thing-capabilities` comando per ottenere l'ID dell'endpoint. Nota anche l'elenco delle azioni possibili.

  Esempio get-managed-thing-capabilities

```
aws iot-managed-integrations get-managed-thing-capabilities \
  --identifier "DEVICE_MANAGED_THING_ID"
```

  Risposta:

```
{
      "ManagedThingId": "DEVICE_MANAGED_THING_ID",
      "CapabilityReport": {
          "version": "1.0.0",
          "nodeId": "zb.539D+4A1D",
          "endpoints": [
              {
                  "id": "1",
                  "deviceTypes": [
                      "Unknown Device"
                  ],
                  "capabilities": [
                      {
                          "id": "matter.OnOff@1.4",
                          "name": "On/Off",
                          "version": "6",
                          "properties": [
```

```
                            "OnOff",
                            "OnOff",
                            "OnTime",
                            "OffWaitTime"
                        ],
                        "actions": [
                            "Off",
                            "On",
                            "Toggle",
                            "OffWithEffect",
                            "OnWithRecallGlobalScene",
                            "OnWithTimedOff"
                        ],
                        ...
  }
```

Per ulteriori informazioni, consulta il get-managed-thing-capabilitiescomando nel Command Reference delle integrazioni AWS CLI gestite.

Passaggio 6: inviare un comando all'oggetto gestito

È possibile utilizzare il send-managed-thing-command comando per inviare un comando di attivazione e disattivazione all'oggetto gestito.

Invia un comando all'oggetto gestito utilizzando un'azione di commutazione.

- Usa il `send-managed-thing-command` comando per inviare un comando toggle action.

  Esempio send-managed-thing-command

```
json=$(jq -cr '.|@json') <<EOF
  [
    {
      "endpointId": "1",
      "capabilities": [
        {
          "id": "matter.OnOff@1.4",
          "name": "On/Off",
          "version": "1",
          "actions": [
            {
              "name": "Toggle",
```

```
              "parameters": {}
            }
          ]
        }
      ]
    }
  ]
  EOF
  aws iot-managed-integrations send-managed-thing-command \
  --managed-thing-id ${device_managed_thing_id} --endpoints ENDPOINT_ID
```

> **Note**
>
> Questo esempio usa jq cli to ma puoi anche passare l'intera stringa `endpointId`

Risposta:

```
{
  "TraceId": TRACE_ID
  }
```

Per ulteriori informazioni, consulta il send-managed-thing-commandcomando nel Command Reference delle integrazioni AWS CLI gestite.

Passaggio 7: verifica lo stato dell'oggetto gestito

Controlla lo stato dell'oggetto gestito per confermare che l'azione di attivazione è riuscita.

Per controllare lo stato del dispositivo di un oggetto gestito

- Usa il `get-managed-thing-state` comando per confermare che l'azione di attivazione è riuscita.

  Esempio get-managed-thing-state

```
  aws iot-managed-integrations get-managed-thing-state --managed-thing-
id DEVICE_MANAGED_THING_ID
```

Risposta:

```
{
      "Endpoints": [
          {
              "endpointId": "1",
              "capabilities": [
                  {
                      "id": "matter.OnOff@1.4",
                      "name": "On/Off",
                      "version": "1.4",
                      "properties": [
                          {
                              "name": "OnOff",
                              "value": {
                                  "propertyValue": true,
                                  "lastChangedAt": "2025-06-03T21:50:39.886Z"
                              }
                          }
                      ]
                  }
              ]
          }
      ]
  }
```

Per ulteriori informazioni, consulta il get-managed-thing-statecomando nel Command Reference delle integrazioni gestite. AWS CLI

Passaggio 8: rimuovi la cosa gestita dal tuo hub

Pulisci l'hub rimuovendo l'elemento gestito.

Per eliminare un oggetto gestito

- Usa il delete-managed-thingcomando per rimuovere un oggetto gestito.

  Esempio delete-managed-thing

```
aws iot-managed-integrations delete-managed-thing \
```

```
  --identifier MANAGED_THING_ID
```

Per ulteriori informazioni, consulta il delete-managed-thingcomando nella Guida ai comandi delle integrazioni AWS CLI gestite.

> **Note**
>
> Se il dispositivo è bloccato in uno `DELETE_IN_PROGRESS` stato, aggiungi il `--force` flag al `delete-managed-thing` comando.

> **Note**
>
> Per i dispositivi Z-wave, è necessario mettere il dispositivo in modalità di associazione dopo aver eseguito il comando.

Crea un gestore di certificati personalizzato per l'archiviazione sicura

La gestione dei certificati dei dispositivi è fondamentale per l'onboarding dell'hub di integrazioni gestite. Sebbene i certificati siano archiviati nel file system per impostazione predefinita, puoi creare un gestore di certificati personalizzato per una maggiore sicurezza e una gestione flessibile delle credenziali.

L'SDK per le integrazioni gestite End device fornisce un gestore di certificati per proteggere l'interfaccia di archiviazione che puoi implementare come libreria di oggetti condivisi (.so). Crea la tua implementazione di archiviazione sicura per leggere e scrivere certificati, quindi collega il file della libreria al processo in fase di esecuzione. HubOnboarding

Definizione e componenti dell'API

Consulta il seguente `secure_storage_cert_handler_interface.hpp` file per comprendere i componenti e i requisiti dell'API per la tua implementazione

Argomenti

- Definizione dell'API

- Componenti chiave

Definizione dell'API

Contenuto di **`secure_storage_cert_hander_interface.hpp`**:

```
/*
    * Copyright 2024 Amazon.com, Inc. or its affiliates. All rights reserved.
    *
    * AMAZON PROPRIETARY/CONFIDENTIAL
    *
    * You may not use this file except in compliance with the terms and
    * conditions set forth in the accompanying LICENSE.txt file.
    *
    * THESE MATERIALS ARE PROVIDED ON AN "AS IS" BASIS. AMAZON SPECIFICALLY
    * DISCLAIMS, WITH RESPECT TO THESE MATERIALS, ALL WARRANTIES, EXPRESS,
    * IMPLIED, OR STATUTORY, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY,
    * FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.
    */
    #ifndef SECURE_STORAGE_CERT_HANDLER_INTERFACE_HPP
    #define SECURE_STORAGE_CERT_HANDLER_INTERFACE_HPP

    #include <iostream>
    #include <memory>

    namespace IoTManagedIntegrationsDevice {
    namespace CertHandler {
    /**
     * @enum CERT_TYPE_T
     * @brief enumeration defining certificate types.
     */
     typedef enum { CLAIM = 0, DHA = 1, PERMANENT = 2 } CERT_TYPE_T;
     class SecureStorageCertHandlerInterface {
      public:
       /**
        * @brief Read certificate and private key value of a particular certificate
        * type from secure storage.
        */
        virtual bool read_cert_and_private_key(const CERT_TYPE_T cert_type,
                                               std::string &cert_value,
                                               std::string &private_key_value) = 0;
        /**
          * @brief Write permanent certificate and private key value to secure storage.
```

```
         */
        virtual bool write_permanent_cert_and_private_key(
            std::string_view cert_value, std::string_view private_key_value) = 0;
        };
        std::shared_ptr<SecureStorageCertHandlerInterface>
createSecureStorageCertHandler();
    } //namespace CertHandler
    } //namespace IoTManagedIntegrationsDevice

    #endif //SECURE_STORAGE_CERT_HANDLER_INTERFACE_HPP
```

Componenti chiave

- CERT_TYPE_T: diversi tipi di certificati sull'hub.
  - CLAIM: il certificato di reclamo originariamente presente nell'hub verrà sostituito con un certificato permanente.
  - DHA: inutilizzato per ora.
  - PERMANENTE: certificato permanente per la connessione con l'endpoint delle integrazioni gestite.
- read_cert_and_private_key - (FUNZIONE DA IMPLEMENTARE) Legge il certificato e il valore della chiave nell'input di riferimento. Questa funzione deve essere in grado di leggere sia il certificato CLAIM che quello PERMANENT ed è differenziata in base al tipo di certificato sopra menzionato.
- write_permanent_cert_and_private_key - (FUNZIONE DA IMPLEMENTARE) scrive il certificato permanente e il valore della chiave nella posizione desiderata.

Esempio di build

Separa le intestazioni di implementazione interne dall'interfaccia pubblica (`secure_storage_cert_handler_interface.hpp`) per mantenere una struttura di progetto pulita. Con questa separazione, puoi gestire i componenti pubblici e privati mentre crei il tuo gestore di certificati.

> **Note**
>
> Dichiara `secure_storage_cert_handler_interface.hpp` come pubblico.

Argomenti

Struttura del progetto

Eredita l'interfaccia

Crea una classe concreta che erediti l'interfaccia. Nascondi questo file di intestazione e altri file in una directory separata in modo che le intestazioni private e pubbliche possano essere facilmente differenziate durante la creazione.

```
#ifndef IOTMANAGEDINTEGRATIONSDEVICE_SDK_STUB_SECURE_STORAGE_CERT_HANDLER_HPP
  #define IOTMANAGEDINTEGRATIONSDEVICE_SDK_STUB_SECURE_STORAGE_CERT_HANDLER_HPP

  #include "secure_storage_cert_handler_interface.hpp"

  namespace IoTManagedIntegrationsDevice::CertHandler {
    class StubSecureStorageCertHandler : public SecureStorageCertHandlerInterface {
      public:
        StubSecureStorageCertHandler() = default;

        bool read_cert_and_private_key(const CERT_TYPE_T cert_type,
                                       std::string &cert_value,
                                       std::string &private_key_value) override;

        bool write_permanent_cert_and_private_key(
            std::string_view cert_value, std::string_view private_key_value) override;
            /*
            * any other resource for function you might need
            */


          };
      }
    #endif //IOTMANAGEDINTEGRATIONSDEVICE_SDK_STUB_SECURE_STORAGE_CERT_HANDLER_HPP
```

Implementazione

Implementa la classe di archiviazione definita sopra,. `src/stub_secure_storage_cert_handler.cpp`

```
/*
  * Copyright 2024 Amazon.com, Inc. or its affiliates. All rights reserved.
  *
  * AMAZON PROPRIETARY/CONFIDENTIAL
  *
  * You may not use this file except in compliance with the terms and
  * conditions set forth in the accompanying LICENSE.txt file.
  *
  * THESE MATERIALS ARE PROVIDED ON AN "AS IS" BASIS. AMAZON SPECIFICALLY
  * DISCLAIMS, WITH RESPECT TO THESE MATERIALS, ALL WARRANTIES, EXPRESS,
  * IMPLIED, OR STATUTORY, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY,
  * FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.
  */

  #include "stub_secure_storage_cert_handler.hpp"

  using namespace IoTManagedIntegrationsDevice::CertHandler;

  bool StubSecureStorageCertHandler::write_permanent_cert_and_private_key(
              std::string_view cert_value, std::string_view private_key_value) {
            // TODO: implement write function
            return true;
  }

  bool StubSecureStorageCertHandler::read_cert_and_private_key(const CERT_TYPE_T
cert_type,
                                                               std::string &cert_value,
                                                               std::string
&private_key_value) {
          std::cout<<"Using Stub Secure Storage Cert Handler, returning dummy values";
          cert_value = "StubCertVal";
          private_key_value = "StubKeyVal";
          // TODO: implement read function
          return true;
  }
```

Implementa la funzione di fabbrica definita nell'interfaccia,`src/secure_storage_cert_handler.cpp`.

```
#include "stub_secure_storage_cert_handler.hpp"


 std::shared_ptr<IoTManagedIntegrationsDevice::CertHandler::SecureStorageCertHandlerInterface>
       IoTManagedIntegrationsDevice::CertHandler::createSecureStorageCertHandler() {
         // TODO: replace with your implementation
       return
 std::make_shared<IoTManagedIntegrationsDevice::CertHandler::StubSecureStorageCertHandler>();
      }
```

CMakeList.txt

```
#project name must stay the same
      project(SecureStorageCertHandler)

      # Public Header files. The interface definition must be in top level with exactly
 the same name
      #ie. Not in anotherDir/secure_storage_cert_hander_interface.hpp
      set(PUBLIC_HEADERS
                ${PROJECT_SOURCE_DIR}/include
      )

      # private implementation headers.
      set(PRIVATE_HEADERS
                ${PROJECT_SOURCE_DIR}/internal/stub
      )

      #set all sources
      set(SOURCES
                ${PROJECT_SOURCE_DIR}/src/secure_storage_cert_handler.cpp
                ${PROJECT_SOURCE_DIR}/src/stub_secure_storage_cert_handler.cpp
        )

      # Create the shared library
      add_library(${PROJECT_NAME} SHARED ${SOURCES})
```

```
    target_include_directories(
              ${PROJECT_NAME}
              PUBLIC
                  ${PUBLIC_HEADERS}
              PRIVATE
                  ${PRIVATE_HEADERS}
    )

    # Set the library output location. Location can be customized but version must
stay the same
    set_target_properties(${PROJECT_NAME} PROPERTIES
              LIBRARY_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/../lib
              VERSION 1.0
              SOVERSION 1
    )

    # Install rules
    install(TARGETS ${PROJECT_NAME}
              LIBRARY DESTINATION lib
              ARCHIVE DESTINATION lib
    )

    install(FILES ${HEADERS}
              DESTINATION include/SecureStorageCertHandler
    )
```

Utilizzo

Dopo la compilazione, avrai un file di libreria di oggetti `libSecureStorageCertHandler.so` condiviso e i relativi link simbolici associati. Copiate sia il file di libreria che i collegamenti simbolici nella posizione della libreria prevista dal file binario. HubOnboarding

Argomenti

- Considerazioni chiave
- Usa l'archiviazione sicura

Considerazioni chiave

- Verifica che il tuo account utente disponga delle autorizzazioni di lettura e scrittura sia per il HubOnboarding file binario che per la libreria. `libSecureStorageCertHandler.so`
- Conservalo `secure_storage_cert_handler_interface.hpp` come unico file di intestazione pubblico. Tutti gli altri file di intestazione devono rimanere nell'implementazione privata.
- Verifica il nome della tua libreria di oggetti condivisi. Durante la compilazione`libSecureStorageCertHandler.so`, HubOnboarding potrebbe essere necessaria una versione specifica nel nome del file, ad esempio. `libSecureStorageCertHandler.so.1.0` Utilizzate il `ldd` comando per controllare le dipendenze delle librerie e creare collegamenti simbolici secondo necessità.
- Se l'implementazione della libreria condivisa ha dipendenze esterne, memorizzale in una directory HubOnboarding accessibile, ad esempio una directory. `/usr/lib or the iotmi_common`

Usa l'archiviazione sicura

Aggiorna il `iotmi_config.json` file impostando entrambe le impostazioni `iot_claim_cert_path` e `iot_claim_pk_path` su**`SECURE_STORAGE`**.

```
{
  "ro": {
    "iot_provisioning_method": "FLEET_PROVISIONING",
    "iot_claim_cert_path": "SECURE_STORAGE",
    "iot_claim_pk_path": "SECURE_STORAGE",
    "fp_template_name": "device-integration-example",
    "iot_endpoint_url": "[ACCOUNT-PREFIX]-ats.iot.AWS-REGION.amazonaws.com",
    "SN": "1234567890",
    "UPC": "1234567890"
  },
  "rw": {
    "iot_provisioning_state": "NOT_PROVISIONED"
  }
}
```

Plugin di protocollo personalizzato

Puoi utilizzare un plug-in di protocollo personalizzato per integrare i tuoi protocolli IoT proprietari nelle integrazioni gestite per AWS IoT Device Management l'ecosistema. Tramite interfacce SDK ben

definite, è possibile integrare dispositivi, definire funzionalità e gestire flussi di controllo in tempo reale mantenendo la piena compatibilità con le integrazioni gestite e i componenti SDK dell'hub.

Gli elenchi seguenti illustrano le funzionalità principali del plug-in del protocollo personalizzato.

Personalizzazione del modello di dati

Definisci i tuoi schemi di modelli di AWS dati e caricali nelle integrazioni gestite durante il flusso di provisioning. Puoi utilizzare questi schemi in un secondo momento nei tuoi flussi di lavoro.

Implementazione flessibile del plug-in

- Crea il tuo componente di plugin usandoClient Hub SDK.
- Implementa plugin separati per diverse funzionalità, come il provisioning e il controllo, o crea un client unificato per entrambe.
- Mantieni un confine chiaro tra le risorse di integrazione gestite e le tue risorse, come gli stack di middleware, per un'implementazione logica di codice disaccoppiata e adatta allo sviluppo.

Compatibilità con le versioni precedenti

Per i clienti esistenti, è possibile integrare senza problemi il nuovo protocollo personalizzato, mantenendo allo stesso tempo i tipi di radio esistenti funzionanti così come sono.

Il diagramma seguente illustra l'architettura del plug-in del protocollo personalizzato.

Client Hub SDK

La libreria Hub SDK Client funge da interfaccia tra le integrazioni gestite Hub SDK e lo stack di protocolli personalizzato in esecuzione sullo stesso hub. Espone una serie di elementi pubblici APIs per facilitare l'interazione dello stack di protocolli con i componenti Device Hub SDK. I casi d'uso includono il controllo personalizzato dei plug-in, il provider di plug-in personalizzato e il controller locale.

Argomenti

- Ottieni le tue integrazioni gestite Hub SDK
- Informazioni sul toolkit Hub SDK
- Crea la tua applicazione personalizzata con il client Hub SDK
- Esecuzione dell'applicazione personalizzata

- Riferimento all'API del client Hub SDK
- Tipi di dati

Ottieni le tue integrazioni gestite Hub SDK

Il client Hub SDK viene fornito con l'SDK per le integrazioni gestite. Contattaci dalla console delle integrazioni gestite per accedere all'hub SDK.

Informazioni sul toolkit Hub SDK

Dopo il download, verrà visualizzata una `IotMI-DeviceSDK-Toolkit` cartella che contiene tutti i file di intestazione pubblici e i `.so` file che è possibile utilizzare nell'applicazione. Il team dedicato alle integrazioni gestite fornisce anche un `main.cpp` esempio a scopo dimostrativo, oltre al binario dell'applicazione demo `bin/` eseguibile direttamente. Facoltativamente, puoi utilizzarlo come punto di partenza per la tua applicazione.

Crea la tua applicazione personalizzata con il client Hub SDK

Utilizza i seguenti passaggi per creare la tua applicazione personalizzata.

1. Includete i file di intestazione (.h) e i file di oggetti condivisi (.so) nell'applicazione.

   È necessario includere i file di intestazione pubblici (.h) e i file di oggetti condivisi (.so) nell'applicazione. Per i file.so, potete metterli in una cartella lib. Il layout finale sarà simile al seguente:

   ```
   ### include
   #   ### iotmi_device_sdk_client
   #   #   ### iotmi_device_sdk_client_common_types.h
   #   ### iotmi_device_sdk_client.h
   #   ### iotshd_status.h
   ### lib
   #   ### libiotmi_devicesdk_client_module.so
   #   ### libiotmi_log_c.so
   ```

2. Crea un client Hub SDK nell'applicazione principale.

   a. Nell'applicazione principale, è necessario inizializzare il client Hub SDK prima che possa essere utilizzato per elaborare le richieste. Puoi semplicemente creare il client con un. `clientId`

b. Dopo aver ottenuto il client, puoi collegarlo al Device SDK delle integrazioni gestite.

   Di seguito è riportato un esempio di creazione del client Hub SDK e di come connettersi.

```
#include <cstdlib>
#include <string>
#include "iotshd_status.h"
#include "iotmi_device_sdk_client.h"

auto client = std::make_unique<DeviceSDKClient>(your_own_clientId);
iotmi_statusCode_t status = client->connect();
```

> **Note**
>
> *your_own_clientId*deve essere uguale a quello specificato nella configurazione guidata dall'utente o start-device-discoverynel flusso di provisioning di Simple Setup. create-managed-thing

3. Pubblica e sottoscrivi effettuando le seguenti operazioni.

   a. Dopo aver stabilito la connessione, ora puoi iscriverti alle attività in entrata dall'Hub SDK per le integrazioni gestite. Le attività in entrata possono essere attività di controllo o attività di fornitura. È inoltre necessario definire la propria funzione di callback in base alle attività ricevute e il proprio contesto personalizzato per i propri scopi di tracciamento.

```
    // subscribe to provisioning tasks
    iotmi_statusCode_t status = client->iotmi_provision_subscribe_to_tasks(
    example_subscriber_callback, custom_context);

    // subscribe to control tasks
    iotmi_statusCode_t status = client->iotmi_control_subscribe_to_tasks(
    example_subscriber_callback, custom_context);
```

   b. Dopo aver stabilito la connessione, ora puoi pubblicare le richieste dall'applicazione all'SDK Hub per le integrazioni gestite. È possibile definire il proprio tipo di messaggio di attività, con payload diversi per scopi aziendali diversi. Le richieste possono includere sia richieste di controllo che richieste di fornitura, in modo simile al flusso di iscrizione. Infine, puoi assegnare un indirizzo per `rspPayload` ricevere la risposta dall'Hub SDK delle integrazioni gestite in modo sincronizzato.

```
// publish control request
  iotmi_client_request_t api_payload = {
            .messageType = C2MIMessageType::C2MI_CONTROL_EVENT,
            .reqPayload = (uint8_t *)"define_your_req_payload",
            .rspPayload = (uint8_t *)calloc(1000, sizeof(uint8_t))
  };

  status = client->iotmi_control_publish_request(&api_payload);

  // publish provision request
  iotmi_client_request_t api_payload = {
            .messageType = C2MIMessageType::C2MI_DEVICE_ONBOARDED,
            .reqPayload = (uint8_t *)"define_your_req_payload",
            .rspPayload = (uint8_t *)calloc(1000, sizeof(uint8_t))
  };
  status = client->iotmi_provision_publish_request(&api_payload);
```

4. Creane uno tuo `CMakeLists.txt` e crea la tua applicazione partendo da lì. L'output finale potrebbe essere un file binario eseguibile come `MyFirstApplication`

Esecuzione dell'applicazione personalizzata

Prima di eseguire l'applicazione personalizzata, completa i seguenti passaggi per configurare l'hub e avviare le integrazioni gestite Hub SDK:

- Segui le istruzioni di onboarding all'indirizzo. Integra i tuoi hub verso integrazioni gestite
- Completa il processo di installazione documentato in. Installa e convalida le integrazioni gestite Hub SDK

Una volta soddisfatti i prerequisiti, è possibile eseguire l'applicazione personalizzata. Per esempio:

```
./MyFirstApplication
```

> **Important**
>
> È necessario aggiornare manualmente lo script di avvio Implementa l'Hub SDK con uno script elencato in uno script per avviare la propria applicazione. L'ordine è importante, non modificarlo.

Aggiorna quanto segue. Modifica

```
./IotMI-DeviceSDK-Toolkit/bin/DeviceSDKClientDemo >> $LOGS_DIR/
logDeviceSDKClientDemo_logs.txt &
```

in

```
./MyFirstApplication >> $LOGS_DIR/MyFirstApplication_logs.txtt &
```

Riferimento all'API del client Hub SDK

Il client Hub SDK (SDKClient classe Device) fornisce un'interfaccia per l'applicazione personalizzata per interagire con le integrazioni gestite Device SDK. Con questo client, puoi fare quanto segue:

- Iscriviti alle attività relative alla fornitura e al controllo dai componenti delle integrazioni gestite.
- Pubblica le richieste relative alla fornitura e al controllo nei componenti delle integrazioni gestite.

Per informazioni sulle integrazioni gestite per, consulta What is. AWS IoT Device Management APIs AWS Lambda

Argomenti

- Inizializzazione del client
- Abbonamento alle attività di fornitura
- Pubblicazione delle attività di fornitura
- Abbonamento alle attività di controllo
- Controlla la pubblicazione delle attività
- Funzionalità di registrazione
- Altro APIs

Inizializzazione del client

Per iniziare a utilizzare`DeviceSDKClient`, inizializzalo con un ID client.

```
iotmi_statusCode_t DeviceSDKClient(const std::string& clientId)
```

Questo crea una nuova `DeviceSDKClient` istanza con il valore specificato`clientId`. `clientId`Deve corrispondere a quella registrata con le integrazioni gestite.

Parametri

`clientId`(stringa): l'ID client per questa istanza.

```
connect()
```

Collega l'`DeviceSDKClient`istanza alle integrazioni gestite.

Valori restituiti

- `IOTMI_STATUS_OK`- La connessione è avvenuta con successo.
- `IOTMI_STATUS_CUSTOM_PLUGIN_CONNECTION_ERROR`- Si è verificato un errore durante la connessione alle integrazioni gestite.

Abbonamento alle attività di fornitura

Utilizza questi metodi per sottoscrivere le attività relative alla fornitura dai componenti delle integrazioni gestite.

```
iotmi_statusCode_t
 iotmi_provision_subscribe_to_tasks(DeviceSDKClient_SubscriberCallback callback, char*
 context)
```

Sottoscrive le attività relative al provisioning, come l'onboarding e il deprovisioning dei dispositivi, dai componenti delle integrazioni gestite.

Parametri

- `callback`(Device SDKClient _SubscriberCallback) - Una funzione di callback che viene eseguita quando viene ricevuta un'attività.
- `context`(char*) - Un contesto personalizzato passato alla funzione di callback.

Valori restituiti

- `IOTMI_STATUS_OK`- L'iscrizione è avvenuta con successo.
- `IOTMI_STATUS_CUSTOM_PLUGIN_CLIENT_NOT_CONNECTED`- L'SDKClient istanza Device non è connessa alle integrazioni gestite.

- `IOTMI_STATUS_CUSTOM_PLUGIN_SUBSCRIBE_ERROR`- Si è verificato un errore durante la sottoscrizione alle attività.

Pubblicazione delle attività di fornitura

Utilizza questi metodi per pubblicare le richieste relative alla fornitura nei componenti delle integrazioni gestite.

```
iotmi_statusCode_t iotmi_provision_publish_request(DataModel::iotmi_client_request_t
 request)
```

Pubblica una richiesta relativa alla fornitura nei componenti delle integrazioni gestite. Ad esempio, un evento di onboarding del dispositivo o lo stato di deprovisioning

Parametri

`request`(DataModel: :iotmi_client_request_t) - Un puntatore a una struttura di richiesta contenente i dettagli.

Valori restituiti

- `IOTMI_STATUS_OK`- La richiesta è stata pubblicata con successo.
- `IOTMI_STATUS_CUSTOM_PLUGIN_CLIENT_NOT_CONNECTED`- L'`DeviceSDKClient`istanza non è connessa alle integrazioni gestite.
- `IOTMI_STATUS_INVALID_PARAMETER`- Uno o più parametri nella richiesta non sono validi.
- `IOTMI_STATUS_INVALID_JSON_OBJECT`- Il payload della richiesta non è un oggetto JSON valido.
- `IOTMI_STATUS_NO_MEMORY`- Si è verificato un errore di allocazione della memoria.

Abbonamento alle attività di controllo

Utilizza questi metodi per sottoscrivere le attività relative al controllo dai componenti delle integrazioni gestite.

```
iotmi_statusCode_t iotmi_control_subscribe_to_tasks(DeviceSDKClient_SubscriberCallback
 callback, char context)
```

Sottoscrive le attività relative al controllo (ad esempio, le richieste di controllo dei dispositivi) dai componenti delle integrazioni gestite.

Parametri

- `callback`(Device SDKClient _SubscriberCallback) - Una funzione di callback che viene eseguita quando viene ricevuta un'attività.
- `context`(char) - Un contesto personalizzato passato alla funzione di callback.

Valori restituiti

- `IOTMI_STATUS_OK`- L'iscrizione è avvenuta con successo.
- `IOTMI_STATUS_CUSTOM_PLUGIN_CLIENT_NOT_CONNECTED`- L'`DeviceSDKClient`istanza non è connessa alle integrazioni gestite.
- `IOTMI_STATUS_CUSTOM_PLUGIN_SUBSCRIBE_ERROR`- Si è verificato un errore durante la sottoscrizione alle attività.

Controlla la pubblicazione delle attività

Utilizza questi metodi per pubblicare le richieste relative al controllo nei componenti delle integrazioni gestite.

```
iotmi_statusCode_t iotmi_control_publish_request(DataModel::iotmi_client_request_t
 request)
```

Pubblica una richiesta relativa al controllo nei componenti delle integrazioni gestite. Ad esempio, eventi non richiesti, richieste di comandi o domande sullo stato del dispositivo.

Parametri

`request`(DataModel: :iotmi_client_request_t) - Un puntatore a una struttura di richiesta contenente i dettagli.

Valori restituiti

- `IOTMI_STATUS_OK`- La richiesta è stata pubblicata con successo.
- `IOTMI_STATUS_CUSTOM_PLUGIN_CLIENT_NOT_CONNECTED`- L'SDKClient istanza del dispositivo non è connessa alle integrazioni gestite.
- `IOTMI_STATUS_INVALID_PARAMETER`- Uno o più parametri nella richiesta non sono validi.
- `IOTMI_STATUS_INVALID_JSON_OBJECT`- Il payload della richiesta non è un oggetto JSON valido.
- `IOTMI_STATUS_NO_MEMORY`- Si è verificato un errore di allocazione della memoria.

Funzionalità di registrazione

Utilizza questi metodi per implementare le funzionalità di registrazione fornite dalle integrazioni gestite.

Inizializzazione del logger

```
void iotmi_devicesdk_log_init(const char* logger_name)
```

È necessario inizializzare il logger prima di utilizzare qualsiasi funzionalità di registrazione.

Parametri

`logger_name`- Il nome del logger specificato. Il valore predefinito è: `MyApplication`

Macro di registrazione

`LOGGER_LOGD(...)`

Utilizzate questa macro nell'applicazione per la registrazione a livello DEBUG.

`LOGGER_LOGI(...)`

Utilizzate questa macro nell'applicazione per la registrazione a livello INFO.

`LOGGER_LOGW(...)`

Utilizzate questa macro nell'applicazione per la registrazione a livello WARN.

`LOGGER_LOGE(...)`

Utilizzate questa macro nell'applicazione per la registrazione del livello di ERRORE.

> **Note**
>
> Per ulteriori informazioni sulle funzionalità di registrazione, consultate la documentazione sulla registrazione dell'Hub. I plugin di protocollo personalizzato supportano completamente tutte le funzionalità di registrazione offerte dalle integrazioni gestite.

Altro APIs

```
std::string get_client_id()
```

Restituisce l'ID client associato all'SDKClient istanza Device.

Valori restituiti

ID client.

Tipi di dati

Questa sezione definisce i tipi di dati utilizzati per il plug-in del protocollo personalizzato.

iotmi_client_request_t

Rappresenta una richiesta da pubblicare nei componenti delle integrazioni gestite.

Tipo di messaggio

Il tipo di messaggio (CommonTypes: MIMessage :C2 Type). L'elenco seguente mostra i valori validi.

- `C2MI_DEVICE_ONBOARDED`: Indica un messaggio di onboarding del dispositivo con relativo payload.
- `C2MI_DE_PROVISIONING_PRE_ASSOCIATED_COMPLETE`: indica una notifica di completamento di un'attività di disattivazione per un dispositivo preassociato.
- `C2MI_DE_PROVISIONING_ACTIVATED_COMPLETE`: indica una notifica di completamento di un'attività di disattivazione per un dispositivo attivato.
- `C2MI_DE_PROVISIONING_COMPLETE_RESPONSE`: indica la risposta completa di un'attività di annullamento del provisioning.
- `C2MI_CONTROL_EVENT`: Indica un evento di controllo con potenziale modifica dello stato del dispositivo.
- `C2MI_CONTROL_SEND_COMMAND`: Indica un comando di controllo da un controller locale.
- `C2MI_CONTROL_SEND_DEVICE_STATE_QUERY`: Indica una richiesta sullo stato del dispositivo di controllo da parte di un controller locale.

ReqPayload

Il payload della richiesta, in genere una stringa in formato JSON.

Payload RSP

Il payload di risposta, popolato dai componenti delle integrazioni gestite.

iotmi_client_event_t

Rappresenta un evento ricevuto dai componenti delle integrazioni gestite.

event_id

L'identificatore univoco dell'evento.

length

La lunghezza dei dati dell'evento.

dati

Un puntatore ai dati dell'evento, incluso il`messageType`. L'elenco seguente mostra i valori possibili.

- `C2MI_PROVISION_UGS_TASK`: Indica un'attività di fornitura per il flusso UGS.
- `C2MI_PROVISION_SS_TASK`: Indica un'attività di fornitura per il SimpleSetup flusso.
- `C2MI_DE_PROVISION_PRE_ASSOCIATED_TASK`: Indica un'attività di annullamento del provisioning per un dispositivo pre-associato.
- `C2MI_DE_PROVISION_ACTIVATED_TASK`: Indica un'attività di annullamento della fornitura per un dispositivo attivato.
- `C2MI_DEVICE_ONBOARDED_RESPONSE`: Indica una risposta all'onboarding del dispositivo.
- `C2MI_CONTROL_TASK`: Indica un'attività di controllo.
- `C2MI_CONTROL_EVENT_NOTIFICATION`: indica una notifica di un evento di controllo per un controller locale.

ctx

Un contesto personalizzato associato all'evento.

Controllo dell'hub

Hub control è un'estensione delle integrazioni gestite End device SDK che gli consente di interfacciarsi con il `MQTTProxy` componente dell'Hub SDK. Con Hub Control, puoi implementare

il codice utilizzando l'SDK del dispositivo finale e controllare l'hub tramite il cloud delle integrazioni gestite come dispositivo separato. L'hub control SDK verrà fornito come pacchetto separato all'interno dell'Hub SDK, etichettato come. `iot-managed-integrations-hub-control-x.x.x`

Argomenti

Prerequisiti

Per configurare Hub Control, è necessario quanto segue:

- Un hub integrato nell'Hub SDK, versione 0.4.0 o successiva.
- Scarica la versione più recente dell'SDK per dispositivi finali dal. Console di gestione AWS
- Un componente proxy MQTT in esecuzione sull'hub, versione 0.5.0 o successiva.

Componenti SDK del dispositivo finale

Utilizza i seguenti componenti dell'SDK del dispositivo finale:

- Generatore di codice per il modello di dati
- Gestore del modello di dati

Poiché Hub SDK dispone già di un processo di onboarding e di una connessione al cloud, non sono necessari i seguenti componenti:

- Fornitore
- Interfaccia PKCS
- Gestore dei lavori
- Agente MQTT

Integrazione con l'SDK del dispositivo finale

1. Segui le istruzioni in Code generator for Data Model per generare il codice C di basso livello.
2. Segui le istruzioni riportate in Integrazione dell'SDK del dispositivo finale per:

   a. Configura l'ambiente di compilazione

      Crea il codice su Amazon Linux 2023/x86_64 come host di sviluppo. Installa le dipendenze di compilazione necessarie:

      ```
      dnf install make gcc gcc-c++ cmake
      ```

   b. Sviluppa funzioni di callback hardware

      Prima di implementare le funzioni di callback hardware, scopri come funziona l'API. Questo esempio utilizza il On/Off cluster e l' OnOff attributo per controllare una funzione del dispositivo. Per i dettagli sull'API, consultaFunzione C di basso livello APIs.

      ```
      struct DeviceState
      {
        struct iotmiDev_Agent *agent;
        struct iotmiDev_Endpoint *endpointLight;
        /* This simulates the HW state of OnOff */
        bool hwState;
      };

      /* This implementation for OnOff getter just reads
         the state from the DeviceState */
      iotmiDev_DMStatus exampleGetOnOff(bool *value, void *user)
      {
         struct DeviceState *state = (struct DeviceState *)(user);
        *value = state->hwState;
        return iotmiDev_DMStatusOk;
      }
      ```

   c. Configura gli endpoint e aggancia le funzioni di callback hardware

      Dopo aver implementato le funzioni, crea gli endpoint e registra i callback. Completa queste attività:

      i. Crea un agente del dispositivo

ii. Riempi i punti della funzione di callback per ogni struttura del cluster che desideri supportare

iii. Configura gli endpoint e registra i cluster supportati

```
struct DeviceState
{
    struct iotmiDev_Agent * agent;
    struct iotmiDev_Endpoint *endpoint1;

    /* OnOff cluster states*/
    bool hwState;
};


/* This implementation for OnOff getter just reads
   the state from the DeviceState */
iotmiDev_DMStatus exampleGetOnOff( bool * value, void * user )
{
    struct DeviceState * state = ( struct DeviceState * ) ( user );
    *value = state->hwState;
    printf( "%s(): state->hwState: %d\n", __func__, state->hwState );
    return iotmiDev_DMStatusOk;
}

iotmiDev_DMStatus exampleGetOnTime( uint16_t * value, void * user )
{
    *value = 0;
    printf( "%s(): OnTime is %u\n", __func__, *value );
    return iotmiDev_DMStatusOk;
}

iotmiDev_DMStatus exampleGetStartUpOnOff( iotmiDev_OnOff_StartUpOnOffEnum *
 value, void * user )
{
    *value = iotmiDev_OnOff_StartUpOnOffEnum_Off;
    printf( "%s(): StartUpOnOff is %d\n", __func__, *value );
    return iotmiDev_DMStatusOk;
}

void setupOnOff( struct DeviceState *state )
{
    struct iotmiDev_clusterOnOff clusterOnOff = {
```

```
        .getOnOff = exampleGetOnOff,
        .getOnTime = exampleGetOnTime,
        .getStartUpOnOff = exampleGetStartUpOnOff,
    };
    iotmiDev_OnOffRegisterCluster( state->endpoint1,
                                   &clusterOnOff,
                                   ( void * ) state);
}


/* Here is the sample setting up an endpoint 1 with OnOff
   cluster. Note all error handling code is omitted. */
void setupAgent(struct DeviceState *state)
{
    struct iotmiDev_Agent_Config config = {
        .thingId = IOTMI_DEVICE_MANAGED_THING_ID,
        .clientId = IOTMI_DEVICE_CLIENT_ID,
    };
    iotmiDev_Agent_InitDefaultConfig(&config);

    /* Create a device agent before calling other SDK APIs */
    state->agent = iotmiDev_Agent_new(&config);

    /* Create endpoint#1 */
    state->endpoint1 = iotmiDev_Agent_addEndpoint( state->agent,
                                                   1,
                                                   "Data Model Handler Test
 Device",
                                                   (const char*[])
{ "Camera" },
                                                   1 );
    setupOnOff(state);
}
```

Esempio: Build Hub Control

Il controllo dell'hub è fornito come parte del pacchetto Hub SDK. Il sottopacchetto Hub Control è etichettato con `iot-managed-integrations-hub-control-x.x.x` e contiene librerie diverse rispetto all'SDK del dispositivo non modificato.

1. Sposta i file generati dal codice nella cartella: `example`

```
cp codegen/out/* example/dm
```

2. Per creare il controllo dell'hub, esegui i seguenti comandi:

```
cd <hub-control-root-folder>
```

```
mkdir build
```

```
cd build
```

```
cmake -DBUILD_EXAMPLE_WITH_MQTT_PROXY=ON -
DIOTMI_USE_MANAGED_INTEGRATIONS_DEVICE_LOG=ON ..
```

```
cmake -build .
```

3. Esegui gli esempi con il `MQTTProxy` componente sull'hub, con i `MQTTProxy` componenti `HubOnboarding` e in esecuzione.

```
./examples/iotmi_device_sample_camera/iotmi_device_sample_camera
```

Vedi Modello di dati di integrazioni gestite per il modello di dati. Segui il Passaggio 5 Inizia a usare End device SDK per configurare gli endpoint e gestire le comunicazioni tra l'utente finale e. iot-managed-integrations

Esempi supportati

I seguenti esempi sono stati creati e testati:

- iotmi_device_dm_air_purifier_demo
- iotmi_device_basic_diagnostica
- iotmi_device_dm_camera_demo

Piattaforme supportate

La tabella seguente mostra le piattaforme supportate per il controllo dell'hub.

Architettura	Sistema operativo	Versione GCC	Versione Binutils
X86_64	Linux	10.5.0	2,37
aarch64	Linux	10,5,0	2,37

Abilita CloudWatch i registri

L'Hub SDK offre funzionalità di registrazione complete. Per impostazione predefinita, Hub SDK scrive i log nel file system locale. Tuttavia, puoi sfruttare l'API cloud per configurare lo streaming dei log su CloudWatch Logs, che offre:

- Monitora le prestazioni del dispositivo: acquisisci registri di runtime dettagliati per una gestione proattiva dei dispositivi. Abilita l'analisi e il monitoraggio avanzati dei log su tutta la tua flotta di dispositivi
- Risolvi i problemi: genera voci di registro granulari per un'analisi diagnostica rapida. Registra gli eventi a livello di sistema e di applicazione per un'indagine approfondita.
- Registrazione flessibile e centralizzata: gestione remota dei registri senza accesso diretto al dispositivo. Aggrega i log di più dispositivi in un unico archivio ricercabile.

Prerequisiti

- Effettua l'onboard del dispositivo gestito nel cloud. Per informazioni dettagliate, vedi Configurazione dell'hub onboarding.
- Verifica l'avvio e la corretta inizializzazione dell'agente Hub. Per informazioni dettagliate, vedi Installa e convalida le integrazioni gestite Hub SDK.

> **Note**
>
> Per creare configurazioni di registrazione, consulta PutRuntimeLogConfiguration API per i dettagli.

> **Warning**
>
> L'abilitazione dei log conta ai fini della misurazione delle quote a più livelli. L'aumento dei livelli di registro comporterà un aumento del volume di messaggi e costi aggiuntivi.

Configurazioni dei log di Setup Hub SDK

Configurate le impostazioni di registro dell'hub SDK chiamando l'API per configurare la configurazione del log di runtime.

Example esempio di richiesta API

```
aws iot-managed-integrations put-runtime-log-configuration \
    --managed-thing-id MANAGED_THING_ID \
    --runtime-log-configurations LogLevel=DEBUG,UploadLog=TRUE
```

RuntimeLogConfigurations attributi

I seguenti attributi sono opzionali e possono essere configurati nell'`RuntimeLogConfigurations`API.

LogLevel

Imposta il livello di severità minimo per le tracce di runtime. Valori: `DEBUG, ERROR, INFO, WARN`

Predefinito: `WARN` (build rilasciata)

LogFlushLevel

Determina il livello di gravità per il trasferimento immediato dei dati nell'archiviazione locale. Valori: `DEBUG, ERROR, INFO, WARN`

Impostazione predefinita: `DISABLED`

LocalStoreLocation

Specifica la posizione di archiviazione per le tracce di runtime. Impostazione predefinita: `/var/log/awsiotmi`

- Registro attivo: `/var/log/awsiotmi/ManagedIntegrationsDeviceSdkHub.log`

- Registri ruotati: `/var/log/awsiotmi/ManagedIntegrationsDeviceSdkHub.N.log` (N indica l'ordine di rotazione)

LocalStoreFileRotationMaxBytes

Attiva la rotazione del file quando il file corrente supera la dimensione specificata.

> **Important**
>
> Per un'efficienza ottimale, mantieni la dimensione del file inferiore a 125 KB. I valori superiori a 125 KB verranno automaticamente limitati.

LocalStoreFileRotationMaxFiles,

Imposta il numero massimo di file di rotazione consentito dal demone di log.

UploadLog

Controlla il trasferimento della traccia in fase di esecuzione sul cloud. I log vengono archiviati nel gruppo `/aws/iotmanagedintegration` CloudWatch Logs.

Default: `false`.

UploadPeriodMinutes

Definisce la frequenza dei caricamenti delle tracce in fase di esecuzione. Impostazione predefinita: `5`

DeleteLocalStoreAfterUpload

Controlla l'eliminazione dei file dopo il caricamento. Impostazione predefinita: `true`

> **Note**
>
> Se impostato su false, i file caricati vengono rinominati in: `/var/log/awsiotmi/ManagedIntegrationsDeviceSdkHub.uploaded.{uploaded_timestamp}`

File di registro di esempio

Di seguito è riportato un esempio di file CloudWatch di registro:

Tipi di dispositivi Zigbee e Z-Wave supportati

Questa pagina elenca i tipi di dispositivi connessi all'hub che sono stati testati con integrazioni gestite e sono supportati. Le integrazioni gestite supportano entrambi Configurazione semplice (SS) e per questi dispositivi. Configurazione guidata dall'utente (UGS)

Questa tabella elenca i dispositivi Zigbee supportati.

Tipo di dispositivo Zigbee	Funzionalità supportate
Lampadina intelligente/Luce dimmerabile/Luce RGB	OnOff, LevelControl, ColorControl
Presa intelligente	OnOff
Interruttore intelligente	OnOff
Striscia LED	OnOff, LevelControl, ColorControl
Valvola d'acqua	OnOff
Valvola per radiatore	Termostato, timer OnOff
Termostato	Termostato,, Timer FanControl OnOff
Apriporta da garage	WindowCovering, OnOff, LevelControl
Allarme antincendio	BooleanState, OnOff, Timer TemperatureMeasurement, Fumo COAlarm
Sensore di movimento	BooleanState
Sensore di occupazione/presenza umana	BooleanState, OccupancySensing
Sensore per porte e finestre	BooleanState
Sensore di perdite d'acqua	BooleanState

Tipo di dispositivo Zigbee	Funzionalità supportate
Sensore di vibrazioni	BooleanState
Sensore di temperatura e umidità	TemperatureMeasurement, RelativeHumidityMeasurement

Questa tabella elenca i dispositivi Z-Wave supportati.

Tipo di dispositivo Z-Wave	Funzionalità supportate
Lampadina intelligente/Luce dimmerabile	OnOff, LevelControl
Presa intelligente	OnOff
Controller per porte da garage	OnOff, LevelControl
Contatore di energia	ElectricalEnergyMeasurement, ElectricalPowerMeasurement
Batteria	LevelControl
Sirena	LevelControl
Sensore di movimento	BooleanState
Sensore per porte e finestre	BooleanState
Sensore di perdite d'acqua	BooleanState
Sensore di temperatura	TemperatureMeasurement
Sensore di CO	Fumo COAlarm
Sensore di fumo	Fumo COAlarm

Esegui integrazioni gestite su Raspberry Pi

> **Note**
>
> Questa implementazione di AWS IoT Hub SDK su Raspberry Pi è un progetto dimostrativo destinato esclusivamente a scopi di apprendimento e test e non è destinata all'uso in ambienti di produzione. Ai fini di questa demo, imposta le seguenti configurazioni per facilitare lo sviluppo:
> AWS archiviazione delle credenziali: solo a scopo dimostrativo, le credenziali e i certificati vengono archiviati in una posizione accessibile per facilitare i test e lo sviluppo. Gli ambienti di produzione devono utilizzare soluzioni di storage sicure come Gestione dei segreti AWS Systems Manager Parameter Store. Devono implementare la crittografia a riposo e seguire le linee guida AWS IoT di sicurezza.
> Privilegi dei container: la demo viene eseguita con privilegi elevati per consentire l'accesso illimitato alle risorse dell'host e semplificare i flussi di lavoro di sviluppo. In produzione, i container devono funzionare con i privilegi minimi richiesti.
> Configurazione del bridge di rete: la demo utilizza una configurazione di bridge di rete che espone il traffico di rete interno per facilitare il debug e il monitoraggio. Negli ambienti di produzione, implementa l'isolamento e la segmentazione della rete adeguati per impedire l'accesso non autorizzato al traffico di rete interno.
> Autorizzazioni per dispositivi USB: l'accesso illimitato ai dispositivi USB è abilitato per facilitare il collegamento di periferiche di sviluppo e dispositivi di test. Per la produzione, implementa controlli e convalida rigorosi dei dispositivi USB per prevenire attacchi di spoofing dei dispositivi.
> Queste configurazioni consentono test semplici e non devono essere utilizzate negli ambienti di produzione. Durante l'implementazione in produzione, segui le migliori pratiche di sicurezza per evitare la compromissione del sistema host e l'accesso non autorizzato alle credenziali.

Come prerequisito, è necessario configurare il dongle USB Sonoff Zigbee prima di configurare Raspberry Pi.

Firmware flash sul dongle USB Sonoff Zigbee

Prerequisiti

- Dongle USB Sonoff Zigbee

- Windows: installa il driver universale per Windows 0x CP21

Esegui il flashing del firmware

1. Scarica Zigbee Dongle Firmware Build 7.4.1.0.
2. Apri Silabs Firmware Flasher.
3. Collega il dongle USB Sonoff Zigbee al computer.
4. Scorri e trova -E. ZBDongle
5. Scegliere Connetti.
6. Attendi che il dispositivo si connetta.
7. Scegli Cambia firmware.
8. Seleziona Carica il tuo firmware.
9. Trova la posizione del download di Zigbee Dongle Firmware Build 7.4.1.0 e selezionalo.
10. Fare clic su Install (Installa).
11. Attendi l'installazione del firmware.
12. Scegli Continua quando l'installazione è completa.

Il dongle è ora pronto per l'uso.

Scegli tra le opzioni elencate di seguito per eseguire le integrazioni gestite Hub SDK sul tuo Raspberry Pi. Le fasi di configurazione e convalida per entrambi gli approcci sono elencate di seguito.

Argomenti

- Integrazioni gestite Immagine Hub SDK su Raspberry Pi
- Integrazioni gestite Contenitore Docker Hub SDK su Raspberry Pi
- Applicazione demo di integrazioni gestite

Integrazioni gestite Immagine Hub SDK su Raspberry Pi

> **Note**
>
> Questa implementazione di AWS IoT Hub SDK su Raspberry Pi è un progetto dimostrativo destinato esclusivamente a scopi di apprendimento e test e non è destinata all'uso in ambienti di produzione. Ai fini di questa demo, imposta le seguenti configurazioni per facilitare lo sviluppo:
> AWS archiviazione delle credenziali: solo a scopo dimostrativo, le credenziali e i certificati vengono archiviati in una posizione accessibile per facilitare i test e lo sviluppo. Gli ambienti di produzione devono utilizzare soluzioni di storage sicure come Gestione dei segreti AWS Systems Manager Parameter Store. Devono implementare la crittografia a riposo e seguire le linee guida AWS IoT di sicurezza.
> Privilegi dei container: la demo viene eseguita con privilegi elevati per consentire l'accesso illimitato alle risorse dell'host e semplificare i flussi di lavoro di sviluppo. In produzione, i container devono funzionare con i privilegi minimi richiesti.
> Configurazione del bridge di rete: la demo utilizza una configurazione di bridge di rete che espone il traffico di rete interno per facilitare il debug e il monitoraggio. Negli ambienti di produzione, implementa l'isolamento e la segmentazione della rete adeguati per impedire l'accesso non autorizzato al traffico di rete interno.
> Autorizzazioni per dispositivi USB: l'accesso illimitato ai dispositivi USB è abilitato per facilitare il collegamento di periferiche di sviluppo e dispositivi di test. Per la produzione, implementa controlli e convalida rigorosi dei dispositivi USB per prevenire attacchi di spoofing dei dispositivi.
> Queste configurazioni consentono test semplici e non devono essere utilizzate negli ambienti di produzione. Durante l'implementazione in produzione, segui le migliori pratiche di sicurezza per evitare la compromissione del sistema host e l'accesso non autorizzato alle credenziali.

Prerequisiti

Completa questi requisiti prima di distribuire l'immagine Raspberry Pi:

- Scarica e installa Raspberry Pi imager.
- Procurati una scheda SD.
- Configura un Raspberry Pi 5 con CPU quad-core a 64 bit da 2,4 Ghz (8 GB di RAM).
- Collega un dongle USB Sonoff Zigbee.

- Firmware flash sul dongle USB Sonoff Zigbee.
- Connect un dongle SLUSB001A di Silicon Labs.
- Registrati per AWS creare un account.
- Installa la versione più recente di AWS CLI da Managed Integrations AWS CLI Command Reference.

Esegui il flashing di un'immagine Raspberry Pi su una nuova scheda SD

Esegui il flashing dell'immagine delle integrazioni gestite sulla tua scheda SD seguendo questi passaggi:

1. Scarica l'immagine SDK di Raspberry Pi Hub per le integrazioni gestite.
2. Avvia Raspberry Pi Imager sul tuo desktop.
3. Inserisci la scheda SD nel lettore di schede SD integrato del computer o nel lettore di schede USB esterno.
4. Seleziona Scegli dispositivo → Raspberry Pi 5.
5. Seleziona Scegli sistema operativo → Usa personalizzato → Trova il file IotMI-HubSDK-RPi-Image - v1.0.0.img.gz → Apri.
6. Seleziona Scegli spazio di archiviazione → Seleziona il tuo lettore di schede SD.
7. Verifica che la tua configurazione corrisponda alla seguente schermata:

8. Fare clic su Avanti.
9. Configura le impostazioni di personalizzazione del sistema operativo:
   - Nome host: seleziona raspberrypi.
   - Nome utente e password:
     - Abilita Imposta nome utente e password:
     - Per nome utente:, inserisci`hub123456`.
     - Per Password:, inserisci`sh123456`.
   - LAN wireless:
     - Abilita Configura LAN wireless.
     - Inserisci l'SSID e la password del router.

       Impostazioni di esempio:

        - SSID: `iotmi-tplink`
        - Password: `********` (minimo 8 caratteri)
      - Imposta Paese: a`US`.
    - Imposta le impostazioni locali:
      - Imposta fuso orario: a`America/Los Angeles`.
      - Imposta il layout della tastiera: su`US`.
    - SSH:
      - Scegli la scheda servizi.
      - Seleziona Abilita SSH.
      - Scegli Usa l'autenticazione tramite password.
10. Conferma tutti i popup per la personalizzazione del sistema operativo e la cancellazione dei dati.
11. Attendi il completamento del processo di scrittura.
12. Verifica il completamento con successo con la seguente schermata:

13. Fai clic su Continue (Continua).
14. Rimuovi la scheda SD e inseriscila nel tuo Raspberry Pi.

Esegui l'Hub SDK sul Raspberry Pi

Avvia i servizi Hub SDK sul tuo Raspberry Pi configurato:

1. Inserisci la scheda SD preparata nel dispositivo Raspberry Pi 5.
2. Collega il dongle USB Sonoff Zigbee e il dongle Silicon Labs SLUSB001A al Raspberry Pi.
3. Accendi il Raspberry Pi.
4. Assicurati che il Raspberry Pi e il tuo computer (da cui utilizzi SSH) siano sulla stessa rete.
5. Accedi tramite SSH al Raspberry Pi utilizzando le credenziali impostate durante la distribuzione dell'immagine.

   ```
   ssh username@hostname
   ```
6. Vai alla directory dell'hub SDK:

   ```
   cd /data/aws/iotmi
   ```

7. Completa la configurazione di Hub onboarding per aggiungere materiali di autenticazione e configurazione.

> **Note**
>
> Per eseguire questo passaggio devi YUL trovarti DUB nella nostra regione.

8. Esegui l'Hub SDK:

```
cd /data/aws/iotmi
bash start_hub_sdk.sh
```

Il sistema visualizza la seguente risposta per un avvio corretto di Hub SDK:

```
-------Stopping SDK running processes---
-------Starting Hub SDK------------
-------Creating logs directory----------
Logs directory created.
-------Verifying Middleware paths-------
All middleware libraries exist
-------Verifying Middleware pre reqs---
AIPC and KVstroage directories exist
-------Starting HubOnboarding-----------
-------Starting MQTT Proxy-----------
-------Staring Log Daemon---
-------Starting Event Manager-----------
-------Starting Zigbee Service----------
--Checking Zigbee network information--
-------Starting Zwave Service----------
/data/aws/iotmi/middleware/AceZwave/bin /data/aws/iotmi
/data/aws/iotmi
-------Starting CDMB--------------------
-------Starting Agent--------------------
-------Starting Provisioner--------------
-------Checking SDK status---------------
hub1234+    1780  0.2  0.1 1093936 16368 pts/1   Sl+  16:34   0:00 ./iotmi_mqtt_proxy -
C /data/aws/iotmi/config/iotmi_config.json
Process 'iotmi_mqtt_proxy' is running.
hub1234+    1884  0.0  0.0 236272  2624 pts/1    Sl+  16:34   0:00 ./middleware/
AceCommon/bin/ace_eventmgr
Process 'ace_eventmgr' is running.
```

```
hub1234+    1892  9.1  0.1 393040  8352 pts/1    Sl+  16:34   0:04 ./middleware/
AceZigbee/bin/ace_zigbee_service
Process 'ace_zigbee_service' is running.
hub1234+    1923  0.0  0.1 1570736 12736 pts/1   Sl+  16:34   0:00 ./zwave_svc
Process 'zwave_svc' is running.
hub1234+    1958  0.0  0.0 1067632 5776 pts/1    Sl+  16:34   0:00 ./iotmi_cdmb
Process 'iotmi_cdmb' is running.
hub1234+    2001  0.2  0.2 2017712 21264 pts/1   Sl+  16:35   0:00 ./iotmi_device_agent
Process 'iotmi_device_agent' is running.
hub1234+    2045  0.0  0.1 1457824 12624 pts/1   Sl+  16:35   0:00 ./
iotmi_lpw_provisioner
Process 'iotmi_lpw_provisioner' is running.
hub1234+    1813  0.0  0.0 875152  6848 pts/1    Sl+  16:34   0:00 ./iotmi_log_daemon
Process 'iotmi_log_daemon' is running.
------Successfully Started Hub SDK----
```

Passaggi successivi

Dopo aver avviato con successo l'Hub SDK, procedi con l'onboarding e la gestione del dispositivo all'indirizzo. Configurazione guidata dall'utente per l'installazione e il funzionamento dei dispositivi

Integrazioni gestite Contenitore Docker Hub SDK su Raspberry Pi

> **Note**
>
> Questa implementazione di AWS IoT Hub SDK su Raspberry Pi è un progetto dimostrativo destinato esclusivamente a scopi di apprendimento e test e non è destinata all'uso in ambienti di produzione. Ai fini di questa demo, imposta le seguenti configurazioni per facilitare lo sviluppo:
> AWS archiviazione delle credenziali: solo a scopo dimostrativo, le credenziali e i certificati vengono archiviati in una posizione accessibile per facilitare i test e lo sviluppo. Gli ambienti di produzione devono utilizzare soluzioni di storage sicure come Gestione dei segreti AWS Systems Manager Parameter Store. Devono implementare la crittografia a riposo e seguire le linee guida AWS IoT di sicurezza.
> Privilegi dei container: la demo viene eseguita con privilegi elevati per consentire l'accesso illimitato alle risorse dell'host e semplificare i flussi di lavoro di sviluppo. In produzione, i container devono funzionare con i privilegi minimi richiesti.
> Configurazione del bridge di rete: la demo utilizza una configurazione di bridge di rete che espone il traffico di rete interno per facilitare il debug e il monitoraggio. Negli ambienti di

produzione, implementa l'isolamento e la segmentazione della rete adeguati per impedire l'accesso non autorizzato al traffico di rete interno.
Autorizzazioni per dispositivi USB: l'accesso illimitato ai dispositivi USB è abilitato per facilitare il collegamento di periferiche di sviluppo e dispositivi di test. Per la produzione, implementa controlli e convalida rigorosi dei dispositivi USB per prevenire attacchi di spoofing dei dispositivi.
Queste configurazioni consentono test semplici e non devono essere utilizzate negli ambienti di produzione. Durante l'implementazione in produzione, segui le migliori pratiche di sicurezza per evitare la compromissione del sistema host e l'accesso non autorizzato alle credenziali.

Prerequisiti

I seguenti prerequisiti sono richiesti per il contenitore docker.

- Scarica e installa Raspberry Pi imager.
- Procurati una scheda SD.
- Configura un Raspberry Pi 5 con CPU quad-core a 64 bit da 2,4 Ghz (8 GB di RAM).
- Collega un dongle USB Sonoff Zigbee.
- Firmware flash sul dongle USB Sonoff Zigbee.
- Connect un dongle SLUSB001A di Silicon Labs.
- Registrati per AWS creare un account.
- Installa la versione più recente di AWS CLI da Managed Integrations AWS CLI Command Reference.
- Accesso SSH al Raspberry Pi con indirizzo IP o nome host.

Usa il contenitore Docker di Managed Integrations Hub SDK su Raspberry Pi

1. Scarica Integrazioni gestite Raspberry Pi Hub SDK Docker.
2. Copia il file su Raspberry Pi usando SCP:

```
scp ~/path/to/IotMI-HubSDK-Docker-v1.0.0.tar.gz [username]@raspberrypi.local:~
```

3. Connect al Raspberry Pi tramite SSH:

```
ssh hub123456@raspberrypi.local
```

4. Installa Docker se non è presente:

```
# Install Docker
cd
curl -fsSL https://get.docker.com | sudo sh

# Add your user to docker group
sudo usermod -aG docker $USER
exit # exit ssh

# Log in again
```

5. Installa Docker Compose se non è presente:

```
# Install Docker Compose
sudo apt-get update
sudo apt-get install -y docker-compose-plugin
```

6. Estrai i file Hub SDK:

```
# Navigate to the home directory
cd

# Extract the hub-docker.tar.gz file
tar -xzf IotMI-HubSDK-Docker-v1.0.0.tar.gz
```

7. Vai alla directory hub-docker:

```
cd IotMI-HubSDK-Docker
```

8. Completa la configurazione di Hub onboarding per configurare l'autenticazione e le impostazioni.

> **Note**
>
> Per eseguire questo passaggio devi YUL trovarti DUB nella nostra regione.

9. Avvia il contenitore Docker:

```
# The first time it's called, it will build the container
```

```
docker compose up -d
docker compose logs -f
```

Output previsto:

```
[+] Running 1/1
 # Container iotmi-hubsdk-docker-hubsdk-1  Started
hubsdk-1  | -\-\-\-\-\-\-Checking USB dongles-\-\-\-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | -\-\-\-\-\-\-Stopping SDK running processes-\-\-
hubsdk-1  | iotmi_mqtt_proxy: no process found
hubsdk-1  | ace_eventmgr: no process found
hubsdk-1  | ace_zigbee_service: no process found
hubsdk-1  | zwave_svc: no process found
hubsdk-1  | iotmi_cdmb: no process found
hubsdk-1  | iotmi_device_agent: no process found
hubsdk-1  | iotmi_lpw_provisioner: no process found
hubsdk-1  | iotmi_log_daemon: no process found
hubsdk-1  | -\-\-\-\-\-\-Starting Hub SDK-\-\-\-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | -\-\-\-\-\-\-Creating logs directory-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | Logs directory created.
hubsdk-1  | -\-\-\-\-\-\-Verifying Middleware paths-\-\-\-\-\-\-
hubsdk-1  | All middleware libraries exist
hubsdk-1  | -\-\-\-\-\-\-Verifying Middleware pre reqs-\-\-
hubsdk-1  | AIPC and KVstroage directories exist
hubsdk-1  | -\-\-\-\-\-\-Starting HubOnboarding-\-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | -\-\-\-\-\-\-Starting MQTT Proxy-\-\-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | -\-\-\-\-\-\-Staring Log Daemon-\-\-
hubsdk-1  | -\-\-\-\-\-\-Starting Event Manager-\-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | -\-\-\-\-\-\-Starting Zigbee Service-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | -\-\-\-\-\-\-Starting Zwave Service-\-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | /data/aws/iotmi/middleware/AceZwave/bin /data/aws/iotmi
hubsdk-1  | /data/aws/iotmi
hubsdk-1  | -\-\-\-\-\-\-Starting CDMB-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | -\-\-\-\-\-\-Starting Agent-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | -\-\-\-\-\-\-Starting Provisioner-\-\-\-\-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | -\-\-\-\-\-\-Checking SDK status-\-\-\-\-\-\-\-\-\-\-\-\-\-\-
hubsdk-1  | root         105  0.2  0.2 1093488 16608 ?       Sl   20:51   0:00 ./
iotmi_mqtt_proxy -C /data/aws/iotmi/config/iotmi_config.json
hubsdk-1  | Process 'iotmi_mqtt_proxy' is running.
hubsdk-1  | root         183  0.0  0.0 236272  3152 ?        Sl   20:51   0:00 ./
middleware/AceCommon/bin/ace_eventmgr
hubsdk-1  | Process 'ace_eventmgr' is running.
```

```
hubsdk-1  | root         190 12.0  0.1 319264  8352 ?        Sl   20:51   0:04 ./
middleware/AceZigbee/bin/ace_zigbee_service
hubsdk-1  | Process 'ace_zigbee_service' is running.
hubsdk-1  | root         200  0.0  0.1 1365792 12480 ?       Sl   20:51   0:00 ./
zwave_svc
hubsdk-1  | Process 'zwave_svc' is running.
hubsdk-1  | root         233  0.0  0.0 1198704 5760 ?        Sl   20:51   0:00 ./
iotmi_cdmb
hubsdk-1  | Process 'iotmi_cdmb' is running.
hubsdk-1  | root         268  0.2  0.2 2017424 21968 ?       Sl   20:51   0:00 ./
iotmi_device_agent
hubsdk-1  | Process 'iotmi_device_agent' is running.
hubsdk-1  | root         311  0.1  0.1 1523072 13008 ?       Sl   20:51   0:00 ./
iotmi_lpw_provisioner
hubsdk-1  | Process 'iotmi_lpw_provisioner' is running.
hubsdk-1  | root         132  0.0  0.0 875024  7232 ?        Sl   20:51   0:00 ./
iotmi_log_daemon
hubsdk-1  | Process 'iotmi_log_daemon' is running.
hubsdk-1  | -\-\-\-\-\-Successfully Started Hub SDK-\-\-\-
```

Dopo aver avviato con successo l'Hub SDK, procedi con l'onboarding e la gestione del dispositivo all'indirizzo. Configurazione guidata dall'utente per l'installazione e il funzionamento dei dispositivi

Note

- Per accedere alla shell bash del contenitore Docker, esegui il seguente comando:

  ```
  docker compose exec hubsdk bash
  ```

- Per riavviare il contenitore dopo il riavvio, esegui il seguente comando:

  ```
  docker compose up -d
  ```

- Per aggiornare l'Hub SDK, sostituisci i file binari nella seguente cartella:

  ```
  hub-docker/iotmi
  ```

- Per riavviare il contenitore in sicurezza preservando i dati, procedi:

  ```
  docker compose down
  docker compose up -d
  ```

```
docker compose logs -f
```

Applicazione demo di integrazioni gestite

> **Note**
>
> Questa implementazione di AWS IoT Hub SDK su Raspberry Pi è un progetto dimostrativo destinato esclusivamente a scopi di apprendimento e test e non è destinato all'uso in ambienti di produzione. Ai fini di questa demo, imposta le seguenti configurazioni per facilitare lo sviluppo:
> AWS archiviazione delle credenziali: solo a scopo dimostrativo, le credenziali e i certificati vengono archiviati in una posizione accessibile per facilitare i test e lo sviluppo. Gli ambienti di produzione devono utilizzare soluzioni di storage sicure come Gestione dei segreti AWS Systems Manager Parameter Store. Devono implementare la crittografia a riposo e seguire le linee guida AWS IoT di sicurezza.
> Privilegi dei container: la demo viene eseguita con privilegi elevati per consentire l'accesso illimitato alle risorse dell'host e semplificare i flussi di lavoro di sviluppo. In produzione, i container devono funzionare con i privilegi minimi richiesti.
> Configurazione del bridge di rete: la demo utilizza una configurazione di bridge di rete che espone il traffico di rete interno per facilitare il debug e il monitoraggio. Negli ambienti di produzione, implementa l'isolamento e la segmentazione della rete adeguati per impedire l'accesso non autorizzato al traffico di rete interno.
> Autorizzazioni per dispositivi USB: l'accesso illimitato ai dispositivi USB è abilitato per facilitare il collegamento di periferiche di sviluppo e dispositivi di test. Per la produzione, implementa controlli e convalida rigorosi dei dispositivi USB per prevenire attacchi di spoofing dei dispositivi.
> Queste configurazioni consentono test semplici e non devono essere utilizzate negli ambienti di produzione. Durante l'implementazione in produzione, segui le migliori pratiche di sicurezza per evitare la compromissione del sistema host e l'accesso non autorizzato alle credenziali.

L'applicazione demo è un'applicazione demo basata su React che mostra le funzionalità di integrazione gestita per la gestione dei dispositivi domestici intelligenti. Questa applicazione dimostra l'onboarding, il controllo e il monitoraggio dei dispositivi Z-Wave e Zigbee attraverso una moderna interfaccia web.

Prerequisiti

- Registrati per creare un AWS account.
- Crea un armadietto per le credenziali e aggiungi l'armadietto delle credenziali al tuo hub.
- Configurazione completa dell'onboarding dell'Hub.
- Node.js 18+ e npm.
- Installa la versione più recente di AWS CLI dal Managed Integrations AWS CLI Command Reference.
- Browser web moderno (Chrome, Firefox, Safari, Edge)

Installa e configura l'applicazione

1. Scarica l'applicazione demo di integrazioni gestite.
2. Estrai il pacchetto:

```
cd ~/Downloads
tar -xzf IotMI-HubSDK-DemoApp-v1.0.0.tar.gz
cd IotManagedIntegrations-DemoApp
```

3. Installare le dipendenze:

```
npm install
```

4. Crea un `.env` file nella directory principale:

```
# AWS Configuration
REACT_APP_AWS_REGION=your_region
REACT_APP_AWS_ACCESS_KEY_ID=your_access_key
REACT_APP_AWS_SECRET_ACCESS_KEY=your_secret_key
REACT_APP_AWS_SESSION_TOKEN=your_session_token

# IoT Managed Integrations Endpoint
REACT_APP_IOT_ENDPOINT=https://your-iot-endpoint.amazonaws.com

# Hub Configuration
REACT_APP_HUB_MANAGED_THING_ID=your_hub_id
REACT_APP_CREDENTIAL_LOCKER_ID=your_credential_locker_id
```

5. Crea e avvia l'applicazione:

```
npm start
```

6. Accedi all'applicazione all'indirizzo:

```
http://localhost:3000
```

Per informazioni sui prezzi, consulta la sezione Integrazioni gestite nella pagina dei prezzi di AWS IoT Device Management.

Hub di integrazioni gestite fuori bordo

Panoramica del processo offboard di Hub SDK

Il processo di offboarding dell'hub rimuove un hub dal Cloud AWS sistema di gestione. Quando il cloud invia una DeleteManagedThingrichiesta, il processo raggiunge due obiettivi principali:

Azioni lato dispositivo:

- Ripristina lo stato interno dell'hub
- Eliminare tutti i dati salvati localmente
- Prepara il dispositivo per potenziali future reonboarding

Azioni lato cloud:

- Rimuovi tutte le risorse cloud associate all'hub
- Disconnessione completa dall'account precedente

I clienti in genere avviano l'offboarding dall'hub quando:

- Modifica dell'account associato all'hub
- Sostituzione di un hub esistente con un nuovo dispositivo

Il processo garantisce una transizione pulita e sicura tra le configurazioni dell'hub, consentendo una gestione senza interruzioni dei dispositivi e la flessibilità dell'account.

Prerequisiti

- È necessario disporre di un hub integrato. Per istruzioni, consulta Configurazione dell'Hub onboarding.
- Nel `iotmi_config.json` file che si trova in/data/aws/iotmi/config/, verifica che `iot_provisioning_state` sia visualizzato. `PROVISIONED`
- Verifica che i certificati e le chiavi permanenti a cui si fa riferimento in `iotmi_config.json` esistano nei percorsi specificati.
- Assicuratevi che Agent HubOnboarding, Provisioner e il proxy MQTT siano configurati e funzionanti correttamente.
- Verificate che l'hub non abbia dispositivi secondari. Utilizza l' DeleteManagedThingAPI per rimuovere tutti i dispositivi secondari prima di procedere.

Processo offboard di Hub SDK

Segui questi passaggi per uscire dall'hub:

Recupera l'ID hub_managed_thing

Il `iotmi_config.json` file viene utilizzato per memorizzare l'ID dell'oggetto gestito per un hub di integrazioni gestite. Questo identificatore è un'informazione fondamentale che consente all'hub di comunicare con il servizio di integrazioni AWS IoT gestite. L'ID dell'oggetto gestito è memorizzato nella sezione rw (lettura-scrittura) del file JSON, sotto il campo. `managed_thing_id` Questo è visibile nella seguente configurazione di esempio:

```
{
      "ro": {
          "iot_provisioning_method": "FLEET_PROVISIONING",
          "iot_claim_cert_path": "PATH",
          "iot_claim_pk_path": "PATH",
          "UPC": "UPC",
          "sh_endpoint_url": "ENDPOINT_URL",
          "SN": "SN",
          "fp_template_name": "TEMPLATENAME"
      },
      "rw": {
          "iot_provisioning_state": "PROVISIONED",
          "client_id": "ID",
```

```
            "managed_thing_id": "ID",
            "iot_permanent_cert_path": "CERT_PATH",
            "iot_permanent_pk_path": "KEY",
            "metadata": {
                "last_updated_epoch_time": 1747766125
            }
        }
}
```

Invia il comando all'hub esterno

Usa le credenziali del tuo account ed esegui il comando con quanto `managed_thing_id` recuperato nella sezione precedente:

```
aws iot-managed-integrations delete-managed-thing \
    --identifier HUB_MANAGED_THING_ID
```

Verifica che l'hub sia stato disattivato

Usa le credenziali del tuo account ed esegui il comando con i `managed_thing_id` dati recuperati nella sezione precedente:

```
aws iot-managed-integrations get-managed-thing \
    --identifier HUB_MANAGED_THING_ID
```

Scenari di successo e fallimento

Scenario di successo

Se il comando di offboard dell'hub ha avuto esito positivo, è prevista la seguente risposta di esempio:

```
{
      "Message" : "Managed Thing resource not found."
}
```

Inoltre, il seguente esempio `iotmi_config.json` verrebbe osservato se il comando hub offboarding avesse esito positivo. Verificate che la sezione rw contenga solo **`iot_provisioning_state`** e facoltativamente metadati. L'assenza di metadati è accettabile. `iot_provisioning_state`deve essere NOT_PROVISIONED.

```
{
      "ro": {
          "iot_provisioning_method": "FLEET_PROVISIONING",
          "iot_claim_cert_path": "PATH",
          "iot_claim_pk_path": "PATH",
          "UPC": "1234567890101",
          "sh_endpoint_url": "ENDPOINT_URL",
          "SN": "1234567890101",
          "fp_template_name": "test-template"
      },
      "rw": {
          "iot_provisioning_state": "NOT_PROVISIONED",
          "metadata": {
              "last_updated_epoch_time": 1747766125
          }
      }
}
```

Scenario di fallimento

Se il comando di offboard dell'hub non ha avuto esito positivo, è prevista la seguente risposta di esempio:

```
{
      "Arn" : "ARN",
      "CreatedAt" : 1.748968266655E9,
      "Id" : "ID",
      "ProvisioningStatus" : "DELETE_IN_PROGRESS",
      "Role" : "CONTROLLER",
      "SerialNumber" : "SERIAL_NO",
      "Tags" : { },
      "UniversalProductCode" : "UPC",
      "UpdatedAt" : 1.748968272107E9
}
```

- In caso ProvisioningStatusaffermativo`DELETE_IN_PROGRESS`, segui le istruzioni in Hub recovery.
- In caso contrario`DELETE_IN_PROGRESS`, il comando per l'offboard dell'hub non ProvisioningStatusè riuscito nel cloud di Managed Integrations o non è stato ricevuto da Managed Integrations Cloud. Segui le istruzioni in Hub Recovery.
- Se l'offboarding non ha avuto successo, il `iotmi_config.json` file avrà l'aspetto del file di esempio riportato di seguito.

```
{
      "ro": {
          "iot_provisioning_method": "FLEET_PROVISIONING",
          "iot_claim_cert_path": "PATH",
          "iot_claim_pk_path": "PATH",
          "UPC": "123456789101",
          "sh_endpoint_url": "ENDPOINT_URL",
          "SN": "123456789101",
          "fp_template_name": "test-template"
      },
      "rw": {
          "iot_provisioning_state": "PROVISIONED",
          "client_id": "ID",
          "managed_thing_id": "ID",
          "iot_permanent_cert_path": "PATH",
          "iot_permanent_pk_path": "PATH",
          "metadata": {
              "last_updated_epoch_time": 1747766125
          }
      }
}
```

(Facoltativo) Dopo l'offboarding di Hub SDK

> **Important**
>
> I seguenti scenari elencano le azioni opzionali da intraprendere dopo l'offboarding di Hub SDK non riuscito o se desideri riavviare l'hub dopo l'offboarding.

Riimbarco

Se l'offboarding ha avuto successo, esegui l'onboarding del tuo Hub SDK seguendo la Fase 3: Crea un dispositivo gestito (approvvigionamento della flotta) e il resto del processo di bordo.

Ripristino dell'Hub

L'offboarding dell'hub del dispositivo è riuscito e l'offboarding sul cloud non riesce

Se la chiamata GetManagedThingAPI non restituisce il `Managed Thing resource not found` messaggio, ma il file viene offboardato. `iotmi_config.json` Vedi Scenario di successo per un file json di esempio.

Per eseguire il ripristino da questo scenario, consulta Eliminazione forzata.

L'offboarding dell'hub del dispositivo non riesce

Questo scenario si verifica quando l'offboard del file non `iotmi_config.json` viene eseguito correttamente. Vedi Scenario di errore per un file json di esempio.

Per eseguire il ripristino da questo scenario, consulta Eliminazione forzata. Se non `iotmi_config.json` è ancora stato scollegato, è necessario ripristinare le impostazioni di fabbrica dell'hub.

L'offboarding dell'hub del dispositivo e l'offboarding sul cloud non riescono

In questo scenario, non `iotmi_config.json` è ancora stato effettuato l'offboarding e lo stato dell'hub è, o. `ACTIVATED` `DISCOVERED`

Per recuperare da questo scenario, consulta Eliminazione forzata. Se l'eliminazione forzata fallisce o non `iotmi_config.json` viene ancora effettuata l'offboard, è necessario ripristinare le impostazioni di fabbrica dell'hub.

L'hub è offline e lo stato dell'hub è DELETE_IN_PROGRESS

In questo scenario, l'hub è offline e il cloud riceve un comando di offboarding.

Per recuperare da questo scenario, vedi Eliminazione forzata.

Eliminazione forzata

Per eliminare le risorse cloud senza che l'offboarding dell'hub del dispositivo abbia esito positivo, procedi nel seguente modo. Questa operazione può causare incongruenze tra gli stati del cloud e del dispositivo, causando potenzialmente problemi con le operazioni future.

Chiama l' DeleteManagedThing API con l'hub `managed_thing_id` e il parametro force:

```
aws iot-managed-integrations delete-managed-thing \
  --identifier HUB_MANAGED_THING_ID \
  --force
```

Quindi, chiama l' GetManagedThing API e verifica che ritorni`Managed Thing resource not found`. Ciò conferma che le risorse cloud vengono eliminate.

> **Note**
>
> Questo approccio non è consigliato, in quanto può portare a incongruenze tra lo stato del cloud e quello del dispositivo. In genere è meglio assicurarsi che l'offboarding dell'hub del dispositivo avvenga correttamente prima di tentare di eliminare le risorse cloud.

Middleware specifico per il protocollo

> **Important**
>
> La documentazione e il codice forniti qui descrivono un'implementazione di riferimento del middleware. Non viene fornita come parte dell'SDK.

Il middleware specifico del protocollo ha un ruolo fondamentale nell'interazione con gli stack di protocolli sottostanti. Sia i componenti di onboarding dei dispositivi che quelli di controllo dei dispositivi delle integrazioni gestite Hub SDK lo utilizzano per interagire con il dispositivo finale.

Il middleware svolge le seguenti funzioni.

- Estrae gli stack APIs di protocolli del dispositivo di diversi fornitori fornendo un set comune di. APIs
- Fornisce la gestione dell'esecuzione del software come lo scheduler dei thread, la gestione delle code degli eventi e la cache dei dati.

Architettura middleware

Lo schema a blocchi seguente rappresenta l'architettura del middleware Zigbee. Anche l'architettura dei middleware di altri protocolli come Z-Wave è simile.

Il middleware specifico del protocollo è composto da tre componenti principali.

- ACS Zigbee DPK: lo Zigbee Device Porting Kit (DPK) viene utilizzato per fornire l'astrazione dall'hardware e dal sistema operativo sottostanti, garantendo così la portabilità. Fondamentalmente questo può essere considerato come l'hardware abstraction layer (HAL), che fornisce un set comune per controllare e comunicare con le radio Zigbee di diversi fornitori. APIs Il middleware

Zigbee contiene l'implementazione dell'API DPK per il framework Zigbee Application di Silicon Labs.

- Servizio ACS Zigbee: il servizio Zigbee viene eseguito come demone dedicato. Include un gestore API che serve le chiamate API dalle applicazioni client attraverso i canali IPC. AIPC viene utilizzato come canale IPC tra l'adattatore Zigbee e il servizio Zigbee. Fornisce altre funzionalità come la gestione di entrambi async/sync i comandi, la gestione degli eventi dall'HAL e l'utilizzo di ACS Event Manager per la registrazione/pubblicazione degli eventi.
- Adattatore ACS Zigbee: l'adattatore Zigbee è una libreria in esecuzione all'interno del processo applicativo (in questo caso, l'applicazione è il plug-in CDMB). L'adattatore Zigbee ne fornisce un set utilizzato dalle applicazioni client, come CDMB/Provisioner i plugin APIs di protocollo per controllare e comunicare con il dispositivo finale.

End-to-end esempio di flusso di comandi middleware

Ecco un esempio del flusso di comandi attraverso il middleware Zigbee.

Ecco un esempio del flusso di comandi tramite il middleware Z-Wave.

Organizzazione del codice middleware specifica del protocollo

Questa sezione contiene informazioni sulla posizione del codice per ogni componente all'interno del `IotManagedIntegrationsDeviceSDK-Middleware` repository. Di seguito è riportato un esempio della struttura delle cartelle in questo repository.

```
./IotManagedIntegrationsDeviceSDK-Middleware
|— greengrass
|— example-iot-ace-dpk
|— example-iot-ace-general
|— example-iot-ace-project
|— example-iot-ace-z3-gateway
|— example-iot-ace-zware
|— example-iot-ace-zwave-mw
```

Argomenti

- Organizzazione del codice middleware Zigbee

- Organizzazione del codice middleware Z-Wave

Organizzazione del codice middleware Zigbee

Di seguito viene illustrata l'organizzazione del codice middleware di riferimento di Zigbee.

Argomenti

- DPK ACS Zigbee
- Zigbee SDK di Silicon Labs
- Servizio ACS Zigbee
- Adattatore ACS Zigbee

DPK ACS Zigbee

Il codice per Zigbee DPK si trova all'interno della directory elencata nell'esempio seguente:

```
./IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-dpk/example/dpk/ace_hal/
|– common
|–   |– fxnDbusClient
|–   |– include
|– kvs
|– log
|– wifi
|–   |– include
|–   |– src
|–   |– wifid
|–       |– fxnWifiClient
|–       |– include
|– zibgee
|–   |– include
|–   |– src
|–   |– zigbeed
|–       |– ember
|–       |– include
|– zwave
|–   |– include
|–   |– src
|–   |– zwaved
|–       |– fxnZwaveClient
|–       |– include
```

```
|–       |– zware
```

Zigbee SDK di Silicon Labs

L'SDK di Silicon Labs è presentato all'interno della cartella. `IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-z3-gateway` Questo livello ACS Zigbee DPK è implementato per questo SDK di Silicon Labs.

```
./IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-zz3-gateway/
|– autogen
|– config
|– gecko_sdk_4.3.2
|–   |– platform
|–   |– protocol
|–   |– util
```

Servizio ACS Zigbee

Il codice per il servizio Zigbee si trova all'interno della cartella. `IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-general/middleware/zigbee/` Le `include` sottocartelle `src` and in questa posizione contengono tutti i file relativi al servizio ACS Zigbee.

```
IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-general/middleware/zigbee/
src/
|– zb_alloc.c
|– zb_callbacks.c
|– zb_database.c
|– zb_discovery.c
|– zb_log.c
|– zb_main.c
|– zb_region_info.c
|– zb_server.c
|– zb_svc.c
|– zb_svc_pwr.c
|– zb_timer.c
|– zb_util.c
|– zb_zdo.c
|– zb_zts.c
IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-general/middleware/zigbee/
include/
|– init.zigbeeservice.rc
```

```
|– zb_ace_log_uml.h
|– zb_alloc.h
|– zb_callbacks.h
|– zb_client_aipc.h
|– zb_client_event_handler.h
|– zb_database.h
|– zb_discovery.h
|– zb_log.h
|– zb_region_info.h
|– zb_server.h
|– zb_svc.h
|– zb_svc_pwr.h
|– zb_timer.h
|– zb_util.h
|– zb_zdo.h
|– zb_zts.h
```

Adattatore ACS Zigbee

Il codice per l'adattatore ACS Zigbee si trova all'interno della cartella. `IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-general/middleware/zigbee/api` Le `include` sottocartelle `src` and in questa posizione contengono tutti i file relativi alla libreria ACS Zigbee Adaptor.

```
IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-general/middleware/zigbee/
api/src/
|– zb_client_aipc.c
|– zb_client_api.c
|– zb_client_event_handler.c
|– zb_client_zcl.c
IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-general/middleware/zigbee/
api/include/
|– ace
|–   |– zb_adapter.h
|–   |– zb_command.h
|–   |– zb_network.h
|–   |– zb_types.h
|–   |– zb_zcl.h
|–   |– zb_zcl_cmd.h
|–   |– zb_zcl_color_control.h
|–   |– zb_zcl_hvac.h
|–   |– zb_zcl_id.h
|–   |– zb_zcl_identify.h
```

```
|—   |— zb_zcl_level.h
|—   |— zb_zcl_measure_and_sensing.h
|—   |— zb_zcl_onoff.h
|—   |— zb_zcl_power.h
```

Organizzazione del codice middleware Z-Wave

Di seguito viene illustrata l'organizzazione del codice middleware di riferimento Z-wave.

Argomenti

- DPK ACS Z-Wave
- Silicon Labs e Zip Gateway ZWare
- Servizio ACS Z-Wave
- Adattatore ACS Z-Wave

DPK ACS Z-Wave

Il codice per Z-Wave DPK si trova all'interno della cartella. `IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-dpk/example/dpk/ace_hal/zwave`

```
./IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-dpk/example/dpk/ace_hal/
|— common
|—   |— fxnDbusClient
|—   |— include
|— kvs
|— log
|— wifi
|—   |— include
|—   |— src
|—   |— wifid
|—       |— fxnWifiClient
|—       |— include
|— zibgee
|—   |— include
|—   |— src
|—   |— zigbeed
|—       |— ember
|—       |— include
|— zwave
```

```
|—   |— include
|—   |— src
|—   |— zwaved
|—       |— fxnZwaveClient
|—       |— include
|—       |— zware
```

Silicon Labs e Zip Gateway ZWare

Il codice per Silicon labs ZWare e Zip Gateway si trova all'interno della cartella. `IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-z3-gateway` Questo livello ACS Z-Wave DPK è implementato per i gateway Z-Wave C e Zip. APIs

```
./IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-z3-gateway/
|— autogen
|— config
|— gecko_sdk_4.3.2
|—   |— platform
|—   |— protocol
|—   |— util
```

Servizio ACS Z-Wave

Il codice per il servizio Z-Wave si trova all'interno della cartella elencata nella cartella. `IoTmanagedintegrationsMiddlewares/example`iot-ace-zwave-mw/ Le `include` cartelle `src` e presenti in questa posizione contengono tutti i file relativi al servizio ACS Z-Wave.

```
IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-zwave-mw/src/
|— zwave_mgr.c
|— zwave_mgr_cc.c
|— zwave_mgr_ipc_aipc.c
|— zwave_svc.c
|— zwave_svc_dispatcher.c
|— zwave_svc_hsm.c
|— zwave_svc_ipc_aipc.c
|— zwave_svc_main.c
|— zwave_svc_publish.c
IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-zwave-mw/include/
|— ace
|—   |— zwave_common_cc.h
|—   |— zwave_common_cc_battery.h
|—   |— zwave_common_cc_doorlock.h
```

```
|—   |— zwave_common_cc_firmware.h
|—   |— zwave_common_cc_meter.h
|—   |— zwave_common_cc_notification.h
|—   |— zwave_common_cc_sensor.h
|—   |— zwave_common_cc_switch.h
|—   |— zwave_common_cc_thermostat.h
|—   |— zwave_common_cc_version.h
|—   |— zwave_common_types.h
|—   |— zwave_mgr.h
|—   |— zwave_mgr_cc.h
|— zwave_log.h
|— zwave_mgr_internal.h
|— zwave_mgr_ipc.h
|— zwave_svc_hsm.h
|— zwave_svc_internal.h
|— zwave_utils.h
```

Adattatore ACS Z-Wave

Il codice per l'adattatore ACS Zigbee si trova all'interno della cartella. `IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-zwave-mw/cli/` La `include` cartella `src` and in questa posizione contiene tutti i file relativi alla libreria ACS Z-Wave Adaptor.

```
IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-zwave-mw/cli/
|— include
|—   |— zwave_cli.h
|— src
|—   |— zwave_cli.yaml
|—   |— zwave_cli_cc.c
|—   |— zwave_cli_event_monitor.c
|—   |— zwave_cli_main.c
|—   |— zwave_cli_net.c
```

Integra il middleware con SDK

L'integrazione del middleware nel nuovo hub è discussa nelle sezioni seguenti.

Argomenti

- Integrazione dell'API Device Porting Kit (DPK)
- Implementazione di riferimento e organizzazione del codice

Integrazione dell'API Device Porting Kit (DPK)

Per integrare l'SDK di qualsiasi fornitore di chipset con il middleware, viene fornita un'interfaccia API standard dal livello DPK (Device porting kit) del sistema intermedio. I fornitori di servizi di integrazioni gestite ODMs devono implementarli in APIs base all'SDK del fornitore supportato dai chipset Zigbee/ Z-wave/Wi -Fi utilizzati nei loro hub IoT.

Implementazione di riferimento e organizzazione del codice

Ad eccezione del middleware, tutti gli altri componenti Device SDK, come le integrazioni gestite Device Agent e Common Data Model Bridge (CDMB), possono essere utilizzati senza alcuna modifica e devono solo essere compilati in modo incrociato.

L'implementazione del middleware si basa sull'SDK Silicon Labs per Zigbee e Z-Wave. Se i chipset Z-Wave e Zigbee utilizzati nel nuovo hub sono supportati dal Silicon Labs SDK presente nel middleware, il middleware di riferimento può essere utilizzato senza alcuna modifica. È sufficiente compilare in modo incrociato il middleware e può quindi essere eseguito sul nuovo hub.

DPK (Device porting kit) APIs per Zigbee è disponibile in `acehal_zigbee.c` e l'implementazione di riferimento del DPK è presente all'interno della cartella. APIs `zigbee`

```
IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-dpk/example/dpk/ace_hal/
zigbee/
|— CMakeLists.txt
|— include
|—   |— zigbee_log.h
|— src
|—   |— acehal_zigbee.c
|— zigbeed
|—   |— CMakeLists.txt
|—   |— ember
|—   |—   |— ace_ember_common.c
|—   |—   |— ace_ember_ctrl.c
|—   |—   |— ace_ember_hal_callbacks.c
|—   |—   |— ace_ember_network_creator.c
|—   |—   |— ace_ember_power_settings.c
|—   |—   |— ace_ember_zts.c
|—   |— include
|—   |—   |— zbd_api.h
|—   |—   |— zbd_callbacks.h
|—   |—   |— zbd_common.h
|—   |—   |— zbd_network_creator.h
```

```
|—   |—   |— zbd_power_settings.h
|—   |—   |— zbd_zts.h
```

DPK APIs per Z-Wave è disponibile nella cartella `acehal_zwave.c` e l'implementazione di riferimento del DPK è presente all'interno della cartella. APIs `zwaved`

```
IotManagedIntegrationsDeviceSDK-Middleware/example-iot-ace-dpk/example/dpk/ace_hal/
zwave/
|— CMakeLists.txt
|— include
|—   |— zwave_log.h
|— src
|—   |— acehal_zwave.c
|— zwaved
|—   |— CMakeLists.txt
|—   |— fxnZwaveClient
|—   |—   |— zwave_client.c
|—   |—   |— zwave_client.h
|—   |— include
|—   |—   |— zwaved_cc_intf_api.h
|—   |—   |— zwaved_common_utils.h
|—   |—   |— zwaved_ctrl_api.h
|—   |— zware
|—   |—   |— ace_zware_cc_intf.c
|—   |—   |— ace_zware_common_utils.c
|—   |—   |— ace_zware_ctrl.c
|—   |—   |— ace_zware_debug.c
|—   |—   |— ace_zware_debug.h
|—   |—   |— ace_zware_internal.h
```

Come punto di partenza per implementare il livello DPK per un SDK di un fornitore diverso, è possibile utilizzare e modificare l'implementazione di riferimento. Saranno necessarie le seguenti due modifiche per supportare un SDK di un fornitore diverso:

1. Sostituisci l'SDK del fornitore corrente con l'SDK del nuovo fornitore nel repository.
2. Implementa il middleware DPK (Device Porting Kit) in base all'SDK del nuovo fornitore. APIs

Integrazioni gestite SDK per dispositivi finali

Crea una piattaforma IoT che collega i dispositivi intelligenti a integrazioni gestite ed elabora i comandi tramite un'interfaccia di controllo unificata. L'SDK per dispositivi finali si integra con il firmware del dispositivo e fornisce una configurazione semplificata con i componenti edge dell'SDK e una connettività sicura e una gestione dei dispositivi. AWS IoT Core AWS IoT Scarica la versione più recente dell'SDK per il dispositivo finale dal Console di gestione AWS

Questa guida descrive come implementare l'SDK del dispositivo finale nel firmware. Esamina l'architettura, i componenti e i passaggi di integrazione per iniziare a creare la tua implementazione.

Argomenti

Cos'è l'SDK per dispositivi finali?

Cos'è l'SDK del dispositivo finale?

L'End device SDK è una raccolta di codice sorgente, librerie e strumenti forniti da. AWS IoT Progettato per ambienti con risorse limitate, l'SDK supporta dispositivi con un minimo di 512 KB di RAM e 4 MB di memoria flash, come fotocamere e purificatori d'aria che funzionano su Linux integrato e sistemi operativi in tempo reale (RTOS). Scarica la versione più recente dell'SDK per dispositivi finali dalla console di gestione.AWS IoT

Componenti principali

L'SDK combina un agente MQTT per la comunicazione cloud, un gestore dei lavori per la gestione delle attività e un gestore di integrazioni gestite, Data Model Handler. Questi componenti

interagiscono per fornire connettività sicura e traduzione automatica dei dati tra i dispositivi e le integrazioni gestite.

Per i requisiti tecnici dettagliati, consulta ilRiferimento tecnico.

Architettura e componenti SDK del dispositivo finale

Questa sezione descrive l'architettura SDK del dispositivo finale e come i suoi componenti interagiscono con le funzioni C di basso livello. Il diagramma seguente illustra i componenti principali e le loro relazioni nel framework SDK.

Componenti SDK per dispositivi finali

L'architettura End Device SDK contiene i seguenti componenti per le integrazioni gestite, l'integrazione delle funzionalità:

Fornitore

Crea risorse per dispositivi nel cloud delle integrazioni gestite, inclusi certificati dei dispositivi e chiavi private per comunicazioni MQTT sicure. Queste credenziali stabiliscono connessioni affidabili tra il dispositivo e le integrazioni gestite.

Agente MQTT

Gestisce le connessioni MQTT tramite una libreria client C thread-safe. Questo processo in background gestisce le code di comandi in ambienti multithread, con dimensioni di coda configurabili per dispositivi con limiti di memoria. I messaggi vengono instradati attraverso integrazioni gestite per l'elaborazione.

Gestore dei lavori

Aggiornamenti dei processi over-the-air (OTA) per il firmware del dispositivo, le patch di sicurezza e la distribuzione dei file. Questo servizio integrato gestisce gli aggiornamenti software per tutti i dispositivi registrati.

Gestore del modello di dati

Traduce le operazioni tra le integrazioni gestite e le funzioni C di basso livello utilizzando l'implementazione del AWS Matter Data Model. Per ulteriori informazioni, consulta la documentazione di Matter su. GitHub

Chiavi e certificati

Gestisce le operazioni crittografiche tramite l'API PKCS #11, supportando sia i moduli di sicurezza hardware che le implementazioni software come core. PKCS11 Questa API gestisce le operazioni relative ai certificati per componenti come Provisionee e MQTT Agent durante le connessioni TLS.

Provisionario

Il provisionee è un componente delle integrazioni gestite che consente l'approvvigionamento della flotta tramite reclamo. Con il provisionee, effettui il provisionee dei tuoi dispositivi in modo sicuro. L'SDK crea le risorse necessarie per il provisioning dei dispositivi, che includono il certificato del dispositivo e le chiavi private ottenute dal cloud delle integrazioni gestite. Quando desideri effettuare il provisioning dei dispositivi o in caso di modifiche che potrebbero richiedere un nuovo provisioning dei dispositivi, puoi utilizzare il provisionee.

Argomenti

- Flusso di lavoro Provisionee
- Impostazione delle variabili di ambiente
- Registra un endpoint personalizzato
- Crea un profilo di provisioning
- Crea un oggetto gestito
- Fornitura Wi-Fi per utenti SDK
- Approvvigionamento della flotta tramite reclamo
- Funzionalità degli oggetti gestiti

Flusso di lavoro Provisionee

Il processo richiede la configurazione sia sul cloud che sul dispositivo. I clienti configurano i requisiti del cloud come endpoint personalizzati, profili di provisioning e oggetti gestiti. Alla prima accensione del dispositivo, il fornitore:

1. Si connette all'endpoint delle integrazioni gestite utilizzando un certificato di richiesta
2. Convalida i parametri del dispositivo tramite gli hook di provisioning della flotta
3. Ottiene e archivia un certificato permanente e una chiave privata sul dispositivo
4. Il dispositivo utilizza il certificato permanente per riconnettersi

5. Rileva e carica le funzionalità del dispositivo nelle integrazioni gestite

Una volta eseguito correttamente il provisioning, il dispositivo comunica direttamente con le integrazioni gestite. Il provisionee si attiva solo per le attività di riapprovvigionamento.

Impostazione delle variabili di ambiente

Imposta le seguenti AWS credenziali nel tuo ambiente cloud:

```
$ export AWS_ACCESS_KEY_ID=YOUR-ACCOUNT-ACCESS-KEY-ID
$ export AWS_SECRET_ACCESS_KEY=YOUR-ACCOUNT-SECRET-ACCESS-KEY
$ export AWS_DEFAULT_REGION=YOUR-DEFAULT-REGION
```

Registra un endpoint personalizzato

Usa il comando RegisterCustomEndpointAPI nel tuo ambiente cloud per creare un endpoint personalizzato per la device-to-cloud comunicazione.

```
aws iot-managed-integrations register-custom-endpoint
```

Esempio di risposta

```
{ "EndpointAddress":"[ACCOUNT-PREFIX]-ats.iot.AWS-REGION.amazonaws.com" }
```

> **Note**
>
> Memorizza l'indirizzo dell'endpoint per configurare i parametri di provisioning. Utilizza l'`GetCustomEndpoint`API per restituire le informazioni sull'endpoint. Per ulteriori informazioni, consulta GetCustomEndpointAPI e API nella Managed Integrations RegisterCustomEndpointAPI Reference Guide.

Crea un profilo di provisioning

Crea un profilo di approvvigionamento che definisca il metodo di approvvigionamento della tua flotta. Esegui l'CreateProvisioningProfileAPI nel tuo ambiente cloud per restituire un certificato di richiesta e una chiave privata per l'autenticazione del dispositivo:

```
aws iot-managed-integrations create-provisioning-profile \
```

```
--provisioning-type "FLEET_PROVISIONING" \
--name "PROVISIONING-PROFILE-NAME"
```

Esempio di risposta

```
{ "Arn":"arn:aws:iot-managed-integrations:AWS-REGION:YOUR-ACCOUNT-ID:provisioning-
profile/PROFILE_NAME",
    "ClaimCertificate":"string",
    "ClaimCertificatePrivateKey":"string",
    "Name":"ProfileName",
    "ProvisioningType":"FLEET_PROVISIONING"}
```

Puoi implementare la libreria di astrazione della PKCS11 piattaforma principale (PAL) per far funzionare la PKCS11 libreria principale con il tuo dispositivo. Le porte PKCS11 PAL principali devono fornire una posizione in cui archiviare il certificato di richiesta e la chiave privata. Utilizzando questa funzione, puoi archiviare in modo sicuro la chiave privata e il certificato del dispositivo. È possibile archiviare la chiave privata e il certificato su un modulo di sicurezza hardware (HSM) o un modulo di piattaforma affidabile (TPM).

Crea un oggetto gestito

Registra il tuo dispositivo con Managed Integrations Cloud utilizzando l'CreateManagedThingAPI. Includi il numero di serie (SN) e il codice di prodotto universale (UPC) del tuo dispositivo:

```
aws iot-managed-integrations create-managed-thing –role DEVICE \
  --authentication-material-type WIFI_SETUP_QR_BAR_CODE \
  --authentication-material "SN:DEVICE-SN;UPC:DEVICE-UPC;"
```

Di seguito viene visualizzato un esempio di risposta API.

```
{
   "Arn":"arn:aws:iot-managed-integrations:AWS-REGION:ACCOUNT-ID:managed-
thing/59d3c90c55c4491192d841879192d33f",
   "CreatedAt":1.730960226491E9,
   "Id":"59d3c90c55c4491192d841879192d33f"
}
```

L'API restituisce il Managed Thing ID che può essere utilizzato per la convalida del provisioning. Dovrai fornire il numero di serie del dispositivo (SN) e il codice universale del prodotto (UPC), che

vengono abbinati all'oggetto gestito approvato durante la transazione di provisioning. La transazione restituisce un risultato simile al seguente:

```
/**
 * @brief Device info structure.
 */
typedef struct iotmiDev_DeviceInfo
{
    char serialNumber[ IOTMI_DEVICE_MAX_SERIAL_NUMBER_LENGTH + 1U ];
    char universalProductCode[ IOTMI_DEVICE_MAX_UPC_LENGTH + 1U ];
    char internationalArticleNumber[ IOTMI_DEVICE_MAX_EAN_LENGTH + 1U ];
} iotmiDev_DeviceInfo_t;
```

Fornitura Wi-Fi per utenti SDK

I produttori di dispositivi e i fornitori di soluzioni dispongono del proprio servizio di fornitura Wi-Fi proprietario per la ricezione e la configurazione delle credenziali Wi-Fi. Il servizio di fornitura Wi-Fi prevede l'utilizzo di app mobili dedicate, connessioni Bluetooth Low Energy (BLE) e altri protocolli proprietari per trasferire in modo sicuro le credenziali Wi-Fi per il processo di configurazione iniziale.

L'utente dell'SDK del dispositivo finale deve implementare il servizio di fornitura Wi-Fi e il dispositivo può connettersi a una rete Wi-Fi.

Approvvigionamento della flotta tramite reclamo

Utilizzando il provisionee, l'utente finale può fornire un certificato univoco e registrarlo nelle integrazioni gestite utilizzando il provisioning by claim.

L'ID client può essere acquisito dalla risposta del modello di provisioning o dal certificato del dispositivo `<common name>“_”<serial number>`

Funzionalità degli oggetti gestiti

Il provisionee scopre le funzionalità degli oggetti gestiti, quindi le carica nelle integrazioni gestite. Rende disponibili le funzionalità alle app e ad altri servizi a cui accedere. I dispositivi, gli altri client Web e i servizi possono aggiornare le funzionalità utilizzando MQTT e l'argomento MQTT riservato oppure HTTP utilizzando l'API REST.

Over-the-Air aggiornamenti

Panoramica dell'architettura OTA

Il processo di aggiornamento Over-the-Air (OTA) prevede la collaborazione di diversi componenti per fornire aggiornamenti del firmware ai dispositivi. Il diagramma seguente illustra come una richiesta di aggiornamento OTA viene gestita attraverso l'interazione tra End device SDK, Hub SDK e la funzionalità.

L'architettura di aggiornamento OTA è composta dai seguenti componenti:

- Cliente: carica i documenti di lavoro in un bucket S3 e avvia gli aggiornamenti tramite API
- Servizio OTA: gestisce la creazione, la convalida e la gestione dei posti di lavoro
- AWS IoT Jobs: gestisce l'esecuzione e la consegna dei lavori ai dispositivi
- Dispositivi: ricevi e applica gli aggiornamenti tramite Harmony SDK

Prerequisiti

Prima di creare attività OTA, è necessario configurare i seguenti prerequisiti:

Configurazione dell'accesso ad Amazon S3

Per abilitare gli aggiornamenti OTA, devi caricare i documenti di lavoro in un bucket Amazon S3 e configurare le autorizzazioni di accesso appropriate:

1. Carica il tuo documento di lavoro OTA in un bucket S3
2. Aggiungi una bucket policy di Amazon S3 che garantisca alle integrazioni gestite l'accesso ai tuoi documenti di lavoro:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "PolicyForS3JobDocument",
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "iotmanagedintegrations.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": [
        "arn:aws:s3:::YOUR_BUCKET/*",
        "arn:aws:s3:::YOUR_BUCKET/ota_job_document.json",
        "arn:aws:s3:::YOUR_BUCKET"
      ]
    }
  ]
}
```

Over-the-AirImplementa attività (OTA)

È possibile creare attività OTA in due modi, a seconda dei requisiti di aggiornamento e della strategia di targeting per dispositivo:

Aggiornamenti delle attività OTA una tantum

Un'attività OTA una tantum contiene un elenco statico di obiettivi (`ManagedThings`) per eseguire gli aggiornamenti OTA. È possibile aggiungere fino a 100 obiettivi alla volta. Il flusso di lavoro utilizza AWS IoT Jobs with Fleet Indexing mantenendo il livello di astrazione delle integrazioni gestite.

Utilizza l'esempio seguente per creare un'attività OTA una tantum:

```
aws iotmanagedintegrations create-ota-task \
  --description "One-time OTA update" \
  --s3-url "s3://test-job-document-bucket/ota-job-document.json" \
  --protocol HTTP \
  --target ["arn:aws:iotmanagedintegrations:region:account id:managed-thing/managed
 thing id"] \
  --ota-mechanism PUSH \
  --ota-type ONE_TIME \
  --client-token "foo" \
  --tags '{"key1":"foo","key2":"foo"}'
```

Aggiornamenti continui delle attività OTA

Il flusso di lavoro di raggruppamento OTA (Over-the-Air) consente di distribuire aggiornamenti del firmware a gruppi di dispositivi in base a attributi specifici, utilizzando AWS IoT Jobs with Fleet

Indexing mantenendo il livello di astrazione delle integrazioni gestite. Le attività OTA continue utilizzano una stringa di query anziché obiettivi specifici. Tutti i dispositivi che soddisfano i criteri di interrogazione vengono sottoposti ad aggiornamenti OTA e i criteri di interrogazione vengono continuamente rivalutati. Gli obiettivi corrispondenti avranno incarichi di lavoro.

Configura i prerequisiti

Prima di creare attività OTA continue, completa questi prerequisiti:

1. Crea un oggetto gestito chiamando l'CreateManagedThingAPI ed esegui il provisioning della flotta.
2. Aggiungi attributi di metadati ai tuoi oggetti gestiti per il targeting delle query.

Aggiungi attributi e metadati all'`ManagedThing`utilizzo dell'API: UpdateManagedThing

```
aws iotmanagedintegrations update-managed-thing \
  --managed-thing-id "YOUR_MANAGED_THING_ID" \
  --meta-data '{"owner":"managedintegrations","version":"1.0"}'
```

Utilizza l'esempio seguente per creare un'attività OTA continua:

```
aws iotmanagedintegrations create-ota-task \
  --description "Continuous OTA update" \
  --s3-url "s3://test-job-document-bucket/ota-job-document.json" \
  --protocol HTTP \
  --ota-mechanism PUSH \
  --ota-type CONTINUOUS \
  --client-token "foo" \
  --ota-target-query-string "attributes.owner=managedintegrations" \
  --tags '{"key1":"foo","key2":"foo"}'
```

Comprendi il flusso di lavoro OTA continuo

Il flusso di lavoro di aggiornamento continuo OTA segue questi passaggi:

1. Aggiorna gli elementi gestiti con attributi utilizzando l'UpdateManagedThingAPI.
2. Crea un lavoro OTA con una stringa di query mirata agli attributi specifici del dispositivo.
3. Il servizio OTA crea un Thing Group dinamico in AWS IoT Core base agli attributi di query
4. IoT Jobs esegue gli aggiornamenti sui dispositivi corrispondenti

5. Monitora i progressi tramite l'ListOtaTaskExecutionsAPI o le notifiche OTA tramite Kinesis stream (se abilitato).

Differenze tra integrazioni gestite, OTA e IoT Jobs

La distinzione fondamentale tra le integrazioni gestite OTA e IoT Jobs risiede nell'orchestrazione e nell'automazione dei servizi. Integrazioni gestite OTA offre una soluzione a servizio singolo che elimina la complessità del coordinamento multiservizio.

Cosa fa automaticamente OTA per le integrazioni gestite:

- Creazione di gruppi di cose dinamici: genera AWS IoT Core automaticamente gruppi di oggetti in base ai criteri di query.
- Risoluzione del target: traduce le stringhe di query (Esempio:`attributes.owner=managedintegrations`) in destinazioni di dispositivi effettive.
- Integrazione dei servizi: si coordina perfettamente tra i AWS IoT Core servizi IoT Jobs e Fleet Indexing.
- Gestione del ciclo di vita: gestisce l'intero flusso di lavoro OTA dalla creazione al monitoraggio dell'esecuzione.

Cosa elimina MI OTA:

- Creazione di gruppi di cose in AWS IoT Core.
- Aggiungere elementi ai gruppi.
- Creazione di posti di lavoro nell'IoT.

Integrazioni gestite OTA gestisce tutte e tre le operazioni internamente in base alla tua stringa di query, scoprendo automaticamente i dispositivi che corrispondono ai tuoi criteri, creando lavori IoT sotto il cofano e orchestrando l'intero flusso di lavoro OTA senza richiedere l'interazione diretta con più servizi. AWS

Configurazione delle attività OTA

È possibile creare configurazioni per gli aggiornamenti OTA per controllare il modo in cui gli aggiornamenti vengono distribuiti ai dispositivi, impostare le condizioni di interruzione e configurare i timeout.

Esempio: CreateOtaTaskConfiguration

Usa l'esempio seguente per creare una configurazione di attività OTA:

```
aws iotmanagedintegrations create-ota-task-configuration \
  --description "OTA configuration" \
  --name "MyOtaConfig" \
  --push-config '{
    "AbortConfig": {
      "AbortConfigCriteriaList": [
        {
          "Action": "CANCEL",
          "FailureType": "FAILED",
          "MinNumberOfExecutedThings": 1,
          "ThresholdPercentage": 90.0
        }
      ]
    },
    "RolloutConfig": {
      "ExponentialRolloutRate": {
        "BaseRatePerMinute": 1,
        "IncrementFactor": 3.0,
        "RateIncreaseCriteria": {
          "numberOfNotifiedThings": 1
        }
      },
      "MaximumPerMinute": 1
    },
    "TimeoutConfig": {
      "InProgressTimeoutInMinutes": 100
    }
  }' \
  --client-token "foo"
```

Applica le impostazioni di configurazione alle attività OTA

Una volta creata la configurazione, ne riceverai una `taskConfigurationId` che verrà aggiunta alla tua `CreateOtaTask` richiesta insieme a configurazioni aggiuntive:

```
aws iotmanagedintegrations create-ota-task \
  --description "OTA with configuration" \
  --s3-url "s3://test-job-document-bucket/ota-job-document.json" \
```

```
  --protocol HTTP \
  --target ["arn:aws:iotmanagedintegrations:region:account id:managed-thing/managed
 thing id"] \
  --ota-mechanism PUSH \
  --ota-type ONE_TIME \
  --client-token "foo" \
  --task-configuration-id "ae4f49352c5443369f43ad6c3a7f1580" \
  --ota-scheduling-config '{
    "EndBehavior": "STOP_ROLLOUT",
    "EndTime": "2024-10-23T17:00",
    "StartTime": "2024-10-20T17:00"
  }' \
  --ota-task-execution-retry-config '{
    "RetryConfigCriteria": [
      {
        "FailureType": "FAILED",
        "MinNumberOfRetries": 1
      }
    ]
  }' \
  --tags '{"key1":"foo","key2":"foo"}'
```

Monitora le notifiche OTA

È possibile monitorare gli aggiornamenti OTA utilizzando due metodi diversi:

Notifiche push tramite Kinesis Data Streams

Quando le notifiche OTA sono abilitate, gli eventi sullo stato dell'aggiornamento vengono automaticamente inviati allo stream Kinesis. Ciò fornisce una visibilità in tempo reale sullo stato di avanzamento degli aggiornamenti del firmware su tutti i dispositivi.

Monitoraggio con ListOtaTaskExecutions API

Puoi utilizzare l'ListOtaTaskExecutionsAPI per controllare manualmente lo stato degli aggiornamenti OTA per i tuoi oggetti gestiti:

```
aws iotmanagedintegrations list-ota-task-executions \
  --task-id "task-123456789" \
  --max-results 25
```

La risposta fornisce lo stato di esecuzione dettagliato per ogni cosa gestita:

```
{
  "taskExecutionSummaries": [
    {
      "taskExecutionSummary": {
        "executionNumber": 1,
        "lastUpdatedAt": 1634567890,
        "queuedAt": 1634567800,
        "startedAt": 1634567830,
        "status": "SUCCEEDED",
        "retryAttempt": 0
      },
      "managedThingId": "device-001"
    },
    {
      "taskExecutionSummary": {
        "executionNumber": 1,
        "lastUpdatedAt": 1634567920,
        "queuedAt": 1634567800,
        "startedAt": 1634567840,
        "status": "IN_PROGRESS",
        "retryAttempt": 0
      },
      "managedThingId": "device-002"
    }
  ],
  "nextToken": "NEXT_TOKEN"
}
```

Questa API consente di recuperare lo stato di esecuzione dettagliato per ogni oggetto gestito destinato a una specifica attività OTA, inclusi i timestamp e lo stato corrente.

Elabora i documenti di lavoro

Quando si crea un'attività OTA, il gestore dei lavori esegue i seguenti passaggi sul dispositivo. Quando è disponibile un aggiornamento, richiede il documento di lavoro tramite MQTT.

1. Sottoscrive gli argomenti di notifica MQTT.
2. Richiama l'StartNextPendingJobExecutionAPI per i lavori in sospeso.
3. Riceve i documenti di lavoro disponibili.
4. Elabora gli aggiornamenti in base ai timeout specificati.

Utilizzando il gestore dei lavori, l'applicazione può determinare se agire immediatamente o attendere fino a un periodo di timeout specificato.

Implementa l'agente OTA

Quando si riceve il documento di lavoro da Managed Integrations, è necessario disporre di un'implementazione del proprio agente OTA che elabori il documento di lavoro, scarichi gli aggiornamenti ed esegua tutte le operazioni di installazione. L'agente OTA deve eseguire i seguenti passaggi:

1. Analizza i documenti di lavoro per il firmware Amazon URLs S3.
2. Scarica gli aggiornamenti del firmware tramite HTTP.
3. Verifica le firme digitali.
4. Installa aggiornamenti convalidati.
5. Chiama `iotmi\_JobsHandler\_updateJobStatus` con `SUCCESS` o `FAILED` status.

Quando il dispositivo completa con successo l'operazione OTA, deve chiamare l'`iotmi \_JobsHandler\_updateJobStatus`API con uno stato pari `JobSucceeded` a per segnalare un lavoro riuscito.

```
/**
 * @brief Enumeration of possible job statuses.
 */
typedef enum{
    JobQueued,     /** The job is in the queue, waiting to be processed. */
    JobInProgress, /** The job is currently being processed. */
    JobFailed,     /** The job processing failed. */
    JobSucceeded,  /** The job processing succeeded. */
    JobRejected    /** The job was rejected, possibly due to an error or invalid
 request. */
} iotmi_JobCurrentStatus_t;

/**
 * @brief Update the status of a job with optional status details.
 *
 * @param[in] pJobId Pointer to the job ID string.
 * @param[in] jobIdLength Length of the job ID string.
 * @param[in] status The new status of the job.
 * @param[in] statusDetails Pointer to a string containing additional details about the
 job status.
```

```
 *                          This can be a JSON-formatted string or NULL if no details
 are needed.
 * @param[in] statusDetailsLength Length of the status details string. Set to 0 if
 `statusDetails` is NULL.
 *
 * @return 0 on success, non-zero on failure.
 */
int iotmi_JobsHandler_updateJobStatus( const char * pJobId,
                                       size_t jobIdLength,
                                       iotmi_JobCurrentStatus_t status,
                                       const char * statusDetails,
                                       size_t statusDetailsLength );
```

Generatore di codice per modelli di dati

Scopri come utilizzare il generatore di codice per il modello di dati. Il codice generato può essere utilizzato per serializzare e deserializzare i modelli di dati scambiati tra il cloud e il dispositivo.

Il repository del progetto contiene uno strumento di generazione di codice per la creazione di gestori di modelli di dati in codice C. I seguenti argomenti descrivono il generatore di codice e il flusso di lavoro.

Argomenti

- Processo di generazione del codice
- Configurazione dell'ambiente
- Genera codice per dispositivi

Processo di generazione del codice

Il generatore di codice crea file sorgente C da tre input principali: AWS l'implementazione del Matter Data Model (.matter file) dalla piattaforma avanzata Zigbee Cluster Library (ZCL), un plug-in Python che gestisce la preelaborazione e modelli Jinja2 che definiscono la struttura del codice. Durante la generazione, il plugin Python elabora i tuoi file.matter aggiungendo definizioni di tipo globali, organizzando i tipi di dati in base alle loro dipendenze e formattando le informazioni per il rendering dei modelli.

L'immagine seguente descrive il generatore di codice che crea i file sorgente C.

L'SDK del dispositivo finale include plugin Python e modelli Jinja2 compatibili con il progetto. codegen.pyconnectedhomeip Questa combinazione genera più file C per ogni cluster in base all'input del file.matter.

I seguenti argomenti secondari descrivono questi file.

- Plugin Python
- Modelli Jinja2
- (Facoltativo) Schema personalizzato

Plugin Python

Il generatore di codice analizza i file.matter e invia le informazioni come oggetti Python al plugin. `codegen.py` Il file del plugin `iotmi_data_model.py` preelabora questi dati e rende i sorgenti con i modelli forniti. La preelaborazione include:

1. L'aggiunta di informazioni non disponibili da`codegen.py`, ad esempio i tipi globali
2. Esecuzione dell'ordinamento topologico sui tipi di dati per stabilire l'ordine di definizione corretto

> **Note**
>
> L'ordinamento topologico garantisce che i tipi dipendenti siano definiti in base alle rispettive dipendenze, indipendentemente dall'ordine originale.

Modelli Jinja2

L'End Device SDK fornisce modelli Jinja2 personalizzati per gestori di modelli di dati e funzioni C di basso livello.

Modelli Jinja2

Modello	Fonte generata	Remarks
`cluster.h.jinja`	`iotmi_device_<cluster>.h`	Crea file di intestazione di funzioni C di basso livello.
`cluster.c.jinja`	`iotmi_device_<cluster>.c`	Implementa e registra i puntatori delle funzioni di

Modello	Fonte generata	Remarks
		callback con Data Model Handler.
`cluster_type_helpers.h.jinja`	`iotmi_device_type_helpers_<cluster>.h`	Definisce prototipi di funzioni per i tipi di dati.
`cluster_type_helpers.c.jinja`	`iotmi_device_type_helpers_<cluster>.c`	Genera prototipi di funzioni relative ai tipi di dati per enumerazioni, bitmap, elenchi e strutture specifici del cluster.
`iot_device_dm_types.h.jinja`	`iotmi_device_dm_types.h`	Definisce i tipi di dati C per i tipi di dati globali.
`iot_device_type_helpers_global.h.jinja`	`iotmi_device_type_helpers_global.h`	Definisce i tipi di dati C per le operazioni globali.
`iot_device_type_helpers_global.c.jinja`	`iotmi_device_type_helpers_global.c`	Dichiara tipi di dati standard tra cui booleani, numeri interi, virgola mobile, stringhe, bitmap, elenchi e strutture.

(Facoltativo) Schema personalizzato

End Device SDK combina il processo di generazione del codice standardizzato con uno schema personalizzato. Ciò consente l'estensione di Matter Data Model per i dispositivi e il software del dispositivo. Gli schemi personalizzati possono aiutare a descrivere le funzionalità di device-to-cloud comunicazione del dispositivo.

Per informazioni dettagliate sui modelli di dati delle integrazioni gestite, inclusi formato, struttura e requisiti, consulta. Modello di dati di integrazioni gestite

Usa `codegen.py` lo strumento per generare file sorgente C per schemi personalizzati, come segue:

> **Note**
>
> Ogni cluster personalizzato richiede lo stesso ID cluster per i seguenti tre file.

- Crea uno schema personalizzato in un `JSON` formato che fornisca una rappresentazione dei cluster per la creazione di report sulle capacità per creare nuovi cluster personalizzati nel cloud. Un file di esempio si trova in. `codegen/custom_schemas/custom.SimpleLighting@1.0`
- Crea il file di definizione ZCL (Zigbee Cluster Library) in un `XML` formato che contenga le stesse informazioni dello schema personalizzato. Usa lo strumento ZAP per generare i tuoi file Matter IDL da ZCL XML. Un file di esempio si trova in. `codegen/zcl/custom.SimpleLighting.xml`
- L'output dello strumento ZAP è `Matter IDL File (.matter)` e definisce i cluster Matter corrispondenti allo schema personalizzato. Questo è l'input per `codegen.py` lo strumento per generare file sorgente C per End device SDK. Un file di esempio si trova in`codegen/matter_files/custom-light.matter`.

Per istruzioni dettagliate su come integrare modelli di dati di integrazioni gestite personalizzate nel flusso di lavoro di generazione del codice, consultaGenera codice per dispositivi.

Configurazione dell'ambiente

Scopri come configurare il tuo ambiente per utilizzare il generatore di `codegen.py` codice.

Argomenti

Prerequisiti

Installa i seguenti elementi prima di configurare l'ambiente:

- Git
- Python 3.10 o versioni successive
- Poetry 1.2.0 o versione successiva

Configura il tuo ambiente

Utilizzate la procedura seguente per configurare l'ambiente per l'utilizzo del generatore di codice codegen.py.

1. Scarica la versione più recente dell'SDK del dispositivo finale da. Console di gestione AWS
2. Configura l'ambiente Python. Il progetto codegen è basato su Python e utilizza Poetry per la gestione delle dipendenze.
   - Installa le dipendenze del progetto usando poetry nella directory: `codegen`

     ```
     poetry run poetry install --no-root
     ```
3. Configura il tuo repository.
   a. Clona il connectedhomeiprepository. Utilizza lo `codegen.py` script che si trova nella `connectedhomeip/scripts/` cartella per la generazione del codice. Per ulteriori informazioni, vedere connectedhomeip on. GitHub

      ```
      git clone -b v1.4.0.0 https://github.com/project-chip/connectedhomeip.git
      ```
   b. Clonala allo stesso livello della cartella principale. `IoT-managed-integrations-End-Device-SDK` La struttura delle cartelle deve corrispondere alla seguente:

      ```
      |-connectedhomeip
      |-IoT-managed-integrations-End-Device-SDK
      ```

> **Note**
>
> Non è necessario clonare in modo ricorsivo i sottomoduli.

Genera codice per dispositivi

Crea codice C personalizzato per i tuoi dispositivi utilizzando gli strumenti di generazione di codice per le integrazioni gestite. Questa sezione descrive come generare codice da file di esempio inclusi nell'SDK o in base a specifiche personalizzate. Scopri come utilizzare gli script di generazione, comprendere il processo di flusso di lavoro e creare codice che soddisfi i requisiti del tuo dispositivo.

Argomenti

Prerequisiti

1. Python 3.10 o versioni successive.

2. Inizia con un file.matter per la generazione del codice. L'SDK del dispositivo finale fornisce due file di esempio in: `codgen/matter_files folder`

- `custom-air-purifier.matter`
- `aws_camera.matter`

> **Note**
>
> Questi file di esempio generano codice per cluster di applicazioni demo.

Genera codice

Esegui questo comando per generare codice nella cartella out:

```
bash ./gen-data-model-api.sh
```

Genera codice per file.matter personalizzati

Per generare il codice per un `.matter` file specifico o fornire un `.matter` file personalizzato, esegui le seguenti operazioni.

Per generare il codice per i file.matter personalizzati

1. Prepara il tuo file.matter
2. Esegui il comando di generazione:

```
./codegen.sh [--format] configs/dm_basic.json path-to-matter-file output-directory
```

(Facoltativo) Per generare codice con schema personalizzato

1. Prepara lo schema personalizzato in JSON formato
2. Esegui il comando di generazione:

```
./codegen.sh [--format] configs/dm_basic.json path-to-matter-file output-directory
 --custom-schemas-dir path-to-custom-schema-directory
```

I comandi precedenti utilizzano diversi componenti per trasformare il `.matter` file in C codice:

- `codegen.py`dal progetto ConnectedHomeIP
- Plugin Python che si trova in `codegen/py_scripts/iotmi_data_model.py`
- Modelli Jinja2 dalla cartella `codegen/py_scripts/templates`

Il plugin definisce le variabili da passare ai modelli Jinja2, che vengono poi utilizzate per generare l'output finale del codice C. L'aggiunta del `--format` flag applica il formato Clang al codice generato.

Workflow di generazione del codice

Il processo di generazione del codice organizza le strutture di dati dei file.matter utilizzando funzioni di utilità e ordinamento topologico. `topsort.py` Ciò garantisce il corretto ordinamento dei tipi di dati e delle loro dipendenze.

Lo script combina quindi le specifiche del file.matter con l'elaborazione del plugin Python per estrarre e formattare le informazioni necessarie. Infine, applica la formattazione del modello Jinja2 per creare l'output finale del codice C.

Questo flusso di lavoro garantisce che i requisiti specifici del dispositivo contenuti nel file.matter siano tradotti accuratamente in codice C funzionale che si integra con il sistema di integrazioni gestite.

Funzione C di basso livello APIs

Integra il codice specifico del dispositivo con integrazioni gestite utilizzando la funzione C di basso livello fornita. APIs Questa sezione descrive le operazioni API disponibili per ogni cluster nel modello di AWS dati per interazioni efficienti da dispositivo a cloud. Scopri come implementare funzioni di callback, emettere eventi, notificare le modifiche degli attributi e registrare i cluster per gli endpoint dei tuoi dispositivi.

I componenti chiave dell'API includono:

1. Strutture di puntatori delle funzioni di callback per attributi e comandi
2. Funzioni di emissione di eventi
3. Funzioni di notifica della modifica degli attributi
4. Funzioni di registrazione del cluster

Implementandole APIs, crei un ponte tra le operazioni fisiche del dispositivo e le funzionalità cloud delle integrazioni gestite, garantendo comunicazione e controllo senza interruzioni.

La sezione seguente illustra l'API del OnOff cluster.

OnOff API del cluster

Il OnOff.xmlcluster supporta questi attributi e comandi:.

- Attributi:
  - `OnOff (boolean)`
  - `GlobalSceneControl (boolean)`
  - `OnTime (int16u)`
  - `OffWaitTime (int16u)`
  - `StartUpOnOff (StartUpOnOffEnum)`
- Comandi:
  - `Off : () -> Status`
  - `On : () -> Status`
  - `Toggle : () -> Status`
  - `OffWithEffect : (EffectIdentifier: EffectIdentifierEnum, EffectVariant: enum8) -> Status`
  - `OnWithRecallGlobalScene : () -> Status`
  - `OnWithTimedOff : (OnOffControl: OnOffControlBitmap, OnTime: int16u, OffWaitTime: int16u) -> Status`

Per ogni comando, forniamo il puntatore di funzione mappato 1:1 che puoi utilizzare per agganciare l'implementazione.

Tutti i callback per attributi e comandi sono definiti all'interno di una struttura C che prende il nome dal cluster.

Esempio di struttura C

```
struct iotmiDev_clusterOnOff
{
  /*
    - Each attribute has a getter callback if it's readable

    - Each attribute has a setter callback if it's writable

    - The type of `value` are derived according to the data type of
      the attribute.

    - `user` is the pointer passed during an endpoint setup

    - The callback should return iotmiDev_DMStatus to report success or not.

    - For unsupported attributes, just leave them as NULL.
   */
  iotmiDev_DMStatus (*getOnTime)(uint16_t *value, void *user);
  iotmiDev_DMStatus (*setOnTime)(uint16_t value, void *user);
  /*
    - Each command has a command callback

    - If a command takes parameters, the parameters will be defined in a struct
      such as `iotmiDev_OnOff_OnWithTimedOffRequest` below.

    - `user` is the pointer passed during an endpoint setup

    - The callback should return iotmiDev_DMStatus to report success or not.

    - For unsupported commands, just leave them as NULL.
   */
  iotmiDev_DMStatus (*cmdOff)(void *user);
  iotmiDev_DMStatus (*cmdOnWithTimedOff)(const iotmiDev_OnOff_OnWithTimedOffRequest
 *request, void *user);
};
```

Oltre alla struttura C, le funzioni di segnalazione delle modifiche agli attributi sono definite per tutti gli attributi.

```
/* Each attribute has a report function for the customer to report
   an attribute change. An attribute report function is thread-safe.
   */
void iotmiDev_OnOff_OnTime_report_attr(struct iotmiDev_Endpoint *endpoint, uint16_t
 newValue, bool immediate);
```

Le funzioni di segnalazione degli eventi sono definite per tutti gli eventi specifici del cluster. Poiché il OnOff cluster non definisce alcun evento, di seguito è riportato un esempio tratto dal `CameraAvStreamManagement` cluster.

```
/* Each event has a report function for the customer to report
   an event. An event report function is thread-safe.
   The iotmiDev_CameraAvStreamManagement_VideoStreamChangedEvent struct is
   derived from the event definition in the cluster.
 */
void iotmiDev_CameraAvStreamManagement_VideoStreamChanged_report_event(struct
 iotmiDev_Endpoint *endpoint, const
 iotmiDev_CameraAvStreamManagement_VideoStreamChangedEvent *event, bool immediate);
```

Ogni cluster ha anche una funzione di registro.

```
iotmiDev_DMStatus iotmiDev_OnOffRegisterCluster(struct iotmiDev_Endpoint *endpoint,
 const struct iotmiDev_clusterOnOff *cluster, void *user);
```

Il puntatore utente passato alla funzione di registro verrà passato alle funzioni di callback.

Interazioni tra funzionalità e dispositivi nelle integrazioni gestite

Questa sezione descrive il ruolo dell'implementazione della funzione C e l'interazione tra il dispositivo e la funzionalità del dispositivo con integrazioni gestite.

Argomenti

- Gestione dei comandi remoti
- Gestione di eventi non richiesti

Gestione dei comandi remoti

I comandi remoti vengono gestiti dall'interazione tra l'SDK del dispositivo finale e la funzionalità. Le azioni seguenti descrivono un esempio di come accendere una lampadina utilizzando questa interazione.

Il client MQTT riceve il payload e lo passa a Data Model Handler

Quando si invia un comando remoto, il client MQTT riceve il messaggio dalle integrazioni gestite in formato JSON. Quindi passa il payload al gestore del modello di dati. Ad esempio, supponiamo di voler utilizzare integrazioni gestite per accendere una lampadina. La lampadina ha un endpoint #1 che supporta il OnOff cluster. In questo caso, quando invii il comando per accendere la lampadina, Managed Integrations invia una richiesta tramite MQTT al dispositivo, indicando che desidera richiamare il comando On sull'endpoint #1.

Data Model Handler verifica la presenza di funzioni di callback e le richiama

Il Data Model Handler analizza la richiesta JSON. Se la richiesta contiene proprietà o azioni, il Data Model Handler trova gli endpoint e richiama in sequenza le funzioni di callback corrispondenti. Ad esempio, nel caso della lampadina, quando il Data Model Handler riceve il messaggio MQTT, verifica se la funzione di callback corrispondente al comando On definito nel cluster è registrata sull'endpoint #1. OnOff

L'implementazione di Handler e C-Function esegue il comando

Il Data Model Handler richiama le funzioni di callback appropriate che ha trovato e le richiama. L'implementazione della funzione C richiama quindi le funzioni hardware corrispondenti per controllare l'hardware fisico e restituisce il risultato dell'esecuzione. Ad esempio, nel caso della lampadina, il Data Model Handler richiama la funzione di callback e memorizza il risultato dell'esecuzione. Di conseguenza, la funzione di callback accende la lampadina.

Data Model Handler restituisce il risultato dell'esecuzione

Una volta chiamate tutte le funzioni di callback, Data Model Handler combina tutti i risultati. Quindi impacchetta la risposta in formato JSON e pubblica il risultato nel cloud di integrazioni gestite utilizzando il client MQTT. Nel caso della lampadina, il messaggio MQTT nella risposta conterrà il risultato che la lampadina è stata accesa dalla funzione di callback.

Gestione di eventi non richiesti

Gli eventi non richiesti vengono gestiti anche dall'interazione tra l'SDK del dispositivo finale e la funzionalità. Le seguenti azioni descrivono come.

Il dispositivo invia una notifica a Data Model Handler

Quando si verifica una modifica di proprietà o un evento, ad esempio quando viene premuto un pulsante fisico sul dispositivo, l'implementazione di C-Function genera una notifica di evento non richiesto e chiama la funzione di notifica corrispondente per inviare la notifica al Data Model Handler.

Data Model Handler traduce la notifica

Il Data Model Handler gestisce la notifica ricevuta e la traduce nel modello di dati. AWS

Data Model Handler pubblica la notifica nel cloud

Il Data Model Handler pubblica quindi un evento non richiesto nel cloud di integrazioni gestite utilizzando il client MQTT.

Inizia a usare End device SDK

Segui questi passaggi per eseguire l'SDK del dispositivo finale su un dispositivo Linux. Questa sezione guida l'utente attraverso la configurazione dell'ambiente, la configurazione della rete, l'implementazione delle funzioni hardware e la configurazione degli endpoint.

> **Important**
>
> Le applicazioni dimostrative presenti nella `examples` directory e la relativa implementazione PAL (Platform Abstraction Layer) `platform/posix` sono solo di riferimento. Non utilizzarle in ambienti di produzione.

Esamina attentamente ogni passaggio della seguente procedura per garantire la corretta integrazione dei dispositivi con le integrazioni gestite.

Integra l'SDK del dispositivo finale

1. Configurazione dell' EC2 istanza Amazon

   Accedi Console di gestione AWS e avvia un' EC2 istanza Amazon utilizzando un'AMI Amazon Linux. Consulta la sezione Introduzione ad Amazon EC2 nella Guida per l'utente di Amazon Elastic Container Registry.

2. Configura l'ambiente di compilazione

   Crea il codice su Amazon Linux 2023/x86_64 come host di sviluppo. Installa le dipendenze di compilazione necessarie:

   ```
   dnf install make gcc gcc-c++ cmake
   ```

3. (Facoltativo) Configura la rete

   L'SDK del dispositivo finale viene utilizzato al meglio con hardware fisico. Se usi Amazon EC2, non seguire questo passaggio.

   Se non utilizzi Amazon EC2 prima di utilizzare l'applicazione di esempio, inizializza la rete e collega il dispositivo a una rete Wi-Fi disponibile. Completa la configurazione della rete prima del provisioning del dispositivo:

   ```
   /* Provisioning the device PKCS11 with claim credential. */
   status = deviceCredentialProvisioning();
   ```

4. Configurare i parametri di provisioning

   > **Note**
   >
   > Segui Provisionee per ottenere il certificato di richiesta e la chiave privata prima di procedere ulteriormente.

   Modifica il file di configurazione `example/project_name/device_config.sh` con i seguenti parametri di provisioning:

Parametri di provisioning

Parametri macro	Description	Come ottenere queste informazioni
`IOTMI_ROOT_CA_PATH`	Il file di certificato CA principale.	Puoi scaricare questo file dalla sezione Scarica il certificato Amazon Root CA nella guida per gli AWS IoT Core sviluppatori.
`IOTMI_CLAIM_CERTIFICATE_PATH`	Il percorso del file del certificato di attestazione.	Per ottenere il certificato di richiesta e la chiave privata, crea un profilo di approvvigionamento utilizzando l'CreateProvisioningProfileAPI. Per istruzioni, consulta Crea un profilo di provisioning.
`IOTMI_CLAIM_PRIVATE_KEY_PATH`	Il percorso del file della chiave privata del reclamo.	
`IOTMI_MANAGEDINTEGRATIONS_ENDPOINT`	URL dell'endpoint per le integrazioni gestite.	Per ottenere l'endpoint delle integrazioni gestite, utilizza l'API. RegisterCustomEndpoint Per istruzioni, consulta Registra un endpoint personalizzato.
IOTMI_MANAGEDINTEGRATIONS_ENDPOINT_PORT	Il numero di porta per l'endpoint delle integrazioni gestite	Per impostazione predefinita, la porta 8883 viene utilizzata per le operazioni di pubblicazione e sottoscrizione di MQTT. La porta 443 è impostata per l'estensione TLS Application Layer Protocol Negotiation (ALPN) utilizzata dai dispositivi.

5. Crea ed esegui le applicazioni demo

   Questa sezione illustra due applicazioni demo Linux: una semplice telecamera di sicurezza e un purificatore d'aria, entrambe utilizzate CMake come sistema di compilazione.

   a. Semplice applicazione per telecamere di sicurezza

      Per creare ed eseguire l'applicazione, esegui questi comandi:

```
>cd <path-to-code-drop>
# If you didn't generate cluster code earlier
>(cd codegen && poetry run poetry install --no-root && ./gen-data-model-api.sh)
>mkdir build
>cd build
>cmake ..
>cmake –build .
>./examples/iotmi_device_sample_camera/iotmi_device_sample_camera
```

Questa demo implementa funzioni C di basso livello per una telecamera simulata con controller di sessione RTC e cluster di registrazione. Completa il flusso indicato in prima dell'esecuzione. Flusso di lavoro Provisionee

Esempio di output dell'applicazione demo:

```
[2406832727][MAIN][INFO] ======= Device initialization and WIFI provisioning
 =======
[2406832728][MAIN][INFO] fleetProvisioningTemplateName: XXXXXXXXXXX
[2406832728][MAIN][INFO] managedintegrationsEndpoint: XXXXXXXXX.account-prefix-
ats.iot.region.amazonaws.com
[2406832728][MAIN][INFO] pDeviceSerialNumber: XXXXXXXXXXX
[2406832728][MAIN][INFO] universalProductCode: XXXXXXXXXXX
[2406832728][MAIN][INFO] rootCertificatePath: XXXXXXXXX
[2406832728][MAIN][INFO] pClaimCertificatePath: XXXXXXXX
[2406832728][MAIN][INFO] pClaimKeyPath: XXXXXXXXXXXXXXXX
[2406832728][MAIN][INFO] deviceInfo.serialNumber XXXXXXXXXXX
[2406832728][MAIN][INFO] deviceInfo.universalProductCode XXXXXXXXXXXXXX
[2406832728][PKCS11][INFO] PKCS #11 successfully initialized.
[2406832728][MAIN][INFO] ============= Start certificate provisioning
 =============
[2406832728][PKCS11][INFO] ======== Loading Root CA and claim credentials
 through PKCS#11 interface ========
[2406832728][PKCS11][INFO] Writing certificate into label "Root Cert".
[2406832728][PKCS11][INFO] Creating a 0x1 type object.
[2406832728][PKCS11][INFO] Writing certificate into label "Claim Cert".
[2406832728][PKCS11][INFO] Creating a 0x1 type object.
[2406832728][PKCS11][INFO] Creating a 0x3 type object.
[2406832728][MAIN][INFO] ======== Fleet-provisioning-by-Claim ========
[2025-01-02 01:43:11.404995144][iotmi_device_sdkLog][INFO] [2406832728]
[MQTT_AGENT][INFO]
```

```
[2025-01-02 01:43:11.405106991][iotmi_device_sdkLog][INFO] Establishing a TLS
 session to XXXXXXXXXXXXXXX.account-prefix-ats.iot.region.amazonaws.com
[2025-01-02 01:43:11.405119166][iotmi_device_sdkLog][INFO]
[2025-01-02 01:43:11.844812513][iotmi_device_sdkLog][INFO] [2406833168]
[MQTT_AGENT][INFO]
[2025-01-02 01:43:11.844842576][iotmi_device_sdkLog][INFO] TLS session
 connected
[2025-01-02 01:43:11.844852105][iotmi_device_sdkLog][INFO]
[2025-01-02 01:43:12.296421687][iotmi_device_sdkLog][INFO] [2406833620]
[MQTT_AGENT][INFO]
[2025-01-02 01:43:12.296449663][iotmi_device_sdkLog][INFO] Session present: 0.
[2025-01-02 01:43:12.296458997][iotmi_device_sdkLog][INFO]
[2025-01-02 01:43:12.296467793][iotmi_device_sdkLog][INFO] [2406833620]
[MQTT_AGENT][INFO]
[2025-01-02 01:43:12.296476275][iotmi_device_sdkLog][INFO] MQTT connect with
 clean session.
[2025-01-02 01:43:12.296484350][iotmi_device_sdkLog][INFO]
[2025-01-02 01:43:13.171056119][iotmi_device_sdkLog][INFO] [2406834494]
[FLEET_PROVISIONING][INFO]
[2025-01-02 01:43:13.171082442][iotmi_device_sdkLog][INFO] Received accepted
 response from Fleet Provisioning CreateKeysAndCertificate API.
[2025-01-02 01:43:13.171092740][iotmi_device_sdkLog][INFO]
[2025-01-02 01:43:13.171122834][iotmi_device_sdkLog][INFO] [2406834494]
[FLEET_PROVISIONING][INFO]
[2025-01-02 01:43:13.171132400][iotmi_device_sdkLog][INFO] Received privatekey
 and certificate with Id: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
[2025-01-02 01:43:13.171141107][iotmi_device_sdkLog][INFO]
[2406834494][PKCS11][INFO] Creating a 0x3 type object.
[2406834494][PKCS11][INFO] Writing certificate into label "Device Cert".
[2406834494][PKCS11][INFO] Creating a 0x1 type object.
[2025-01-02 01:43:18.584615126][iotmi_device_sdkLog][INFO] [2406839908]
[FLEET_PROVISIONING][INFO]
[2025-01-02 01:43:18.584662031][iotmi_device_sdkLog][INFO] Received accepted
 response from Fleet Provisioning RegisterThing API.
[2025-01-02 01:43:18.584671912][iotmi_device_sdkLog][INFO]
[2025-01-02 01:43:19.100030237][iotmi_device_sdkLog][INFO] [2406840423]
[FLEET_PROVISIONING][INFO]
[2025-01-02 01:43:19.100061720][iotmi_device_sdkLog][INFO] Fleet-provisioning
 iteration 1 is successful.
[2025-01-02 01:43:19.100072401][iotmi_device_sdkLog][INFO]
[2406840423][MQTT][ERROR] MQTT Connection Disconnected Successfully
[2025-01-02 01:43:19.216938181][iotmi_device_sdkLog][INFO] [2406840540]
[MQTT_AGENT][INFO]
```

```
[2025-01-02 01:43:19.216963713][iotmi_device_sdkLog][INFO] MQTT agent thread
 leaves thread loop for iotmiDev_MQTTAgentStop.
[2025-01-02 01:43:19.216973740][iotmi_device_sdkLog][INFO]
[2406840540][MAIN][INFO] iotmiDev_MQTTAgentStop is called to break thread loop
 function.
[2406840540][MAIN][INFO] Successfully provision the device.
[2406840540][MAIN][INFO] Client ID :
 XXXXXXXXXXXXXXXXXXXX_XXXXXXXXXXXXXXXXXXXXXXXX
[2406840540][MAIN][INFO] Managed thing ID : XXXXXXXXXXXXXXXXXXXXXXX
[2406840540][MAIN][INFO] ======================== application loop
 =================
[2025-01-02 01:43:19.217094828][iotmi_device_sdkLog][INFO] [2406840540]
[MQTT_AGENT][INFO]
[2025-01-02 01:43:19.217124600][iotmi_device_sdkLog][INFO] Establishing a TLS
 session to XXXXXXXXX.account-prefix-ats.iot.region.amazonaws.com:8883
[2025-01-02 01:43:19.217138724][iotmi_device_sdkLog][INFO]
[2406840540][Cluster OnOff][INFO] exampleOnOffInitCluster() for endpoint#1
[2406840540][MAIN][INFO] Press Ctrl+C when you finish testing...
[2406840540][Cluster ActivatedCarbonFilterMonitoring][INFO]
 exampleActivatedCarbonFilterMonitoringInitCluster() for endpoint#1
[2406840540][Cluster AirQuality][INFO] exampleAirQualityInitCluster() for
 endpoint#1
[2406840540][Cluster CarbonDioxideConcentrationMeasurement][INFO]
 exampleCarbonDioxideConcentrationMeasurementInitCluster() for endpoint#1
[2406840540][Cluster FanControl][INFO] exampleFanControlInitCluster() for
 endpoint#1
[2406840540][Cluster HepaFilterMonitoring][INFO]
 exampleHepaFilterMonitoringInitCluster() for endpoint#1
[2406840540][Cluster Pm1ConcentrationMeasurement][INFO]
 examplePm1ConcentrationMeasurementInitCluster() for endpoint#1
[2406840540][Cluster Pm25ConcentrationMeasurement][INFO]
 examplePm25ConcentrationMeasurementInitCluster() for endpoint#1
[2406840540][Cluster TotalVolatileOrganicCompoundsConcentrationMeasurement]
[INFO]
 exampleTotalVolatileOrganicCompoundsConcentrationMeasurementInitCluster() for
 endpoint#1
[2025-01-02 01:43:19.648185488][iotmi_device_sdkLog][INFO] [2406840971]
[MQTT_AGENT][INFO]
[2025-01-02 01:43:19.648211988][iotmi_device_sdkLog][INFO] TLS session
 connected
[2025-01-02 01:43:19.648225583][iotmi_device_sdkLog][INFO]

[2025-01-02 01:43:19.938281231][iotmi_device_sdkLog][INFO] [2406841261]
[MQTT_AGENT][INFO]
```

```
[2025-01-02 01:43:19.938304799][iotmi_device_sdkLog][INFO] Session present: 0.
[2025-01-02 01:43:19.938317404][iotmi_device_sdkLog][INFO]
```

b. Semplice applicazione del purificatore d'aria

Per creare ed eseguire l'applicazione, esegui i seguenti comandi:

```
>cd <path-to-code-drop>
# If you didn't generate cluster code earlier
>(cd codegen && poetry run poetry install --no-root && ./gen-data-model-api.sh)
>mkdir build
>cd build
>cmake ..
>cmake --build .
>./examples/iotmi_device_dm_air_purifier/iotmi_device_dm_air_purifier_demo
```

Questa demo implementa funzioni C di basso livello per un purificatore d'aria simulato con 2 endpoint e i seguenti cluster supportati:

Cluster supportati per l'endpoint del purificatore d'aria

Endpoint	Cluster
Endpoint #1: purificatore d'aria	OnOff
	Controllo della ventola
	Monitoraggio del filtro HEPA
	Monitoraggio dei filtri a carbone attivo
Endpoint #2: sensore di qualità dell'aria	Qualità dell'aria
	Misura della concentrazione di anidride carbonica
	Misurazione della concentrazione di formaldeide
	Misurazione della concentrazione di Pm25
	Misura della concentrazione di Pm1

Endpoint	Cluster
	Misurazione della concentrazione totale di composti organici volatili

L'output è simile all'applicazione demo della fotocamera, con diversi cluster supportati.

6. Passaggi successivi:

Le integrazioni gestite, l'SDK per i dispositivi finali e le applicazioni demo sono ora in esecuzione sulle tue istanze Amazon EC2 . Ciò ti consente di sviluppare e testare le tue applicazioni sul tuo hardware fisico. Con questa configurazione, puoi sfruttare il servizio di integrazioni gestite per controllare i tuoi AWS IoT dispositivi.

a. Sviluppa funzioni di callback hardware

Prima di implementare le funzioni di callback hardware, scopri come funziona l'API. Questo esempio utilizza il On/Off cluster e l'OnOffattributo per controllare una funzione del dispositivo. Per i dettagli sull'API, consultaFunzione C di basso livello APIs.

```
struct DeviceState
{
  struct iotmiDev_Agent *agent;
  struct iotmiDev_Endpoint *endpointLight;
  /* This simulates the HW state of OnOff */
  bool hwState;
};

/* This implementation for OnOff getter just reads
   the state from the DeviceState */
iotmiDev_DMStatus exampleGetOnOff(bool *value, void *user)
{
   struct DeviceState *state = (struct DeviceState *)(user);
  *value = state->hwState;
  return iotmiDev_DMStatusOk;
}
```

b. Configura gli endpoint e aggancia le funzioni di callback hardware

Dopo aver implementato le funzioni, crea gli endpoint e registra i callback. Completa le seguenti operazioni:

i. Crea un agente del dispositivo.

   A. Crea un agente del dispositivo utilizzando `iotmiDev_Agent_new()` prima di richiamare qualsiasi altra funzione SDK.

   B. La configurazione deve includere almeno i parametri ThingID e ClientID.

   C. Utilizzate la `iotmiDev_Agent_initDefaultConfig()` funzione per impostare valori predefiniti ragionevoli per parametri quali le dimensioni della coda e il numero massimo di endpoint.

   D. Quando finisci di utilizzare le risorse, liberale con la `iotmiDev_Agent_free()` funzione. In questo modo si evitano perdite di memoria e si garantisce una corretta gestione delle risorse dell'applicazione.

ii. Compila i puntatori delle funzioni di callback per ogni struttura di cluster che desideri supportare.

iii. Configura gli endpoint e registra i cluster supportati.

   Crea endpoint con`iotmiDev_Agent_addEndpoint()`, il che richiede:

   A. Un ID endpoint univoco.

   B. Un nome descrittivo dell'endpoint

   C. Uno o più tipi di dispositivi conformi alle definizioni dei modelli di AWS dati.

   D. Dopo aver creato gli endpoint, registra i cluster utilizzando le funzioni di registrazione specifiche del cluster appropriate.

   E. Ogni registrazione del cluster richiede funzioni di callback per attributi e comandi. Il sistema passa il puntatore del contesto utente ai callback per mantenere lo stato tra le chiamate.

```
struct DeviceState
{
    struct iotmiDev_Agent * agent;
    struct iotmiDev_Endpoint *endpoint1;

    /* OnOff cluster states*/
    bool hwState;
};
```

```
/* This implementation for OnOff getter just reads
   the state from the DeviceState */
iotmiDev_DMStatus exampleGetOnOff( bool * value, void * user )
{
    struct DeviceState * state = ( struct DeviceState * ) ( user );
    *value = state->hwState;
    printf( "%s(): state->hwState: %d\n", __func__, state->hwState );
    return iotmiDev_DMStatusOk;
}

iotmiDev_DMStatus exampleGetOnTime( uint16_t * value, void * user )
{
    *value = 0;
    printf( "%s(): OnTime is %u\n", __func__, *value );
    return iotmiDev_DMStatusOk;
}

iotmiDev_DMStatus exampleGetStartUpOnOff( iotmiDev_OnOff_StartUpOnOffEnum *
 value, void * user )
{
    *value = iotmiDev_OnOff_StartUpOnOffEnum_Off;
    printf( "%s(): StartUpOnOff is %d\n", __func__, *value );
    return iotmiDev_DMStatusOk;
}

void setupOnOff( struct DeviceState *state )
{
    struct iotmiDev_clusterOnOff clusterOnOff = {
        .getOnOff = exampleGetOnOff,
        .getOnTime = exampleGetOnTime,
        .getStartUpOnOff = exampleGetStartUpOnOff,
    };
    iotmiDev_OnOffRegisterCluster( state->endpoint1,
                                   &clusterOnOff,
                                   ( void * ) state);
}


/* Here is the sample setting up an endpoint 1 with OnOff
   cluster. Note all error handling code is omitted. */
void setupAgent(struct DeviceState *state)
{
    struct iotmiDev_Agent_Config config = {
        .thingId = IOTMI_DEVICE_MANAGED_THING_ID,
```

```
        .clientId = IOTMI_DEVICE_CLIENT_ID,
    };
    iotmiDev_Agent_InitDefaultConfig(&config);

    /* Create a device agent before calling other SDK APIs */
    state->agent = iotmiDev_Agent_new(&config);

    /* Create endpoint#1 */
    state->endpoint1 = iotmiDev_Agent_addEndpoint( state->agent,
                                                   1,
                                                   "Data Model Handler Test
 Device",
                                                   (const char*[])
{ "Camera" },
                                                   1 );
    setupOnOff(state);
}
```

c. Utilizzate il gestore dei lavori per ottenere il documento relativo al lavoro

   i. Avvia una chiamata alla tua applicazione OTA:

```
static iotmi_JobCurrentStatus_t processOTA( iotmi_JobData_t * pJobData )
{
    iotmi_JobCurrentStatus_t jobCurrentStatus = JobSucceeded;

...
    // This function should create OTA tasks
    jobCurrentStatus = YOUR_OTA_FUNCTION(iotmi_JobData_t * pJobData);
...

    return jobCurrentStatus;
}
```

   ii. Chiama `iotmi_JobsHandler_start` per inizializzare il gestore dei lavori.

   iii. Chiama `iotmi_JobsHandler_getJobDocument` per recuperare il documento di lavoro dalle integrazioni gestite.

   iv. Quando il Jobs Document viene ottenuto correttamente, scrivi l'operazione OTA personalizzata nella `processOTA` funzione e restituisci uno `JobSucceeded` stato.

```
static void prvJobsHandlerThread( void * pParam )
{
```

```
    JobsHandlerStatus_t status = JobsHandlerSuccess;
    iotmi_JobData_t jobDocument;
    iotmiDev_DeviceRecord_t * pThreadParams = ( iotmiDev_DeviceRecord_t * )
 pParam;
    iotmi_JobsHandler_config_t config = { .pManagedThingID = pThreadParams-
>pManagedThingID, .jobsQueueSize = 10 };

    status = iotmi_JobsHandler_start( &config );

    if( status != JobsHandlerSuccess )
    {
        LogError( ( "Failed to start Jobs Handler." ) );
        return;
    }

    while( !bExit )
    {
        status = iotmi_JobsHandler_getJobDocument( &jobDocument, 30000 );

        switch( status )
        {
            case JobsHandlerSuccess:
            {
                LogInfo( ( "Job document received." ) );
                LogInfo( ( "Job ID: %.*s", ( int ) jobDocument.jobIdLength,
 jobDocument.pJobId ) );
                LogInfo( ( "Job document: %.*s", ( int )
 jobDocument.jobDocumentLength, jobDocument.pJobDocument ) );

                /* Process the job document */
                iotmi_JobCurrentStatus_t jobStatus =
 processOTA( &jobDocument );

                iotmi_JobsHandler_updateJobStatus( jobDocument.pJobId,
 jobDocument.jobIdLength, jobStatus, NULL, 0 );

                iotmiJobsHandler_destroyJobDocument(&jobDocument);

                break;
            }
            case JobsHandlerTimeout:
            {
                LogInfo( ( "No job document available. Polling for job
 document." ) );
```

```
                iotmi_JobsHandler_pollJobDocument();

                break;
            }
            default:
            {
                LogError( ( "Failed to get job document." ) );
                break;
            }
        }
    }

    while( iotmi_JobsHandler_getJobDocument( &jobDocument, 0 ) ==
JobsHandlerSuccess )
    {
        /* Before stopping the Jobs Handler, process all the remaining
jobs. */

        LogInfo( ( "Job document received before stopping." ) );
        LogInfo( ( "Job ID: %.*s", ( int ) jobDocument.jobIdLength,
jobDocument.pJobId ) );
        LogInfo( ( "Job document: %.*s", ( int )
jobDocument.jobDocumentLength, jobDocument.pJobDocument ) );

        storeJobs( &jobDocument );

        iotmiJobsHandler_destroyJobDocument(&jobDocument);
    }

    iotmi_JobsHandler_stop();

    LogInfo( ( "Job handler thread end." ) );

}
```

Porta l'SDK del dispositivo finale sul tuo dispositivo

Esegui il trasferimento dell'SDK del dispositivo finale sulla piattaforma del tuo dispositivo. Segui questi passaggi per connettere i tuoi dispositivi a AWS IoT Device Management.

Scarica e verifica l'SDK del dispositivo finale

1. Scarica l'ultima versione dell'SDK per il dispositivo finale dalla console delle integrazioni gestite.
2. Verifica che la tua piattaforma sia nell'elenco delle piattaforme supportate in. Riferimento: piattaforme supportate

   > **Note**
   >
   > L'SDK del dispositivo finale è stato testato sulle piattaforme specificate. Altre piattaforme potrebbero funzionare, ma non sono state testate.

3. Estrai (decomprimi) i file SDK nel tuo spazio di lavoro.
4. Configura il tuo ambiente di compilazione con le seguenti impostazioni:

   - Percorsi dei file di origine
   - Directory dei file di intestazione
   - Librerie richieste
   - Bandiere del compilatore e del linker
5. Prima di eseguire il porting di Platform Abstraction Layer (PAL), assicurati che le funzionalità di base della piattaforma siano inizializzate. Le funzionalità includono:

   - Attività del sistema operativo
   - Periferiche
   - Interfacce di rete
   - Requisiti specifici della piattaforma

Trasferisci il PAL al tuo dispositivo

1. Crea una nuova directory per le implementazioni specifiche della piattaforma nella directory della piattaforma esistente. Ad esempio, se usi FreerTOS, crea una directory in. `platform/freertos`

   Example Struttura delle cartelle SDK

   ```
   ### <SDK_ROOT_FOLDER>
   #   ### CMakeLists.txt
   ```

```
#   ### LICENSE.txt
#   ### cmake
#   ### commonDependencies
#   ### components
#   ### docs
#   ### examples
#   ### include
#   ### lib
#   ### platform
#   ### test
#   ### tools
```

2. Copia i file di implementazione di riferimento POSIX (.c e .h) dalla cartella posix nella nuova directory della piattaforma. Questi file forniscono un modello per le funzioni che dovrai implementare.

   - Gestione della memoria flash per l'archiviazione delle credenziali
   - Implementazione PKCS #11
   - Interfaccia di trasporto di rete
   - Sincronizzazione oraria
   - Funzioni di riavvio e ripristino del sistema
   - Meccanismi di registrazione
   - Configurazioni specifiche del dispositivo

3. Configura l'autenticazione Transport Layer Security (TLS) con TLS. MBed

   - Utilizza l'implementazione POSIX fornita se disponi già di una versione MBed TLS che corrisponde alla versione SDK sulla tua piattaforma.
   - Con una versione TLS diversa, implementate gli hook di trasporto per il vostro stack TLS con stack. TCP/IP

4. Confronta la configurazione mbedTLS della tua piattaforma con i requisiti SDK in. `platform/posix/mbedtls/mbedtls_config.h` Assicurati che tutte le opzioni richieste siano abilitate.

5. L'SDK si affida a CoreMQTT per interagire con il cloud. Pertanto, è necessario implementare un livello di trasporto di rete che utilizzi la seguente struttura:

```
typedef struct TransportInterface
{
    TransportRecv_t recv;
```

```
    TransportSend_t send;
    NetworkContext_t * pNetworkContext;
} TransportInterface_t;
```

Per ulteriori informazioni, consulta la documentazione dell'interfaccia Transport sul sito Web di FreerTOS.

6. (Facoltativo) L'SDK utilizza l'API PCKS #11 per gestire le operazioni relative ai certificati. CorePKCS #11 è un'implementazione PKCS non specifica per l'hardware per la prototipazione. Ti consigliamo di utilizzare criptoprocessori sicuri come Trusted Platform Module (TPM), Hardware Security Module (HSM) o Secure Element nel tuo ambiente di produzione:

   - Consulta l'implementazione PKCS #11 di esempio che utilizza il file system Linux per la gestione delle credenziali all'indirizzo. `platform/posix/corePKCS11-mbedtls`
   - Implementa il livello PKCS #11 PAL su. `commonDependencies/core_pkcs11/corePKCS11/source/include/core_pkcs11.h`
   - Implementa il file system Linux su`platform/posix/corePKCS11-mbedtls/source/iotmi_pal_Pkcs11Operations.c`.
   - Implementa la funzione store and load del tuo tipo di storage su`platform/include/iotmi_pal_Nvm.h`.
   - Implementa l'accesso standard ai file in`platform/posix/source/iotmi_pal_Nvm.c`.

   Per istruzioni dettagliate sul porting, consulta Porting the core PKCS11 library nella FreerTOS User Guide.

7. Aggiungi le librerie statiche dell'SDK al tuo ambiente di compilazione:

   - Imposta i percorsi delle librerie per risolvere eventuali problemi relativi ai linker o conflitti tra simboli
   - Verifica che tutte le dipendenze siano collegate correttamente

Metti alla prova la tua porta

È possibile utilizzare l'applicazione di esempio esistente per testare la porta. La compilazione deve essere completata senza errori o avvisi.

> **Note**
>
> Ti consigliamo di iniziare con l'applicazione multitasking più semplice possibile. L'applicazione di esempio fornisce un equivalente multitasking.

1. Trovate l'applicazione di esempio in. `examples/[device_type_sample]`
2. Convertite il `main.c` file nel vostro progetto e aggiungete una voce per chiamare la funzione main () esistente.
3. Verificate di poter compilare correttamente l'applicazione demo.

Riferimento tecnico

Argomenti

- Riferimento: piattaforme supportate
- Riferimento: requisiti tecnici
- Riferimento: API comune

Riferimento: piattaforme supportate

La tabella seguente mostra le piattaforme supportate per l'SDK.

Piattaforme supportate

Platform (Piattaforma)	Architecture	Sistema operativo
Linux x86_64	x86_64	Linux
Ambarella	Braccio v8 () AArch64	Linux
AmeBad	Armv8-M a 32 bit	FreeRTOS
ESP32S3	LX7 Xtensa 32 bit	FreeRTOS

Riferimento: requisiti tecnici

La tabella seguente mostra i requisiti tecnici per l'SDK, incluso lo spazio RAM. Lo stesso SDK del dispositivo finale richiede da 5 a 10 MB di spazio ROM quando si utilizza la stessa configurazione.

Spazio RAM

SDK e componenti	Requisiti di spazio (byte utilizzati)
SDK del dispositivo finale stesso	180 KB
Coda di comandi predefinita di MQTT Agent	480 byte (può essere configurato)
Coda in entrata predefinita dell'agente MQTT	320 byte (può essere configurato)

Riferimento: API comune

Questa sezione contiene un elenco di operazioni API non specifiche di un cluster.

```
/* return code for data model related API */
enum iotmiDev_DMStatus
{
  /* The operation succeeded */
  iotmiDev_DMStatusOk = 0,
  /* The operation failed without additional information */
  iotmiDev_DMStatusFail = 1,
  /* The operation has not been implemented yet. */
  iotmiDev_DMStatusNotImplement = 2,
  /* The operation is to create a resource, but the resource already exists. */
  iotmiDev_DMStatusExist = 3,
}

/* The opaque type to represent a instance of device agent. */
struct iotmiDev_Agent;

/* The opaque type to represent an endpoint. */
struct iotmiDev_Endpoint;

/* A device agent should be created before calling other API */
struct iotmiDev_Agent* iotmiDev_create_agent();

/* Destroy the agent and free all occupied resources */
```

```
void iotmiDev_destroy_agent(struct iotmiDev_Agent *agent);

/* Add an endpoint, which starts with empty capabilities */
struct iotmiDev_Endpoint* iotmiDev_addEndpoint(struct iotmiDev_Agent *handle, uint16
 id, const char *name);

/* Test all clusters registered within an endpoint.
   Note: this API might exist only for early drop. */
void iotmiDev_testEndpoint(struct iotmiDev_Endpoint *endpoint);
```

Sicurezza nelle integrazioni gestite per AWS IoT Device Management

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di data center e architetture di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra te e te. AWS Il modello di responsabilità condivisa descrive questo aspetto come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi in Cloud AWS. AWS fornisce inoltre servizi che è possibile utilizzare in modo sicuro. I revisori esterni testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei AWS Programmi di AWS conformità dei Programmi di conformità dei di . Per ulteriori informazioni sui programmi di conformità che si applicano alle integrazioni gestite, consulta AWS Servizi nell'ambito del programma di conformitàAWS .
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. L'utente è anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della propria azienda e le leggi e normative vigenti.

Questa documentazione aiuta a capire come applicare il modello di responsabilità condivisa quando si utilizzano integrazioni gestite. I seguenti argomenti mostrano come configurare le integrazioni gestite per soddisfare gli obiettivi di sicurezza e conformità. Imparerai anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse gestite per le integrazioni.

Argomenti

Protezione dei dati nelle integrazioni gestite

Il modello di responsabilità AWS condivisa modello di di si applica alla protezione dei dati nelle integrazioni gestite per. AWS IoT Device Management 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 con integrazioni gestite AWS IoT Device Management o altro Servizi AWS utilizzando 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.

Crittografia dei dati inattiva per integrazioni gestite

Le integrazioni gestite per AWS IoT Device Management crittografare i dati sensibili dei clienti archiviati per impostazione predefinita utilizzando chiavi di crittografia.

Esistono due tipi di chiavi di crittografia utilizzate per proteggere i dati sensibili per i clienti che utilizzano integrazioni gestite:

Chiavi gestite dal cliente (CMK)

Le integrazioni gestite supportano l'uso di chiavi simmetriche gestite dal cliente che puoi creare, possedere e gestire. L'utente ha il controllo completo su queste chiavi KMS, tra cui la definizione e il mantenimento delle policy chiave, delle policy IAM e delle concessioni, la loro attivazione e disattivazione, la rotazione del materiale crittografico, l'aggiunta di tag, la creazione di alias relativi alle chiavi KMS e la programmazione di chiavi KMS per l'eliminazione.

AWS chiavi di proprietà

Le integrazioni gestite utilizzano queste chiavi per impostazione predefinita per crittografare automaticamente i dati sensibili dei clienti. Non puoi visualizzarne, gestirne o verificarne l'utilizzo. Non è necessario intraprendere alcuna azione o modificare alcun programma per proteggere le chiavi che crittografano i dati. La crittografia predefinita dei dati a riposo aiuta a ridurre il sovraccarico operativo e la complessità associati alla protezione dei dati sensibili. Allo stesso tempo, consente di creare applicazioni sicure che soddisfano i rigorosi requisiti normativi e di conformità alla crittografia.

La chiave di crittografia predefinita utilizzata sono le chiavi AWS possedute. In alternativa, l'API opzionale per aggiornare la chiave di crittografia è `PutDefaultEncryptionConfiguration`.

Per ulteriori informazioni sui tipi di chiavi di AWS KMS crittografia, consulta AWS KMS keys.

AWS KMS utilizzo per integrazioni gestite

Le integrazioni gestite crittografano e decrittografano tutti i dati dei clienti utilizzando la crittografia a busta. Questo tipo di crittografia prenderà i dati in chiaro e li crittograferà con una chiave dati. Successivamente, una chiave di crittografia denominata chiave di wrapping crittograferà la chiave dati originale utilizzata per crittografare i dati in chiaro. Nella crittografia a busta, è possibile utilizzare chiavi di wrapping aggiuntive per crittografare le chiavi di wrapping esistenti che sono più vicine per

gradi di separazione rispetto alla chiave dati originale. Poiché la chiave dati originale è crittografata da una chiave di wrapping memorizzata separatamente, è possibile archiviare la chiave dati originale e i dati crittografati in chiaro nella stessa posizione. Un portachiavi viene utilizzato per generare, crittografare e decrittografare le chiavi dati, oltre alla chiave di wrapping per crittografare e decrittografare la chiave dati.

> **Note**
>
> Il AWS Database Encryption SDK fornisce la crittografia a busta per l'implementazione della crittografia lato client. Per ulteriori informazioni sul AWS Database Encryption SDK, consulta Cos'è il Database Encryption SDK? AWS

Per ulteriori informazioni sulla crittografia delle buste, sulle chiavi dati, sulle chiavi di avvolgimento e sui portachiavi, consulta Envelope encryption, Data key, Wrapping key e Keyrings.

Le integrazioni gestite richiedono che i servizi utilizzino la chiave gestita dal cliente per le seguenti operazioni interne:

- Invia `DescribeKey` richieste per AWS KMS verificare che l'ID della chiave gestita dal cliente simmetrico sia stato fornito durante la rotazione delle chiavi dati.
- Invia `GenerateDataKeyWithoutPlaintext` richieste per AWS KMS generare chiavi dati crittografate dalla chiave gestita dal cliente.
- Invia `ReEncrypt*` richieste per AWS KMS crittografare nuovamente le chiavi di dati utilizzando la chiave gestita dal cliente.
- Invia `Decrypt` richieste per AWS KMS decrittografare i dati utilizzando la chiave gestita dal cliente.

Tipi di dati crittografati tramite chiavi di crittografia

Le integrazioni gestite utilizzano chiavi di crittografia per crittografare più tipi di dati archiviati su disco. L'elenco seguente descrive i tipi di dati crittografati a riposo utilizzando chiavi di crittografia:

- Eventi del connettore Cloud--to-Cloud (C2C) come il rilevamento e l'aggiornamento dello stato del dispositivo.
- Creazione di un oggetto gestito che rappresenta il dispositivo fisico e di un profilo del dispositivo contenente le funzionalità per un tipo di dispositivo specifico. Per ulteriori informazioni su un dispositivo e sul profilo del dispositivo, vedere Dispositivo eDispositivo.

- Notifiche di integrazioni gestite su vari aspetti dell'implementazione del dispositivo. Per ulteriori informazioni sulle notifiche di integrazioni gestite, consulta. Configura le notifiche relative alle integrazioni gestite
- Informazioni di identificazione personale (PII) di un utente finale, come materiale di autenticazione del dispositivo, numero di serie del dispositivo, nome dell'utente finale, identificatore del dispositivo e Amazon Resource Name (arn) del dispositivo.

In che modo le integrazioni gestite utilizzano le politiche chiave in AWS KMS

Per la rotazione delle chiavi delle filiali e le chiamate asincrone, le integrazioni gestite richiedono una policy chiave per l'utilizzo della chiave di crittografia. Una politica chiave viene utilizzata per i seguenti motivi:

- Autorizza a livello di codice l'uso di una chiave di crittografia per altri principali. AWS

Per un esempio di policy chiave utilizzata per gestire l'accesso alla chiave di crittografia nelle integrazioni gestite, vedi Crea una chiave di crittografia

> **Note**
>
> Per una chiave AWS di proprietà, non è richiesta una policy chiave in quanto la chiave AWS proprietaria è di proprietà di AWS e non è possibile visualizzarla, gestirla o utilizzarla. Le integrazioni gestite utilizzano per impostazione predefinita la chiave AWS proprietaria per crittografare automaticamente i dati sensibili dei clienti.

Oltre a utilizzare le policy chiave per la gestione della configurazione di crittografia con AWS KMS chiavi, le integrazioni gestite utilizzano le politiche IAM. Per ulteriori informazioni sulle politiche IAM, consulta Politiche e autorizzazioni in. AWS Identity and Access Management

Crea una chiave di crittografia

È possibile creare una chiave di crittografia utilizzando Console di gestione AWS o il AWS KMS APIs.

Per creare una chiave di crittografia

Segui i passaggi per la creazione di una chiave KMS nella Guida per gli AWS Key Management Service sviluppatori.

Policy della chiave

Una dichiarazione politica chiave controlla l'accesso a una AWS KMS chiave. Ogni AWS KMS chiave conterrà solo una politica chiave. Questa politica chiave determina quali AWS presidi possono utilizzare la chiave e come possono usarla. Per ulteriori informazioni sulla gestione dell'accesso e dell'utilizzo delle AWS KMS chiavi utilizzando le dichiarazioni politiche chiave, vedere Gestione dell'accesso tramite le politiche.

Di seguito è riportato un esempio di dichiarazione politica chiave che puoi utilizzare per gestire l'accesso e l'utilizzo delle AWS KMS chiavi archiviate nelle tue Account AWS integrazioni gestite:

```
{
   "Statement" : [
    {
      "Sid" : "Allow access to principals authorized to use managed integrations",
      "Effect" : "Allow",
      "Principal" : {
        //Note: Both role and user are acceptable.
        "AWS": "arn:aws:iam::111122223333:user/username",
        "AWS": "arn:aws:iam::111122223333:role/roleName"
      },
      "Action" : [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt",
        "kms:ReEncrypt*"
      ],
      "Resource" : "arn:aws:kms:region:111122223333:key/key_ID",
      "Condition" : {
        "StringEquals" : {
          "kms:ViaService" : "iotmanagedintegrations.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContext:aws-crypto-ec:iotmanagedintegrations": "111122223333"
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:iotmanagedintegrations:<region>:<accountId>:managed-thing/
<managedThingId>",
            "arn:aws:iotmanagedintegrations:<region>:<accountId>:credential-locker/
<credentialLockerId>",
            "arn:aws:iotmanagedintegrations:<region>:<accountId>:provisioning-profile/
<provisioningProfileId>",
            "arn:aws:iotmanagedintegrations:<region>:<accountId>:ota-task/<otaTaskId>"
```

```
          ]
        }
      }
    },
    {
      "Sid" : "Allow access to principals authorized to use managed integrations for
 async flow",
      "Effect" : "Allow",
      "Principal" : {
        "Service": "iotmanagedintegrations.amazonaws.com"
      },
      "Action" : [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt",
        "kms:ReEncrypt*"
      ],
      "Resource" : "arn:aws:kms:region:111122223333:key/key_ID",
      "Condition" : {
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContext:aws-crypto-ec:iotmanagedintegrations": "111122223333"
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:iotmanagedintegrations:<region>:<accountId>:managed-thing/
<managedThingId>",
            "arn:aws:iotmanagedintegrations:<region>:<accountId>:credential-locker/
<credentialLockerId>",
            "arn:aws:iotmanagedintegrations:<region>:<accountId>:provisioning-profile/
<provisioningProfileId>",
            "arn:aws:iotmanagedintegrations:<region>:<accountId>:ota-task/<otaTaskId>"
          ]
        }
      }
    },
    {
      "Sid" : "Allow access to principals authorized to use managed integrations for
 describe key",
      "Effect" : "Allow",
      "Principal" : {
        "AWS": "arn:aws:iam::111122223333:user/username"
      },
      "Action" : [
        "kms:DescribeKey",
      ],
```

```
      "Resource" : "arn:aws:kms:region:111122223333:key/key_ID",
      "Condition" : {
        "StringEquals" : {
          "kms:ViaService" : "iotmanagedintegrations.amazonaws.com"
        }
      }
    },
    {
      "Sid": "Allow access for key administrators",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
       },
      "Action" : [
        "kms:*"
       ],
      "Resource": "*"
    }
  ]
}
```

Per ulteriori informazioni sui key store, consulta Key stores.

Aggiornamento della configurazione di crittografia

La capacità di aggiornare senza problemi la configurazione di crittografia è fondamentale per gestire l'implementazione della crittografia dei dati per le integrazioni gestite. Quando inizierai a utilizzare le integrazioni gestite, ti verrà richiesto di selezionare la configurazione di crittografia. Le tue opzioni saranno le chiavi di AWS proprietà predefinite o la creazione di una chiave personalizzata. AWS KMS

Console di gestione AWS

Per aggiornare la configurazione di crittografia in Console di gestione AWS, apri la home page del AWS IoT servizio, quindi vai a Managed Integration for Unified Control > Impostazioni > Encryption. Nella finestra delle impostazioni di crittografia, puoi aggiornare la configurazione di crittografia selezionando una nuova AWS KMS chiave per una protezione di crittografia aggiuntiva. Scegli Personalizza le impostazioni di crittografia (avanzate) per selezionare una AWS KMS chiave esistente oppure puoi scegliere Crea una AWS KMS chiave per creare la tua chiave gestita dal cliente.

Comandi API

Esistono due modi APIs per gestire la configurazione di crittografia delle AWS KMS chiavi nelle integrazioni gestite: `PutDefaultEncryptionConfiuration` e`GetDefaultEncryptionConfiguration`.

Per aggiornare la configurazione di crittografia predefinita, chiama`PutDefaultEncryptionConfiuration`. Per ulteriori informazioni su `PutDefaultEncryptionConfiuration`, consulta PutDefaultEncryptionConfiuration.

Per visualizzare la configurazione di crittografia predefinita, chiama`GetDefaultEncryptionConfiguration`. Per ulteriori informazioni su `GetDefaultEncryptionConfiguration`, consulta GetDefaultEncryptionConfiguration.

Gestione delle identità e degli accessi per integrazioni gestite

AWS Identity and Access Management (IAM) è uno strumento Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle risorse. AWS Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse di integrazione gestite. IAM è uno strumento Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell’accesso tramite policy
- AWS politiche gestite per integrazioni gestite
- Come funzionano le integrazioni gestite con IAM
- Esempi di policy basate sull'identità per integrazioni gestite
- Risoluzione dei problemi relativi alle integrazioni gestite, all'identità e all'accesso
- Utilizzo di ruoli collegati ai servizi per integrazioni gestite

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 relativi alle integrazioni gestite, all'identità e all'accesso)
- Amministratore del servizio: determina l'accesso degli utenti e invia le richieste di autorizzazione (consulta Come funzionano le integrazioni gestite con IAM)
- Amministratore IAM: scrivi policy per gestire l'accesso (consulta Esempi di policy basate sull'identità per integrazioni gestite)

Autenticazione con identità

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 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 dell'utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Identità federata

Come procedura ottimale, richiedi agli utenti umani di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory aziendale, del provider di identità Web o Directory Service che accede Servizi AWS utilizzando le credenziali di una fonte di identità. Le identità federate assumono ruoli che forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, si consiglia di utilizzare AWS IAM Identity Center. Per ulteriori informazioni, consulta Che cos'è il Centro identità IAM? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. 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 federato degli utenti, le autorizzazioni utente IAM temporanee, l'accesso tra account, l'accesso tra servizi 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 collegandole a identità o risorse. AWS 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.

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.

AWS politiche gestite per integrazioni gestite

Per aggiungere autorizzazioni a utenti, gruppi e ruoli, è più facile utilizzare le policy AWS gestite che scriverle 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 lancia 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.

AWS politica gestita: AWSIo TManaged IntegrationsFullAccess

È possibile allegare la policy `AWSIoTManagedIntegrationsFullAccess` alle identità IAM.

Questa politica concede le autorizzazioni di accesso complete alle integrazioni gestite e ai servizi correlati. Per visualizzare questa politica nel, vedere. Console di gestione AWSAWSIoTManagedIntegrationsFullAccess

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `iotmanagedintegrations`— Fornisce l'accesso completo alle integrazioni gestite e ai servizi correlati per gli utenti, i gruppi e i ruoli IAM a cui aggiungi questa policy.
- `iam`— Consente agli utenti, ai gruppi e ai ruoli IAM assegnati di creare un ruolo collegato al servizio in un. Account AWS

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iotmanagedintegrations:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/
iotmanagedintegrations.amazonaws.com/AWSServiceRoleForIoTManagedIntegrations",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "iotmanagedintegrations.amazonaws.com"
        }
      }
    }
```

```
  ]
}
```

AWS politica gestita: Io AWS TManaged IntegrationsRolePolicy

È possibile allegare la policy `AWS IoTManagedIntegrationsRolePolicy` alle identità IAM.

Questa politica concede alle integrazioni gestite l'autorizzazione a pubblicare CloudWatch log e metriche di Amazon per tuo conto.

Per visualizzare questa politica nel, consulta. Console di gestione AWSAWSIoTManagedIntegrationsRolePolicy

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni:

- `logs`— Fornisce la possibilità di creare gruppi di CloudWatch log Amazon e di trasmettere log ai gruppi.
- `cloudwatch`— Fornisce la possibilità di pubblicare i CloudWatch parametri di Amazon. Per ulteriori informazioni sui CloudWatch parametri di Amazon, consulta Metrics in Amazon. CloudWatch

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/iotmanagedintegrations/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalAccount": "${aws:ResourceAccount}"
```

```
        }
      }
    },
    {
      "Sid": "CloudWatchStreams",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/iotmanagedintegrations/*:log-stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalAccount": "${aws:ResourceAccount}"
        }
      }
    },
    {
      "Sid": "CloudWatchMetrics",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": [
            "AWS/IoTManagedIntegrations",
            "AWS/Usage"
          ]
        }
      }
    }
  ]
}
```

Integrazioni gestite, aggiornamenti alle politiche gestite. AWS

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per le integrazioni gestite 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 documenti delle integrazioni gestite.

Modifica	Descrizione	Data
Le integrazioni gestite hanno iniziato a tenere traccia delle modifiche	Le integrazioni gestite hanno iniziato a tenere traccia delle modifiche relative alle politiche AWS gestite.	03 marzo 2025

Come funzionano le integrazioni gestite con IAM

Prima di utilizzare IAM per gestire l'accesso alle integrazioni gestite, scopri quali funzionalità IAM sono disponibili per l'uso con le integrazioni gestite.

Funzionalità IAM che puoi utilizzare con le integrazioni gestite

Funzionalità IAM	Supporto per integrazioni gestite
Policy basate sull'identità	Sì
Policy basate su risorse	No
Operazioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione delle policy	Sì
ACLs	No
ABAC (tag nelle policy)	No
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì
Ruoli di servizio	Sì

Funzionalità IAM	Supporto per integrazioni gestite
Ruoli collegati al servizio	Sì

Per avere una visione di alto livello di come le integrazioni gestite e gli altri AWS servizi funzionano con la maggior parte delle funzionalità IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Politiche basate sull'identità per integrazioni gestite

Supporta le policy basate sull'identità: sì

Le policy basate sull'identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente di IAM.

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Esempi di policy basate sull'identità per integrazioni gestite

Per visualizzare esempi di politiche basate sull'identità di integrazioni gestite, consulta. Esempi di policy basate sull'identità per integrazioni gestite

Politiche basate sulle risorse all'interno delle integrazioni gestite

Supporta le policy basate su risorse: no

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy di bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio specificare un'entità principale. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata sulle risorse. Per ulteriori informazioni, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Azioni politiche per le integrazioni gestite

Supporta le operazioni di policy: si

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.

Per visualizzare un elenco delle azioni di integrazione gestite, consulta Azioni definite dalle integrazioni gestite nel Service Authorization Reference.

Le azioni politiche nelle integrazioni gestite utilizzano il seguente prefisso prima dell'azione:

```
iot-mi
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "iot-mi:action1",
      "iot-mi:action2"
         ]
```

Per visualizzare esempi di politiche basate sull'identità per le integrazioni gestite, consulta. Esempi di policy basate sull'identità per integrazioni gestite

Risorse politiche per le integrazioni gestite

Supporta le risorse relative alle policy: sì

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": "*"
```

Per visualizzare un elenco dei tipi di risorse per le integrazioni gestite e relativi ARNs, consulta Risorse definite dalle integrazioni gestite nel Service Authorization Reference. Per sapere con quali azioni puoi specificare l'ARN di ogni risorsa, vedi Azioni definite dalle integrazioni gestite.

Per visualizzare esempi di politiche basate sull'identità per le integrazioni gestite, consulta. Esempi di policy basate sull'identità per integrazioni gestite

Chiavi delle condizioni delle policy per le integrazioni gestite

Supporta le chiavi di condizione delle policy specifiche del servizio: sì

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.

Per visualizzare un elenco delle chiavi di condizione delle integrazioni gestite, consulta Condition Keys for Managed integrations nel Service Authorization Reference. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, vedi Azioni definite dalle integrazioni gestite.

Per visualizzare esempi di politiche basate sull'identità per le integrazioni gestite, consulta. Esempi di policy basate sull'identità per integrazioni gestite

ACLs nelle integrazioni gestite

Supporti ACLs: No

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.

ABAC con integrazioni gestite

Supporta ABAC (tag nelle policy): parzialmente

Il controllo degli accessi basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi, chiamati tag. Puoi allegare tag a entità e AWS risorse IAM, quindi progettare politiche ABAC per consentire le operazioni quando il tag del principale corrisponde al tag sulla risorsa.

Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Sì. Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per maggiori informazioni su ABAC, consulta Definizione delle autorizzazioni con autorizzazione ABAC nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Utilizzo di credenziali temporanee con integrazioni gestite

Supporta le credenziali temporanee: sì

Le credenziali temporanee forniscono l'accesso a breve termine alle AWS risorse e vengono create automaticamente quando si utilizza la federazione o si cambia ruolo. AWS consiglia di generare dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza temporanee in IAM e Servizi AWS compatibili con IAM nella Guida per l'utente IAM.

Autorizzazioni principali multiservizio per le integrazioni gestite

Supporta l'inoltro delle sessioni di accesso (FAS): sì

Le sessioni di accesso inoltrato (FAS) utilizzano le autorizzazioni del principale chiamante an Servizio AWS, combinate con la richiesta di effettuare richieste Servizio AWS ai servizi downstream. Per i dettagli delle policy relative alle richieste FAS, consulta Forward access sessions.

Ruoli di servizio per le integrazioni gestite

Supporta i ruoli di servizio: sì

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe interrompere la funzionalità delle integrazioni gestite. Modifica i ruoli di servizio solo quando le integrazioni gestite forniscono indicazioni in tal senso.

Ruoli collegati ai servizi per le integrazioni gestite

Supporta i ruoli collegati ai servizi: sì

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati al servizio, ma non modificarle.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che include un `Yes` nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Esempi di policy basate sull'identità per integrazioni gestite

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse di integrazione gestite. Per concedere agli utenti l’autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM (console) nella Guida per l'utente di IAM.

Per informazioni dettagliate sulle azioni e sui tipi di risorse definiti dalle integrazioni gestite, incluso il formato di ARNs per ogni tipo di risorsa, consulta Azioni, risorse e chiavi di condizione per le integrazioni gestite nel Service Authorization Reference.

Argomenti

- Best practice per le policy
- Utilizzo della console di integrazione gestita
- Consentire agli utenti di visualizzare le loro autorizzazioni

Best practice per le policy

Le politiche basate sull'identità determinano se qualcuno può creare, accedere o eliminare le risorse di integrazione gestite 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 di integrazione gestita

Per accedere alla console di integrazioni gestite, devi disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse di integrazione gestite presenti nel 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.

Non è necessario consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso AWS CLI o l'API. AWS Al contrario, è opportuno concedere l'accesso solo alle azioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che utenti e ruoli possano continuare a utilizzare la console di integrazione gestita, collega anche le integrazioni gestite `ConsoleAccess` o la policy `ReadOnly` AWS gestita alle entità. Per maggiori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente di IAM.

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": "*"
        }
    ]
}
```

Risoluzione dei problemi relativi alle integrazioni gestite, all'identità e all'accesso

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con integrazioni gestite e IAM.

Argomenti

Non sono autorizzato a eseguire un'azione nelle integrazioni gestite

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` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `iot-mi:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: iot-
mi:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `iot-mi:`*`GetWidget`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue politiche devono essere aggiornate per consentirti di assegnare un ruolo alle integrazioni gestite.

Alcuni Servizi AWS consentono di trasferire 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` tenta di utilizzare la console per eseguire un'azione nelle integrazioni gestite. Tuttavia, l'azione 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 risorse gestite per le integrazioni

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 le integrazioni gestite supportano queste funzionalità, consulta. Come funzionano le integrazioni gestite con IAM
- Per scoprire come fornire l'accesso alle tue risorse su tutto Account AWS ciò che possiedi, 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.

Utilizzo di ruoli collegati ai servizi per integrazioni gestite

Le integrazioni gestite per AWS IoT Device Management utilizzano ruoli collegati ai servizi AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente alle integrazioni gestite. I ruoli collegati ai servizi sono predefiniti dalle integrazioni gestite e includono tutte le autorizzazioni richieste dal servizio per chiamare altri servizi per tuo conto. AWS

Un ruolo collegato ai servizi semplifica la configurazione delle integrazioni gestite perché non è necessario aggiungere manualmente le autorizzazioni necessarie. Le integrazioni gestite per AWS IoT Device Management definiscono le autorizzazioni dei relativi ruoli collegati ai servizi e, se non diversamente definito, solo le integrazioni gestite possono assumerne i ruoli. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato al servizio solo dopo avere eliminato le risorse correlate. In questo modo proteggi le tue risorse gestite per le integrazioni perché non puoi rimuovere inavvertitamente l'autorizzazione ad accedere alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta i AWS servizi che funzionano con IAM e cerca i servizi con Sì nella colonna Ruoli collegati ai servizi. Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato al servizio per tale servizio.

Autorizzazioni di ruolo collegate ai servizi per le integrazioni gestite

Le integrazioni gestite per la gestione dei AWS IoT dispositivi utilizzano il ruolo collegato al servizio denominato AWSServiceRoleForIoTManagedIntegrazioni: fornisce alle integrazioni gestite per AWS IoT Device Management l'autorizzazione a pubblicare log e metriche per conto dell'utente.

Il ruolo collegato ai servizi di AWSService RoleForIo TManaged Integrations prevede che i seguenti servizi assumano il ruolo:

- `iotmanagedintegrations.amazonaws.com`

La politica di autorizzazione dei ruoli denominata AWSIo TManaged IntegrationsServiceRolePolicy consente alle integrazioni gestite di completare le seguenti azioni sulle risorse specificate:

- Operazione: `logs:CreateLogGroup, logs:DescribeLogGroups, logs:CreateLogStream, logs:PutLogEvents, logs:DescribeLogStreams, cloudwatch:PutMetricData` on all of your managed integrations resources.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Sid" : "CloudWatchLogs",
      "Effect" : "Allow",
      "Action" : [
        "logs:CreateLogGroup",
        "logs:DescribeLogGroups"
      ],
      "Resource" : [
        "arn:aws:logs:*:*:log-group:/aws/iotmanagedintegrations/*"
      ]
    },
    {
      "Sid" : "CloudWatchStreams",
      "Effect" : "Allow",
      "Action" : [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource" : [
        "arn:aws:logs:*:*:log-group:/aws/iotmanagedintegrations/*:log-stream:*"
      ]
    },
    {
      "Sid" : "CloudWatchMetrics",
      "Effect" : "Allow",
      "Action" : [
        "cloudwatch:PutMetricData"
      ],
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "cloudwatch:namespace" : [
```

```
            "AWS/IoTManagedIntegrations",
            "AWS/Usage"
          ]
        }
      }
    }
  ]
}
```

Per consentire a utenti, gruppi o ruoli di creare, modificare o eliminare un ruolo orientato ai servizi, devi configurare le autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente IAM.

Creazione di un ruolo collegato ai servizi per le integrazioni gestite

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando causi un tipo di evento, ad esempio una chiamata a `PutRuntimeLogConfigurationCreateEventLogConfiguration`, o ai comandi `RegisterCustomEndpoint` API contenuti in Console di gestione AWS, the, o nell' AWS API AWS CLI, le integrazioni gestite creano automaticamente il ruolo collegato al servizio. Per ulteriori informazioni su`PutRuntimeLogConfiguration`, o`CreateEventLogConfiguration`, consulta `RegisterCustomEndpoint` PutRuntimeLogConfiguration, CreateEventLogConfigurationo. RegisterCustomEndpoint

Se elimini questo ruolo collegato al servizio, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando causi un tipo di evento, ad esempio la chiamata ai comandi `PutRuntimeLogConfigurationCreateEventLogConfiguration`, o `RegisterCustomEndpoint` API, Managed Integrations crea nuovamente il ruolo collegato al servizio. In alternativa, puoi contattare l'assistenza AWS clienti tramite AWS Support Center Console. Per ulteriori informazioni sui piani di AWS supporto, consulta Compare AWS Support Plans.

Puoi anche utilizzare la console IAM per creare un ruolo collegato ai servizi con lo use case IoT ManagedIntegrations - Managed Role. Nella AWS CLI o nell' AWS API, crea un ruolo collegato al servizio con il nome del servizio. `iotmanagedintegrations.amazonaws.com` Per ulteriori informazioni, consulta Creazione di un ruolo collegato ai servizi nella Guida per l'utente IAM. Se elimini il ruolo collegato ai servizi, è possibile utilizzare lo stesso processo per crearlo nuovamente.

Modifica di un ruolo collegato al servizio per le integrazioni gestite

le integrazioni gestite non consentono di modificare il ruolo collegato al servizio di Integrations. AWSService RoleForIo TManaged Dopo avere creato un ruolo collegato al servizio, non sarà possibile modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato al servizio per le integrazioni gestite

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato al servizio prima di poterlo eliminare manualmente.

> **Note**
>
> Se le integrazioni gestite utilizzano il ruolo quando si tenta di eliminare le risorse, l'eliminazione potrebbe non riuscire. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

Per eliminare manualmente il ruolo collegato ai servizi mediante IAM

Utilizza la console IAM AWS CLI, o l' AWS API per eliminare il ruolo collegato al servizio AWSService RoleForIo TManaged Integrations. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente IAM.

Regioni supportate per le integrazioni gestite, ruoli collegati ai servizi

Le integrazioni gestite per la gestione dei AWS IoT dispositivi supportano l'utilizzo di ruoli collegati al servizio in tutte le regioni in cui il servizio è disponibile. Per maggiori informazioni, consulta Regioni ed endpoint di AWS.

Utilizzalo Gestione dei segreti AWS per la protezione dei dati per i flussi di lavoro C2C

Gestione dei segreti AWS è un servizio di archiviazione segreto che puoi utilizzare per proteggere le credenziali del database, le chiavi API e altre informazioni segrete. Quindi, nel codice, puoi sostituire

le credenziali hardcoded con una chiamata API a Secrets Manager. Questo aiuta a garantire che il segreto non possa essere compromesso da qualcuno che esamina il codice, perché il segreto non è presente. Per una panoramica, consulta la Guida per l'utente di Gestione dei segreti AWS.

Secrets Manager crittografa i segreti utilizzando AWS Key Management Service le chiavi. Per ulteriori informazioni consulta la sezione Crittografia e decrittografia dei segreti in AWS Key Management Service.

Integrazioni gestite per AWS IoT Device Management l'integrazione Gestione dei segreti AWS in modo da poter archiviare i dati in Secrets Manager e utilizzare l'ID segreto nelle configurazioni.

In che modo le integrazioni gestite utilizzano i segreti

Open Authorization (OAuth) è uno standard aperto per l'autorizzazione all'accesso delegato, che consente agli utenti di concedere a siti Web o applicazioni l'accesso alle proprie informazioni su altri siti Web senza condividere le proprie password. È un modo sicuro per le applicazioni di terze parti di accedere ai dati degli utenti per conto dell'utente, fornendo un'alternativa più sicura alla condivisione delle password.

In OAuth, un ID client e un client secret sono credenziali che identificano e autenticano un'applicazione client quando richiede un token di accesso.

Integrazioni gestite per consentire OAuth agli AWS IoT Device Management utenti di comunicare con i clienti che utilizzano i flussi di lavoro C2C. I clienti devono fornire l'ID client e il segreto del client per comunicare. I clienti delle integrazioni gestite memorizzeranno un ID cliente e un segreto del cliente nei propri AWS account, mentre le integrazioni gestite leggeranno l'ID cliente e il segreto del cliente nel nostro account cliente.

Come creare un segreto

Per creare un segreto, segui la procedura descritta in Creare un Gestione dei segreti AWS segreto nella Guida per l' Gestione dei segreti AWS utente.

Devi creare il tuo segreto con una AWS KMS chiave gestita dal cliente per consentire alle integrazioni gestite di leggere il valore segreto. Per ulteriori informazioni, consulta Autorizzazioni per la AWS KMS chiave nella Guida per l'utente. Gestione dei segreti AWS

È inoltre necessario utilizzare le politiche IAM nella sezione seguente.

Concedi l'accesso alle integrazioni gestite AWS IoT Device Management per recuperare il segreto

Per consentire alle integrazioni gestite di recuperare il valore segreto da Secrets Manager, includi le seguenti autorizzazioni nella politica delle risorse per il segreto al momento della creazione.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal" : {
      "Service" : "iotmanagedintegrations.amazonaws.com"
    },
    "Action" : [ "secretsmanager:GetSecretValue" ],
    "Resource" : "*" ,
    "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:iotmanagedintegrations:AWS Region:account-
id:account-association:account-association-id"

        }
      }
  } ]
}
```

Aggiungi la seguente dichiarazione alla politica per la tua chiave gestita dal cliente. AWS KMS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Principal": {
```

```
                "Service": [
                    "iotmanagedintegrations.amazonaws.com"
                ]
            },
            "Resource": [
            "arn:aws:kms:us-east-1:123456789012:key/*"
            ]
        }

    ]
}
```

Convalida della conformità per le integrazioni gestite

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 .

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.

Utilizza integrazioni gestite con endpoint VPC di interfaccia

Puoi stabilire una connessione privata tra le tue integrazioni Amazon VPC e AWS IoT Managed creando un endpoint Amazon VPC di interfaccia. Gli endpoint di interfaccia sono alimentati da AWS PrivateLink, una tecnologia che consente di accedere in modo privato ai servizi utilizzando indirizzi IP privati. AWS PrivateLink limita tutto il traffico di rete tra le integrazioni gestite da VPC e IoT alla rete Amazon. Non è necessario un gateway Internet, un dispositivo NAT o una connessione VPN.

Non è necessario utilizzarlo AWS PrivateLink, ma è consigliato. Per ulteriori informazioni sugli AWS PrivateLink endpoint VPC, consulta Accesso ai AWS servizi AWS PrivateLink nella Guida.AWS PrivateLink

Argomenti

Considerazioni sugli endpoint AWS IoT VPC per integrazioni gestite

Prima di configurare un endpoint VPC di interfaccia per le integrazioni AWS IoT gestite, consulta le proprietà e le limitazioni degli endpoint dell'interfaccia nella Guida.AWS PrivateLink

AWS IoT Le integrazioni gestite supportano l'esecuzione di chiamate a tutte le sue azioni API dal tuo VPC tramite gli endpoint VPC dell'interfaccia.

Endpoint supportati

AWS IoT Le integrazioni gestite supportano gli endpoint VPC per le seguenti interfacce di servizio:

- API del piano di controllo: `com.amazonaws.region.iotmanagedintegrations.api`

Endpoint non supportati

I seguenti endpoint di integrazioni AWS IoT gestite non supportano gli endpoint VPC:

- Endpoint MQTT: i dispositivi MQTT vengono in genere implementati negli ambienti degli utenti finali anziché all'interno, rendendo superflua l'integrazione. AWS VPCs AWS PrivateLink
- OAuth endpoint di callback: molte piattaforme di terze parti non funzionano all'interno dell' AWS infrastruttura, il che riduce i vantaggi del supporto per i flussi. AWS PrivateLink OAuth

Disponibilità

AWS IoT Gli endpoint VPC con integrazioni gestite sono disponibili nelle seguenti regioni: AWS

- Canada (Centrale) - ca-central-1

- Europa (Irlanda) - eu-west-1

Le regioni aggiuntive saranno supportate man mano che AWS IoT Managed integrations ne amplierà la disponibilità.

Supporto dual-stack

AWS IoT Integrazioni gestite Gli endpoint VPC supportano IPv4 sia il traffico che il traffico. IPv6 Puoi creare endpoint VPC con i seguenti tipi di indirizzi IP:

- IPv4: assegna IPv4 gli indirizzi alle interfacce di rete degli endpoint
- IPv6: assegna IPv6 gli indirizzi alle interfacce di rete degli endpoint (richiede solo sottoreti) IPv6
- Dualstack: assegna entrambi gli indirizzi alle interfacce di rete degli endpoint IPv4 IPv6

Creazione di un endpoint VPC di interfaccia per integrazioni gestite AWS IoT

Puoi creare un endpoint VPC per il servizio di integrazioni AWS IoT gestite utilizzando la console Amazon VPC o (CLI). AWS CLI AWS

Per creare un endpoint VPC di interfaccia per integrazioni AWS IoT gestite (console)

1. Apri la console Amazon VPC su Amazon VPC Console.
2. Nel pannello di navigazione, seleziona Endpoints (Endpoint).
3. Seleziona Crea endpoint.
4. Per Service category (Categoria servizio), scegli AWS services.
5. Per Nome del servizio, seleziona il nome del servizio che corrisponde alla tua AWS regione. Ad esempio:
   - `com.amazonaws.ca-central-1.iotmanagedintegrations.api`
   - `com.amazonaws.eu-west-1.iotmanagedintegrations.api`
6. Per VPC, seleziona il VPC da cui accederai alle integrazioni gestite. AWS IoT
7. Per impostazioni aggiuntive, l'opzione Abilita nome DNS è selezionata per impostazione predefinita. Ti consigliamo di mantenere questa impostazione. Ciò garantisce che le richieste agli endpoint del servizio pubblico di AWS IoT Managed integrations vengano risolte sul tuo endpoint Amazon VPC.

8. Per Subnet, seleziona le sottoreti in cui creare interfacce di rete endpoint. È possibile selezionare una sottorete per zona di disponibilità.
9. Per IP address type (Tipo di indirizzo IP), seleziona una delle opzioni seguenti:
   - IPv4: assegna IPv4 indirizzi alle interfacce di rete degli endpoint
   - IPv6: Assegna IPv6 indirizzi alle interfacce di rete degli endpoint (supportata solo se tutte le sottoreti selezionate sono -only) IPv6
   - Dualstack: assegna entrambi gli indirizzi E alle interfacce di rete degli endpoint IPv4 IPv6
10. Per Security groups (Gruppi di sicurezza), seleziona i gruppi di sicurezza da associare alle interfacce di rete dell'endpoint. Le regole del gruppo di sicurezza devono consentire la comunicazione tra l'interfaccia di rete dell'endpoint e le risorse del VPC che comunicano con il servizio.
11. Per Policy, scegli Accesso completo per consentire tutte le operazioni da parte di tutti i principali su tutte le risorse sull'endpoint dell'interfaccia. Per limitare l'accesso, scegli Personalizzato e specifica una politica.
12. (Facoltativo) Per aggiungere un tag, scegli Add new tag (Aggiungi nuovo tag) e immetti la chiave e il valore del tag.
13. Seleziona Crea endpoint.

Per creare un endpoint VPC di interfaccia per IoT Managed Integrations (AWS CLI)

Usa il create-vpc-endpointcomando e specifica l'ID VPC, il tipo di endpoint VPC (interfaccia), il nome del servizio, le sottoreti che utilizzeranno l'endpoint e i gruppi di sicurezza da associare alle interfacce di rete degli endpoint.

```
aws ec2 create-vpc-endpoint \
  --vpc-id vpc-12345678 \
  --route-table-ids rtb-12345678 \
  --service-name com.amazonaws.ca-central-1.iotmanagedintegrations.api \
  --vpc-endpoint-type Interface \
  --subnet-ids subnet-12345678 subnet-87654321 \
  --security-group-ids sg-12345678
```

Test del tuo endpoint VPC

Dopo aver creato l'endpoint VPC, puoi testare la connessione effettuando chiamate API alle integrazioni AWS IoT gestite da un'istanza EC2 nel tuo VPC.

Prerequisiti

- Un' EC2 istanza in una sottorete privata all'interno del tuo VPC
- Autorizzazioni IAM appropriate per AWS IoT le operazioni di integrazione gestite
- Regole del gruppo di sicurezza che consentono il traffico HTTPS (porta 443) verso l'endpoint VPC

Test della connessione

1. Connettiti alla tua EC2 istanza Amazon nella sottorete privata.
2. Verifica la risoluzione DNS per il nome DNS privato:

```
dig api.iotmanagedintegrations.region.api.aws
```

3. Verifica la connettività HTTPS:

```
curl -v https://api.iotmanagedintegrations.region.api.aws
```

4. Effettua una chiamata API per le integrazioni AWS IoT gestite:

```
aws iot-managed-integrations list-destinations \
  --region region \
  --endpoint-url https://api.iotmanagedintegrations.region.api.aws
```

Sostituisci `region` con la tua AWS regione (ad esempio,`ca-central-1`).

Controllo dell'accesso ai servizi tramite endpoint VPC

Una policy degli endpoint VPC è una policy delle risorse IAM che si collega a un endpoint VPC di interfaccia quando si crea o si modifica l'endpoint. Se non colleghi una policy durante la creazione di un endpoint, viene collegata una policy predefinita che consente l'accesso completo al servizio. Una policy endpoint non esclude né sostituisce policy dell'utente IAM o policy specifiche del servizio. Si tratta di una policy separata per controllare l'accesso dall'endpoint al servizio specificato.

Le policy endpoint devono essere scritte in formato JSON. Per ulteriori informazioni, consulta Controllo degli accessi ai servizi con endpoint VPC in Guida per l'utente di Amazon VPC.

Esempio: policy sugli endpoint VPC per AWS IoT le azioni di integrazione gestite

Di seguito è riportato un esempio di policy sugli endpoint per le integrazioni gestite. AWS IoT Questa policy consente agli utenti di connettersi alle integrazioni AWS IoT gestite tramite l'endpoint VPC per accedere alle destinazioni, ma nega l'accesso agli armadietti delle credenziali.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "iotmanagedintegrations:ListDestinations",
                "iotmanagedintegrations:GetDestination",
                "iotmanagedintegrations:CreateDestination",
                "iotmanagedintegrations:UpdateDestination",
                "iotmanagedintegrations:DeleteDestination"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "iotmanagedintegrations:ListCredentialLockers",
                "iotmanagedintegrations:GetCredentialLocker",
                "iotmanagedintegrations:CreateCredentialLocker",
                "iotmanagedintegrations:UpdateCredentialLocker",
                "iotmanagedintegrations:DeleteCredentialLocker"
            ],
            "Resource": "*"
        }
    ]
}
```

Esempio: policy degli endpoint VPC che limita l'accesso a uno specifico ruolo IAM

La seguente policy sugli endpoint VPC consente l'accesso alle integrazioni AWS IoT gestite solo per i principali IAM che hanno il ruolo IAM specificato nella loro catena di fiducia. A tutti gli altri principali IAM viene negato l'accesso.

```
{
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/
IoTManagedIntegrationsVPCRole"
                }
            }
        }
    ]
}
```

Prezzi

Ti vengono addebitate tariffe standard per la creazione e l'utilizzo di un endpoint VPC di interfaccia con AWS IoT integrazioni gestite. Per ulteriori informazioni, consultare Prezzi di AWS PrivateLink.

Limitazioni

- L'CreateAccountAssociationAPI è progettata per funzionare OAuth con servizi cloud di terze parti, che richiedono la richiesta di lasciare la rete Amazon. Questo è importante per i clienti che desiderano contenere il traffico all'interno del VPC, in quanto AWS PrivateLink non possono fornire un end-to-end contenimento completo per questa chiamata API AWS PrivateLink .
- Gli endpoint VPC per le integrazioni AWS IoT gestite non sono disponibili in. AWS GovCloud (US) Regions

Per le limitazioni generali degli endpoint VPC, consulta le proprietà e le limitazioni degli endpoint dell'interfaccia nella Amazon VPC User Guide.

Connect a integrazioni gestite per endpoint AWS IoT Device Management FIPS

AWS IoT fornisce un endpoint del piano di controllo che supporta il Federal Information Processing Standard (FIPS) 140-2. Gli endpoint conformi a FIPS sono diversi dagli endpoint standard. AWS Per

interagire con le integrazioni gestite AWS IoT Device Management in modo conforme allo standard FIPS, è necessario utilizzare gli endpoint descritti di seguito con il client conforme a FIPS. AWS IoT La console non è conforme a FIPS.

Le sezioni seguenti descrivono come accedere all' AWS IoT endpoint conforme a FIPS utilizzando l'API REST, un SDK o il. AWS CLI

Endpoint del piano di controllo

Gli endpoint del piano di controllo conformi a FIPS che supportano le operazioni di integrazione gestita e i relativi comandi sono elencati in FIPS Endpoints by Service. AWS CLI In FIPS Endpoints by Service, trova il servizio AWS IoT Device Management - Managed integrations 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

Monitoraggio delle integrazioni gestite

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni delle integrazioni gestite e delle altre soluzioni AWS. AWS fornisce i seguenti strumenti di monitoraggio per monitorare le integrazioni gestite, segnalare quando qualcosa non va e intraprendere azioni automatiche quando necessario:

- AWS CloudTrailacquisisce le chiamate API e gli eventi correlati effettuati da o per conto del tuo AWS account e invia i file di log a un bucket Amazon S3 da te specificato. Puoi identificare quali utenti e account hanno chiamato AWS, l'indirizzo IP di origine da cui sono state effettuate le chiamate e quando sono avvenute le chiamate. Per ulteriori informazioni, consulta la Guida per l'utente AWS CloudTrail.

Registrazione delle chiamate API di integrazioni gestite utilizzando AWS CloudTrail

Le integrazioni gestite sono integrate con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o un. Servizio AWS CloudTrail acquisisce tutte le chiamate API per le integrazioni gestite come eventi. Le chiamate acquisite includono le chiamate dalla console delle integrazioni gestite e le chiamate in codice alle operazioni dell'API delle integrazioni gestite. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare la richiesta effettuata alle integrazioni gestite, l'indirizzo IP da cui è stata effettuata la richiesta, quando è stata effettuata e dettagli aggiuntivi.

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 utente root o utente.
- Se la richiesta è stata effettuata per conto di un utente del Centro identità IAM.
- 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 Servizio AWS.

CloudTrail è attivo nel tuo account Account AWS quando crei l'account e hai automaticamente accesso alla cronologia degli CloudTrail eventi. La cronologia CloudTrail degli eventi fornisce un

record visualizzabile, ricercabile, scaricabile e immutabile degli ultimi 90 giorni di eventi di gestione registrati in un. Regione AWSPer ulteriori informazioni, consulta Lavorare con la cronologia degli CloudTrail eventi nella Guida per l'utente.AWS CloudTrail Non sono CloudTrail previsti costi per la visualizzazione della cronologia degli eventi.

Per una registrazione continua degli eventi degli Account AWS ultimi 90 giorni, crea un trail o un data store di eventi CloudTrailLake.

CloudTrail sentieri

Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Tutti i percorsi creati utilizzando il Console di gestione AWS sono multiregionali. È possibile creare un trail per una singola Regione o per più Regioni tramite AWS CLI. La creazione di un percorso multiregionale è consigliata in quanto consente di registrare l'intera attività del proprio Regioni AWS account. Se si crea un trail per una singola Regione, è possibile visualizzare solo gli eventi registrati nella Regione AWS del trail. Per ulteriori informazioni sui trail, consulta Creating a trail for your Account AWS e Creating a trail for an organization nella Guida per l'utente di AWS CloudTrail .

Puoi inviare gratuitamente una copia dei tuoi eventi di gestione in corso al tuo bucket Amazon S3 CloudTrail creando un percorso, tuttavia ci sono costi di storage di Amazon S3. Per ulteriori informazioni sui CloudTrail prezzi, consulta la pagina Prezzi.AWS CloudTrail Per informazioni sui prezzi di Amazon S3, consulta Prezzi di Amazon S3.

CloudTrail Lake Event Data Store

CloudTrail Lake ti consente di eseguire query basate su SQL sui tuoi eventi. CloudTrail Lake converte gli eventi esistenti in formato JSON basato su righe in formato Apache ORC. ORC è un formato di archiviazione a colonne ottimizzato per il recupero rapido dei dati. Gli eventi vengono aggregati in archivi di dati degli eventi, che sono raccolte di eventi immutabili basate sui criteri selezionati applicando i selettori di eventi avanzati. I selettori applicati a un archivio di dati degli eventi controllano quali eventi persistono e sono disponibili per l'esecuzione della query. Per ulteriori informazioni su CloudTrail Lake, consulta Working with AWS CloudTrail Lake nella Guida per l'utente.AWS CloudTrail

CloudTrail Gli archivi e le richieste di dati sugli eventi di Lake comportano dei costi. Quando crei un datastore di eventi, scegli l'opzione di prezzo da utilizzare per tale datastore. L'opzione di prezzo determina il costo per l'importazione e l'archiviazione degli eventi, nonché il periodo di conservazione predefinito e quello massimo per il datastore di eventi. Per ulteriori informazioni sui CloudTrail prezzi, consulta Prezzi.AWS CloudTrail

Eventi gestionali in CloudTrail

Gli eventi di gestione forniscono informazioni sulle operazioni di gestione eseguite sulle risorse di Account AWS. Queste operazioni sono definite anche operazioni del piano di controllo (control-plane). Per impostazione predefinita, CloudTrail registra gli eventi di gestione.

Le integrazioni gestite registrano le seguenti operazioni del piano di controllo delle integrazioni gestite come eventi di CloudTrail gestione.

- `CreateCloudConnector`
- `UpdateCloudConnector`
- `GetCloudConnector`
- `DeleteCloudConnector`
- `ListCloudConnectors`
- `CreateConnectorDestination`
- `UpdateConnectorDestination`
- `GetConnectorDestination`
- `DeleteConnectorDestination`
- `ListConnectorDestinations`
- `CreateAccountAssociation`
- `UpdateAccountAssociation`
- `GetAccountAssociation`
- `DeleteAccountAssociation`
- `ListAccountAssociations`
- `StartAccountAssociationRefresh`
- `ListManagedThingAccountAssociations`
- `RegisterAccountAssociation`
- `DeregisterAccountAssociation`
- `SendConnectorEvent`
- `ListDeviceDiscoveries`
- `ListDiscoveredDevices`

Esempi di eventi

Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'operazione API richiesta, la data e l'ora dell'operazione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia stack ordinata delle chiamate API pubbliche, quindi gli eventi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra un CloudTrail evento che dimostra il corretto funzionamento dell'`CreateCloudConnector`API.

CloudTrail Evento riuscito con l'operazione **`CreateCloudConnector`** API.


```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/EXAMPLE",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLEKYSBQSCGRIC",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAZOZQFKYSFZVB2J2GN",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2025-06-05T18:26:16Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2025-06-05T18:30:40Z",
    "eventSource": "iotmanagedintegrations.amazonaws.com",
    "eventName": "CreateCloudConnector",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "PostmanRuntime/7.44.0",
    "requestParameters": {
        "EndpointType": "LAMBDA",
        "Description": "Manual testing for C2C CT Validation",
```

```
        "ClientToken": "abc7460",
        "EndpointConfig": {
            "lambda": {
                "arn": "arn:aws:lambda:us-
east-1:111122223333:function:LightweightMockConnector7460"
            }
        },
        "Name": "EdenManualTestCloudConnector"
    },
    "responseElements": {
        "X-Frame-Options": "DENY",
        "Access-Control-Expose-Headers": "Content-Length,Content-Type,X-Amzn-
Errortype,X-Amzn-Requestid",
        "Strict-Transport-Security": "max-age:47304000; includeSubDomains",
        "Cache-Control": "no-store, no-cache",
        "X-Content-Type-Options": "nosniff",
        "Content-Security-Policy": "upgrade-insecure-requests; default-src 'none';
 object-src 'none'; frame-ancestors 'none'; base-uri 'none'",
        "Pragma": "no-cache",
        "Id": "f7e633e719404c4a933596b4d0cc276e",
        "Arn": "arn:aws:iotmanagedintegrations:us-east-1:111122223333:cloud-connector/
EXAMPLE404c4a933596b4d0cc276e"
    },
    "requestID": "c0071fd1-b8e0-400a-bcc0-EXAMPLE9e4",
    "eventID": "95b318ea-2f63-4183-9c22-EXAMPLE3e",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

L'esempio seguente mostra un CloudTrail evento che dimostra un'operazione `ListDiscoveredDevices` API riuscita.

CloudTrail Evento riuscito con l'operazione **`ListDiscoveredDevices`** API.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EZAMPLE",
        "arn": "arn:aws:sts::444455556666:assumed-role/Admin/EXAMPLE",
```

```
        "accountId": "444455556666",
        "accessKeyId": "EXAMPLERJ26PYMH",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLE",
                "arn": "arn:aws:iam::444455556666:role/Admin",
                "accountId": "444455556666",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2025-06-10T23:37:31Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2025-06-10T23:38:07Z",
    "eventSource": "iotmanagedintegrations.amazonaws.com",
    "eventName": "ListDiscoveredDevices",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "EXAMPLE-runtime/2.4.0",
    "requestParameters": {
        "Identifier": "EXAMPLE4f268483a17d8060f014"
    },
    "responseElements": null,
    "requestID": "27ae1f61-e2e6-43e4-bf17-EXAMPLEa568",
    "eventID": "34734e81-76a8-49a4-9641-EXAMPLE28ed",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "444455556666",
    "eventCategory": "Management"
}
```


Per informazioni sul contenuto dei CloudTrail record, consulta il contenuto dei CloudTrail record nella Guida AWS CloudTrail per l'utente.

Cronologia dei documenti per la Guida per gli sviluppatori alle integrazioni gestite

La tabella seguente descrive le versioni della documentazione per le integrazioni gestite.

Modifica	Descrizione	Data
Versione di disponibilità generale	Versione a disponibilità generale della Guida per gli sviluppatori alle integrazioni gestite	25 giugno 2025
Versione di anteprima iniziale	Versione di anteprima iniziale della Guida per gli sviluppatori alle integrazioni gestite	3 marzo 2025

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.