Guía del usuario de

AWS Private Certificate Authority

Version latest

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

AWS Private Certificate Authority: Guía del usuario de

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

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon, de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es Autoridad de certificación privada de AWS? ... 1
Disponibilidad regional ... 1
Servicios integrados ... 2
Algoritmos compatibles ... 2
Conformidad con la RFC 5280 ... 4
Precios ... 5
Términos y conceptos para AWS Private CA ... 6
Confianza ... 6
Certificados de servidor TLS ... 7
Firma del certificado ... 7
Certificate authority (Autoridad de certificado) ... 7
CA raíz ... 7
Certificado de entidad de certificación ... 8
Certificado de entidad de certificación raíz ... 9
Certificado de entidad final ... 9
Certificados autofirmados ... 9
Certificado privado ... 10
Ruta de acceso del certificado ... 11
Restricción de longitud de ruta ... 11
¿Cuál es el mejor servicio de certificados para mis necesidades? ... 12
Prácticas recomendadas ... 14
Documentación de la estructura y las políticas de entidad de certificación (CA) ... 14
Minimizar el uso de la entidad de certificación (CA) raíz si es posible ... 14
Dele los suyos a la CA raíz Cuenta de AWS ... 15
Funciones separadas de administrador y emisor ... 15
Implementar la revocación gestionada de certificados ... 16
Encienda AWS CloudTrail ... 16
Rotar la clave privada de la CA ... 16
Eliminar los no utilizados CAs ... 17
Bloquee el acceso público a su CRLs ... 17
Prácticas recomendadas de aplicaciones de Amazon EKS ... 17
AWS Private CA Úselo con el AWS SDK para Java ... 18
Ejemplos de API ... 18
Crear y activar una CA raíz mediante programación ... 19

Crear y activar una CA subordinada mediante programación 28
CreateCertificateAuthority 37
Utilización CreateCertificateAuthority para dar soporte a Active Directory 41
CreateCertificateAuthorityAuditReport 50
CreatePermission 52
DeleteCertificateAuthority 55
DeletePermission 57
DeletePolicy 59
DescribeCertificateAuthority 61
DescribeCertificateAuthorityAuditReport 63
GetCertificate 66
GetCertificateAuthorityCertificate 69
GetCertificateAuthorityCsr 71
GetPolicy 74
ImportCertificateAuthorityCertificate 76
IssueCertificate 78
ListCertificateAuthorities 82
ListPermissions 86
ListTags 88
PutPolicy 90
RestoreCertificateAuthority 93
RevokeCertificate 94
TagCertificateAuthorities 97
UntagCertificateAuthority 99
UpdateCertificateAuthority 101
Cree CAs y certifique con nombres de asunto personalizados 103
Cree certificados con extensiones personalizadas 112
Ejemplos de materias 127
Activar una autoridad de certificación de productos (PAA) 127
Activar un intermedio de certificación de producto (PAI) 138
Crear un certificado de atestación de dispositivos (DAC) 149
Active una CA raíz para los certificados operativos de nodo (NOC). 153
Activar una CA subordinada para los certificados operativos de nodo (NOC) 163
Crear un certificado operativo de nodo (NOC) 173
Ejemplos de mDL 178
Activar un certificado de autoridad emisora de certificados (IACA) 178

Crear un certificado de firmante de documentos 187
Diseñe su solución para AWS Private CA 192
Diseñe una jerarquía de CA 192
Valide los certificados de la entidad final 194
Planifique la estructura de una jerarquía de CA 196
Establezca restricciones de longitud en la ruta de certificación 199
Gestione el ciclo de vida de la CA 201
Elija los períodos de validez 201
Administre la sucesión de CA 203
Revocar una CA 205
Planifique la revocación del certificado 205
Requisitos 208
Configurar una CRL 208
Personalice la URL de OCSP 215
Modo CA 219
De uso general (predeterminado) 219
Certificado de corta duración 219
Planifique la resiliencia 220
Redundancia y recuperación de desastres 220
Entidades de certificación 222
Configuración 223
Inscríbase en una Cuenta de AWS 223
Creación de un usuario con acceso administrativo 224
Instale el AWS Command Line Interface 225
Cree una CA privada 225
Ejemplos de CLI 234
Instale el certificado de CA 247
Algoritmos de firma compatibles 247
Instale un certificado de CA raíz 249
Instale un certificado de CA subordinado hospedado por Autoridad de certificación privada de AWS 257
Instale un certificado de CA subordinado firmado por una CA principal externa 259
Control del acceso a 259
Cree permisos de cuenta única para un usuario de IAM 260
Asociar una política para el acceso entre cuentas 263
Lista privada CAs 265

Ver una CA privada ... 268
Agregar etiquetas. ... 271
Estado CA ... 273
Relación entre el estado de la CA y el ciclo de vida de la CA ... 276
Actualización de una CA ... 277
Actualizar una CA (consola) ... 277
Actualizar una CA (CLI) ... 281
Eliminación de una CA ... 289
Restauración de una CA ... 291
Restauración de una CA privada (consola) ... 291
Restaure una CA privada (AWS CLI) ... 292
Certificados de CA firmados externamente ... 293
Emitir y administrar certificados ... 298
Emita certificados de entidad final privada ... 298
Emita un certificado estándar (AWS CLI) ... 300
Emita un certificado con un nombre de asunto personalizado mediante una plantilla APIPassthrough ... 302
Emita un certificado con extensiones personalizadas mediante una plantilla APIPassthrough ... 305
Recupera un certificado privado ... 306
Enumere los certificados privados ... 307
Exportación de un certificado ... 312
Revocar un certificado privado ... 313
Certificados y OCSP revocados ... 314
Certificados revocados en una CRL ... 314
Certificados revocados en un informe de auditoría ... 315
Automatice la exportación ... 316
Plantillas de certificado ... 317
Variedades de plantillas ... 318
Orden de operaciones de la plantilla ... 329
Definiciones de plantilla ... 330
Seguridad ... 375
IAM ... 376
Permisos de la API ... 377
AWS políticas gestionadas ... 383
Políticas administradas por el cliente ... 385

Políticas insertadas 386
Acceso entre cuentas 392
Políticas basadas en recursos 393
Protección de datos 397
Cumplimiento de las normas de seguridad y almacenamiento de las claves Autoridad de certificación privada de AWS privadas 398
Cifrado de datos en AWS Private CA Connector for Active Directory 398
Validación de conformidad 398
Cree un informe de auditoría 399
Seguridad de la infraestructura 406
Puntos de enlace de la VPC (AWS PrivateLink) 407
Compatibilidad con puntos de conexión de doble pila 412
Uso de IPv6 direcciones en IAM y AWS Private CA 412
CP/CPS 414
Requisitos y responsabilidades del CP/CPS 415
Supervisión de recursos de 427
AWS Private CA CloudWatch métricas 428
AWS Private CA Supervise con CloudWatch eventos 429
Éxito o error al crear una CA privada 429
Éxito o error al emitir un certificado 430
Éxito al revocar un certificado 431
Éxito o error al generar una CRL 432
Éxito o Error al crear un informe de auditoría de CA 434
CloudTrail registros 435
AWS Private CA información en CloudTrail 436
AWS Private CA eventos de gestión 437
Ejemplos AWS Private CA de eventos 438
Solución de problemas 441
Problemas de revocación de certificados 441
Latencia de respuesta OCSP 441
Revocación de certificados autofirmados 441
Mensajes de excepción 441
Errores de certificados compatibles con Matter 445
Proteja Kubernetes con AWS Private CA 449
Conceptos 449
Consideraciones 451

Uso multicuenta del administrador de certificados 452
Introducción 452
Instale cert-manager 455
Configurar los permisos de IAM 456
Instale y configure el emisor del AWS Private CA clúster 458
Administre el certificado del AWS Private CA cliente con cert-manager 463
Emita su primer certificado TLS 464
Ejemplos 465
Supervisión 465
Solución de problemas 466
Conector para Active Directory 398
¿Es la primera vez que utiliza el Conector para AD? 468
Conector de acceso para AD 469
Precios 469
Configuración 469
Paso 1: Cree una CA privada mediante AWS Private CA 469
Paso 2: Configurar un Active Directory 469
(Solo para Active Directory Connector) Paso 3: delegue los permisos a la cuenta de servicio 470
Paso 4: Crear una política de IAM 471
Paso 5: Comparta su CA privada con Connector para AD 474
Paso 6: Crear un registro de directorio 474
Paso 7: Configurar grupos de seguridad 474
Paso 8: Configurar el acceso a la red para los objetos de directorio 475
Introducción 475
Antes de empezar 476
Paso 1: Crear un conector 476
Paso 2: Configurar las políticas de Microsoft Active Directory 476
Paso 3: Crea una plantilla 478
Paso 4: Configurar los permisos de grupo de Microsoft 479
Conectores para Active Directory 479
Crea conector 479
Crear plantilla 482
Actualizar la plantilla 486
Enumera conectores 487
Enumerar plantillas 488

Ver el conector ... 489
Ver plantilla ... 491
Registros de directorios ... 493
Plantillas de entradas de control de acceso ... 495
Nombre de la entidad principal del servicio ... 497
Etiquetas ... 498
Integrating EventBridge with ... 499
¿Cómo direcciona los eventos de EventBridge Connector for AD ... 499
Conector para eventos de AD ... 500
Creación de patrones de eventos ... 500
Recibir eventos ... 501
Solucione los problemas de Connector para Active Directory ... 501
Conector para códigos de error de AD ... 502
Fallo en la creación del conector ... 507
Error de creación de SPN ... 512
Problemas de actualización de plantillas ... 513
Conector para SCEP ... 515
Características ... 515
¿Cómo empezar a utilizar Connector for SCEP ... 516
Servicios relacionados ... 516
Conector de acceso para SCEP ... 516
Precios ... 517
Conceptos ... 517
Consideraciones y limitaciones ... 519
Consideraciones ... 519
Limitaciones ... 520
Configuración ... 520
Paso 1: Crear una política AWS Identity and Access Management ... 521
Paso 2: Crear una CA privada ... 522
Paso 3: Crear un recurso compartido ... 523
Introducción ... 524
Antes de empezar ... 524
Paso 1: Crear un conector ... 525
Paso 2: Copie los detalles del conector en su sistema MDM ... 527
Configure su sistema MDM ... 527
Conector de uso general ... 528

AWS Private CA Conector para SCEP para Microsoft Intune 529
Configura Jamf Pro 530
Configurar Microsoft Intune 537
Configure Omnissa Workspace ONE 540
Seguridad 546
Puntos de enlace de la VPC (AWS PrivateLink) 547
Supervisión 551
Automatice usando EventBridge 552
CloudTrail registros 557
Solución de problemas 566
Errores de HTTP 567
Errores del cliente 589
Service Quotas 591
Historial de revisiones del documento 593
Actualizaciones anteriores 603
..... dciv

¿Qué es Autoridad de certificación privada de AWS?

Autoridad de certificación privada de AWS permite la creación de jerarquías de autoridades de certificación (CA) privadas, incluidas las principales y las subordinadas CAs, sin los costes de inversión y mantenimiento que implica operar una CA local. Su empresa privada CAs puede emitir certificados X.509 de entidad final, útiles en situaciones como las siguientes:

- Creación de canales de comunicación TLS cifrados
- Autenticación de usuarios, equipos, puntos de conexión de API y dispositivos IoT
- Firmar código criptográficamente
- Implementación del Protocolo de estado de certificados en línea (OCSP) para obtener el estado de revocación de certificados

Autoridad de certificación privada de AWS se puede acceder a las operaciones desde Consola de administración de AWS, mediante la Autoridad de certificación privada de AWS API o mediante la. AWS CLI

Temas

- Disponibilidad regional para AWS Private Certificate Authority
- Servicios integrados con AWS Private Certificate Authority
- Algoritmos criptográficos compatibles en AWS Private Certificate Authority
- Conformidad con la RFC 5280 en AWS Private Certificate Authority
- Precios para AWS Private Certificate Authority
- Términos y conceptos para AWS Private CA

Disponibilidad regional para AWS Private Certificate Authority

Como la mayoría de AWS los recursos, las autoridades de certificación privadas (CAs) son recursos regionales. Para usar el CAs modo privado en más de una región, debe crear el suyo CAs en esas regiones. No puedes copiar archivos privados CAs entre regiones. Visite Regiones y puntos de conexión de AWS en la Referencia general de AWS o la Tabla de regiones de AWS para ver la disponibilidad regional de Autoridad de certificación privada de AWS.

> **Note**
>
> El ACM está disponible actualmente en algunas regiones, pero no lo Autoridad de certificación privada de AWS está.

Servicios integrados con AWS Private Certificate Authority

Si AWS Certificate Manager solía solicitar un certificado privado, puede asociar ese certificado a cualquier servicio que esté integrado con ACM. Esto se aplica tanto a los certificados encadenados a una Autoridad de certificación privada de AWS raíz como a los certificados encadenados a una raíz externa. Para obtener más información, consulte Servicios integrados en la Guía del AWS Certificate Manager usuario.

También puede integrar el sector privado CAs en Amazon Elastic Kubernetes Service para emitir certificados dentro de un clúster de Kubernetes. Para obtener más información, consulte Proteja Kubernetes con AWS Private Certificate Authority.

> **Note**
>
> Amazon Elastic Kubernetes Service no es un servicio integrado de ACM.

Si utiliza la Autoridad de certificación privada de AWS API AWS CLI para emitir un certificado o exportar un certificado privado desde ACM, puede instalar el certificado en cualquier lugar que desee.

Algoritmos criptográficos compatibles en AWS Private Certificate Authority

Autoridad de certificación privada de AWS admite los siguientes algoritmos criptográficos para la generación de claves privadas y la firma de certificados.

Algoritmo compatible

Algoritmos de clave privada	Algoritmos de firma
ML_DSA_44	ML_DSA_44

Algoritmos de clave privada	Algoritmos de firma
ML_DSA_65	ML_DSA_65
ML_DSA_87	ML_DSA_87
RSA_2048	SHA256CON RSA
RSA_3072	SHA384CON RSA
RSA_4096	SHA512CON RSA
EC_prime256v1	SHA256CON ECDSA
EC_secp384r1	SHA384CON ECDSA
EC_SECP521R1	SHA512CON ECDSA
SM2 (Solo regiones de China)	SM3WITHSM2

Esta lista se aplica únicamente a los certificados emitidos directamente Autoridad de certificación privada de AWS a través de su consola, API o línea de comandos. Cuando AWS Certificate Manager emite certificados mediante una CA desde Autoridad de certificación privada de AWS, es compatible con algunos de estos algoritmos, pero no con todos. Para obtener más información, consulte Solicitar un certificado privado en la Guía del AWS Certificate Manager usuario.

> **Note**
>
> En el caso de RSA o ECDSA, la familia de algoritmos de firma especificada debe coincidir con la familia de algoritmos clave de la clave privada de la CA.
> En el caso del ML-DSA, la función hash se define como parte del propio algoritmo. No existe la opción de seleccionar una función hash diferente con ML-DSA. Para mantener la compatibilidad con versiones anteriores APIs, se utiliza el mismo valor para el algoritmo clave y el algoritmo de firma.

Conformidad con la RFC 5280 en AWS Private Certificate Authority

Autoridad de certificación privada de AWS no impone ciertas restricciones definidas en el RFC 5280. La situación inversa también es cierta: se aplican determinadas restricciones adicionales apropiadas para una entidad de certificación privada.

Forzada

- No después de la fecha. Conforme a RFC 5280, Autoridad de certificación privada de AWS impide la emisión de certificados con una fecha `Not After` posterior a la fecha `Not After` del certificado de entidad de certificación.
- Restricciones básicas. Autoridad de certificación privada de AWS impone las restricciones básicas y la longitud de ruta en los certificados de CA importados.

  Las restricciones básicas indican si el recurso identificado por el certificado es una entidad de certificación y puede emitir certificados. Los certificados de entidades de certificación importados a Autoridad de certificación privada de AWS deben incluir la extensión de restricciones básicas y la extensión debe estar marcada `critical`. Además de la `critical` bandera, `CA=true` debe estar configurada. Autoridad de certificación privada de AWS impone restricciones básicas al fallar con una excepción de validación por los siguientes motivos:

  - La extensión no se incluye en el certificado de entidad de certificación.
  - La extensión no está marcada `critical`.

  La longitud de la ruta (pathLenConstraint) determina cuántos subordinados CAs pueden existir aguas abajo del certificado de CA importado. Autoridad de certificación privada de AWS impone la longitud de la ruta al fallar con una excepción de validación por los siguientes motivos:

  - La importación de un certificado de entidad de certificación violaría la restricción de longitud de ruta en el certificado de entidad de certificación o en cualquier certificado de entidad de certificación de la cadena.
  - La emisión de un certificado violaría una restricción de longitud de ruta.
- Las restricciones de nombres indican un espacio de nombres dentro del cual deben ubicarse todos los nombres de los sujetos de los certificados subsiguientes de una ruta de certificación. Se aplican restricciones al nombre distintivo del sujeto y a los nombres alternativos del sujeto.

No se aplica

- Políticas de certificación. Las políticas de certificados regulan las condiciones en las que una entidad emisora de certificados emite certificados.
- Inhibe cualquier política. Se utiliza en los certificados emitidos a. CAs
- Nombre alternativo del emisor. Permite asociar identidades adicionales al emisor del certificado de CA.
- Restricciones políticas. Estas restricciones limitan la capacidad de una entidad de certificación para emitir certificados de entidad de certificación subordinada.
- Mapeos de políticas. Se utiliza en los certificados de CA. Muestra uno o más pares de OIDs; cada par incluye un issuerDomainPolicy y unsubjectDomainPolicy.
- Atributos del directorio de temas. Se utiliza para transmitir los atributos de identificación del sujeto.
- Acceso a la información del sujeto. Cómo acceder a la información y los servicios relacionados con el asunto del certificado en el que aparece la extensión.
- Identificador de clave de sujeto (SKI) e Identificador de clave de autoridad (AKI). El RFC requiere un certificado de entidad de certificación para contener la extensión SKI. Los certificados emitidos por la CA deben contener una extensión de AKI que coincida con el SKI del certificado de CA. AWS no hace cumplir estos requisitos. Si su certificado de entidad de certificación no contiene un SKI, la entidad final emitida o el certificado de entidad de certificación subordinada será el hash SHA-1 de la clave pública del emisor.
- SubjectPublicKeyInfoy el nombre alternativo del sujeto (SAN). Al emitir un certificado, Autoridad de certificación privada de AWS copia las extensiones del SAN SubjectPublicKeyInfo y de la CSR proporcionada sin realizar la validación.

Precios para AWS Private Certificate Authority

A su cuenta se le cobra un precio mensual por cada entidad de certificación privada a partir del momento en que la crea. También se le cobrará cada certificado que emita. Este cargo incluye los certificados que exporte desde ACM y los certificados que cree desde la Autoridad de certificación privada de AWS API o la Autoridad de certificación privada de AWS CLI. Una vez que se elimina una CA privada, no se le aplicarán cargos por ella. Sin embargo, si restaura una entidad de certificación privada, se le cobrará por el tiempo que ha pasado entre su eliminación y su restauración. Los certificados privados en los que no tiene acceso a la clave privada son gratis. Estos incluyen los certificados que se utilizan con servicios integrados, CloudFront como Elastic Load Balancing y API Gateway.

Para obtener la información Autoridad de certificación privada de AWS de precios más reciente, consulte AWS Private Certificate Authority Precios. También puede utilizar la calculadora de precios de AWS para estimar los costos.

Términos y conceptos para AWS Private CA

Los siguientes términos y conceptos pueden ayudarle a medida que trabaja con ellos AWS Private Certificate Authority.

Temas

- Confianza
- Certificados de servidor TLS
- Firma del certificado
- Certificate authority (Autoridad de certificado)
- CA raíz
- Certificado de entidad de certificación
- Certificado de entidad de certificación raíz
- Certificado de entidad final
- Certificados autofirmados
- Certificado privado
- Ruta de acceso del certificado
- Restricción de longitud de ruta

Confianza

Para que un navegador web confíe en la identidad de un sitio web, el navegador debe tener la posibilidad de verificar el certificado del sitio web. Los navegadores, sin embargo, solo confían en una pequeña cantidad de certificados conocidos como certificados raíz de la CA. Una tercera parte de confianza, conocida como entidad de certificación (CA), valida la identidad del sitio web y emite un certificado digital firmado para el operador del sitio web. El navegador puede comprobar la firma digital para validar la identidad del sitio web. Si la validación se realiza correctamente, el navegador muestra un icono de un candado en la barra de direcciones.

Certificados de servidor TLS

Las transacciones HTTPS requieren certificados de servidor para autenticar un servidor. Un certificado de servidor es una estructura de datos X.509 v3 que vincula la clave pública del certificado al asunto del certificado. Un certificado TLS lo firma una entidad de certificación (CA). Contiene el nombre del servidor, el período de validez, la clave pública, el algoritmo de firma y más.

Firma del certificado

Una firma digital es un hash cifrado a través de un certificado. La firma se utiliza para confirmar la integridad de los datos del certificado. Su entidad de certificación privada crea una firma mediante una función de hash criptográfico, por ejemplo, SHA256 sobre el contenido del certificado de tamaño variable. Esta función hash produce una cadena de datos de tamaño fijo que no se puede falsificar de forma efectiva. Esta cadena se denomina hash. El CA cifra después este valor hash con la clave privada y concatena los hash cifrados con el certificado.

Certificate authority (Autoridad de certificado)

Una entidad de certificación (CA) emite certificados digitales y, si es necesario, los revoca. El tipo más común de certificado digital se basa en el estándar ISO X.509. Un certificado X.509 confirma la identidad del sujeto del certificado y asocia esa identidad a una clave pública. El sujeto puede ser un usuario, una aplicación, un equipo o cualquier otro dispositivo. La CA firma un certificado aplicando una función hash al contenido y cifrando después el hash con la clave privada que está asociada a la clave pública del certificado. Una aplicación cliente (por ejemplo, un navegador web) que necesite confirmar la identidad de un sujeto utilizará la clave pública para descifrar la firma del certificado. A continuación, aplicará una función hash al contenido del certificado y comparará el valor hash con la firma descifrada para determinar si coinciden. Para obtener más información sobre la firma de solicitudes, consulte Firma del certificado.

Puede utilizarla Autoridad de certificación privada de AWS para crear una CA privada y utilizarla para emitir certificados. Su CA privada solo emite SSL/TLS certificados privados para su uso en su organización. Para obtener más información, consulte Certificado privado. Para que pueda utilizarse, la CA privada también necesita un certificado. Para obtener más información, consulte Certificado de entidad de certificación.

CA raíz

Un bloque de creación criptográfico y raíz de confianza en función de la cual se pueden emitir certificado. Se compone de una clave privada para firmar (emitir) certificados y un certificado

de raíz que identifica la entidad de certificación raíz y enlaza la clave privada al nombre de la entidad de certificación. El certificado raíz se distribuye a los almacenes de confianza de cada entidad de un entorno. Los administradores crean almacenes de confianza para incluir solo aquellos en los CAs que confían. Los administradores actualizan o crean los almacenes de confianza en los sistemas operativos, las instancias y las imágenes de las máquinas host de las entidades de su entorno. Cuando los recursos intentan conectarse entre sí, verifican los certificados que presenta cada entidad. Un cliente comprueba la validez de los certificados y si existe una cadena desde el certificado hasta un certificado raíz instalado en el almacén de confianza. Si se cumplen esas condiciones, se produce un “apretón de manos” entre los recursos. Este apretón de manos demuestra criptográficamente la identidad de cada entidad con la otra y crea un canal de comunicación cifrado (TLS/SSL) entre ellas.

Certificado de entidad de certificación

El certificado de una entidad de certificación (CA) confirma la identidad de la CA y la asocia con la clave pública incluida en el certificado.

Puede utilizarlos Autoridad de certificación privada de AWS para crear una CA raíz privada o una CA subordinada privada, cada una respaldada por un certificado de CA. Los certificados de entidad de certificación subordinada están firmados por otro certificado de entidad de certificación superior en una cadena de confianza. Pero en el caso de una entidad de certificación raíz, el certificado está autofirmado. También puede establecer una autoridad raíz externa (alojada en las instalaciones, por ejemplo). A continuación, puede utilizar su autoridad raíz para firmar un certificado de entidad de certificación raíz subordinada alojado por Autoridad de certificación privada de AWS.

En el siguiente ejemplo, se muestran los campos típicos de un certificado de CA Autoridad de certificación privada de AWS X.509. Tenga en cuenta que, en los certificados de CA, el valor `CA:` del campo `Basic Constraints` está establecido en `TRUE`.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4121 (0x1019)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=Washington, L=Seattle, O=Example Company Root CA, OU=Corp,
 CN=www.example.com/emailAddress=corp@www.example.com
        Validity
            Not Before: Feb 26 20:27:56 2018 GMT
            Not After : Feb 24 20:27:56 2028 GMT
```

```
        Subject: C=US, ST=WA, L=Seattle, O=Examples Company Subordinate CA,
OU=Corporate Office, CN=www.example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c0: ... a3:4a:51
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                F8:84:EE:37:21:F2:5E:0B:6C:40:C2:9D:C6:FE:7E:49:53:67:34:D9
            X509v3 Authority Key Identifier:
                keyid:0D:CE:76:F2:E3:3B:93:2D:36:05:41:41:16:36:C8:82:BC:CB:F8:A0

            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Digital Signature, CRL Sign
    Signature Algorithm: sha256WithRSAEncryption
         6:bb:94: ... 80:d8
```

Certificado de entidad de certificación raíz

Por lo general, una autoridad de certificación (CA) existe dentro de una estructura jerárquica que contiene varias otras, CAs con relaciones padre-hijo claramente definidas entre ellas. El hijo o el subordinado CAs reciben la certificación de sus padres CAs, lo que crea una cadena de certificados. La CA de la parte superior de la jerarquía se denomina “raíz de la CA” y su certificado se denomina “certificado raíz”. Este certificado suele estar autofirmado.

Certificado de entidad final

Un certificado de identidad final identifica un recurso, como un servidor, instancia, contenedor o dispositivo. A diferencia de los certificados de entidades de certificación, los certificados de entidad final no se pueden usar para emitir certificados. Otros términos comunes para el certificado de entidad final son certificado «cliente» o «hoja».

Certificados autofirmados

Un certificado firmado por el emisor en lugar de una entidad de certificación superior. A diferencia de los certificados emitidos desde una raíz segura mantenida por una CA, los certificados autofirmados funcionan como su propia raíz y, por lo tanto, tienen limitaciones importantes: se pueden utilizar para

proporcionar cifrado electrónico, pero no para verificar la identidad, y no se pueden revocar. Son inaceptables desde el punto de vista de la seguridad. Sin embargo, las organizaciones los utilizan porque son fáciles de generar, no requieren experiencia ni infraestructura, y muchas aplicaciones los aceptan. No existen controles para la emisión de certificados autofirmados. Las organizaciones que los utilizan corren un mayor riesgo de sufrir interrupciones causadas por la caducidad de los certificados porque no tienen forma de hacer un seguimiento de las fechas de caducidad.

Certificado privado

Autoridad de certificación privada de AWS los certificados son SSL/TLS certificados privados que puede usar en su organización, pero que no son de confianza en la Internet pública. Utilícelos para identificar recursos, como clientes, servidores, aplicaciones, servicios, dispositivos y usuarios. Al establecer un canal de comunicación cifrado y seguro, cada recurso utiliza un certificado como el siguiente junto con técnicas de cifrado para demostrar su identidad a otro recurso Los puntos de conexión de la API interna, los servidores web, los usuarios de VPN, los dispositivos de IoT y muchas otras aplicaciones utilizan certificados privados para establecer los canales de comunicación cifrados que son necesarios para poder trabajar con seguridad. De forma predeterminada, los certificados privados no son de confianza pública. Un administrador interno debe configurar explícitamente las aplicaciones para que confíen en los certificados privados y distribuyan los certificados.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            e8:cb:d2:be:db:12:23:29:f9:77:06:bc:fe:c9:90:f8
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=WA, L=Seattle, O=Example Company CA, OU=Corporate,
 CN=www.example.com
        Validity
            Not Before: Feb 26 18:39:57 2018 GMT
            Not After : Feb 26 19:39:57 2019 GMT
        Subject: C=US, ST=Washington, L=Seattle, O=Example Company, OU=Sales,
 CN=www.example.com/emailAddress=sales@example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00...c7
                Exponent: 65537 (0x10001)
```

```
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Authority Key Identifier:
                keyid:AA:6E:C1:8A:EC:2F:8F:21:BC:BE:80:3D:C5:65:93:79:99:E7:71:65

            X509v3 Subject Key Identifier:
                C6:6B:3C:6F:0A:49:9E:CC:4B:80:B2:8A:AB:81:22:AB:89:A8:DA:19
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://NA/crl/12345678-1234-1234-1234-123456789012.crl

    Signature Algorithm: sha256WithRSAEncryption
         58:32:...:53
```

Ruta de acceso del certificado

Un cliente que se basa en un certificado valida que existe una ruta de acceso desde el certificado de entidad final, posiblemente a través de una cadena de certificados intermedios, a una raíz de confianza. El cliente comprueba que cada certificado de la ruta es válido (no está revocado). También comprueba que el certificado de la entidad final no haya caducado, que sea íntegro (no haya sido manipulado ni modificado) y que se cumplan las restricciones del certificado.

Restricción de longitud de ruta

Las restricciones básicas pathLenConstraintde un certificado de CA establecen el número de certificados de CA subordinados que pueden existir en la cadena inferior. Por ejemplo, un certificado de CA con una restricción de longitud de ruta igual a cero no puede tener ningún subordinado. CAs Una CA con una restricción de longitud de ruta igual a uno puede tener hasta un nivel de subordinado CAs por debajo. El RFC 5280 lo define como «el número máximo de certificados non-self-issued intermedios que pueden seguir a este certificado en una ruta de certificación válida». El valor de longitud de ruta excluye el certificado de la entidad final, aunque puede incluirse en un lenguaje informal sobre la “longitud” o la “profundidad” de una cadena de validación, lo que genera confusión.

¿Cuál es el mejor servicio de certificados para mis necesidades?

Hay dos AWS servicios para emitir e implementar certificados X.509. Elija el que mejor se adapte a sus necesidades. Las consideraciones incluyen si necesita certificados públicos o privados, certificados personalizados, certificados que desee implementar en otros AWS servicios o una administración y renovación automatizadas de los certificados.

1. Autoridad de certificación privada de AWS: este servicio es para clientes empresariales que estén creando una infraestructura de clave pública (PKI) dentro de la nube de AWS destinada a uso privado en una organización. Con él Autoridad de certificación privada de AWS, puede crear su propia jerarquía de entidades de certificación y emitir certificados con ella para autenticar a los usuarios internos, los ordenadores, las aplicaciones, los servicios, los servidores y otros dispositivos, así como para firmar el código de los equipos. Los certificados emitidos por una entidad de certificación privada solo son de confianza dentro de su organización, no en Internet.

   Después de crear una entidad de certificación privada, puede emitir certificados directamente (es decir, sin obtener la validación de una entidad de certificación de terceros) y personalizarlos para satisfacer las necesidades internas de su organización. Por ejemplo, es posible que desee:

   - Crear certificados con cualquier nombre de sujeto.
   - Crear certificados con cualquier fecha de caducidad.
   - Utilizar cualquier algoritmo de clave privada y cualquier longitud de clave que sean compatibles.
   - Utilizar cualquier algoritmo de firma compatible.
   - Controlar la emisión de certificados mediante plantillas.

   Está en el lugar adecuado para este servicio. Para empezar, inicie sesión en la consola de https://console.aws.amazon.com/acm-pca/.
2. AWS Certificate Manager (ACM): este servicio administra los certificados para clientes empresariales que necesitan una presencia web segura y de confianza pública mediante TLS. Puede implementar certificados ACM en AWS Elastic Load Balancing, Amazon CloudFront, Amazon API Gateway y otros servicios integrados. La aplicación más frecuente de este tipo es un sitio web público seguro con importantes requisitos de tráfico.

   Puede utilizar los certificados públicos que proporciona ACM (certificados de ACM) o los certificados que importe a ACM. Si lo utiliza Autoridad de certificación privada de AWS para crear

una CA, ACM puede gestionar la emisión de certificados desde esa CA privada y automatizar las renovaciones de los certificados.

Para obtener más información, consulte la Guía del usuario de AWS Certificate Manager.

Autoridad de certificación privada de AWS mejores prácticas

Las mejores prácticas son recomendaciones que pueden ayudarle a utilizarlas de Autoridad de certificación privada de AWS forma eficaz. Las siguientes prácticas recomendadas se basan en la experiencia real de clientes actuales AWS Certificate Manager y de Autoridad de certificación privada de AWS clientes.

Documentación de la estructura y las políticas de entidad de certificación (CA)

AWS recomienda documentar todas sus políticas y prácticas para operar su CA. Esto podría incluir:

- Razonamiento de sus decisiones sobre la estructura de la entidad de certificación
- Un diagrama que muestre su relación con CAs la de ellos
- Políticas sobre períodos de validez de la entidad de certificación
- Planificación de la sucesión de la entidad de certificación
- Políticas sobre la longitud de la ruta
- Catálogo de permisos
- Descripción de las estructuras de control administrativo
- Seguridad

Puede capturar esta información en dos documentos, conocidos como Política de certificación (CP) y Declaración de prácticas de certificación (CPS). Consulte RFC 3647 para obtener un marco para capturar información importante sobre las operaciones de entidades de certificación.

Minimizar el uso de la entidad de certificación (CA) raíz si es posible

Por lo general, una CA raíz solo debe usarse para emitir certificados para empresas intermedias CAs. Esto permite almacenar la CA raíz de forma segura, mientras que la entidad emisora intermedia CAs realiza la tarea diaria de emitir los certificados de la entidad final.

Sin embargo, si la práctica actual de su organización es emitir certificados de entidad final directamente desde una CA raíz, Autoridad de certificación privada de AWS puede respaldar este

flujo de trabajo y, al mismo tiempo, mejorar los controles operativos y de seguridad. La emisión de certificados de entidad final en este escenario requiere una política de permisos de IAM que permita a la entidad de certificación raíz utilizar una plantilla de certificado de entidad final. Para obtener información acerca de las políticas de IAM, consulte Identity and Access Management (IAM) para AWS Private Certificate Authority.

> **Note**
>
> Esta configuración impone limitaciones que pueden dar lugar a desafíos operativos. Por ejemplo, si la entidad de certificación raíz se ha puesto en riesgo o se pierde, debe crear una entidad de certificación raíz nueva y distribuirla a todos los clientes del entorno. Hasta que se complete este proceso de recuperación, no podrá emitir certificados nuevos. La emisión de certificados directamente desde una entidad de certificación raíz también impide restringir el acceso y limitar el número de certificados emitidos desde la raíz, que se consideran prácticas recomendadas para administrar una entidad de certificación raíz.

Dele los suyos a la CA raíz Cuenta de AWS

Se recomienda crear una CA raíz y una CA subordinada en dos AWS cuentas diferentes. Si lo hace, puede proporcionarle protección adicional y controles de acceso para su entidad de certificación raíz. Puede hacerlo exportando la CSR desde la entidad de certificación subordinada en una cuenta y firmarla con una entidad de certificación raíz en otra cuenta. La ventaja de este enfoque es que puedes separar el control de tu cuenta CAs por cuenta. La desventaja es que no se puede utilizar el Consola de administración de AWS asistente para simplificar el proceso de firma del certificado de CA de una CA subordinada desde la CA raíz.

> **Important**
>
> Recomendamos encarecidamente el uso de la autenticación multifactor (MFA) cada vez que acceda. Autoridad de certificación privada de AWS

Funciones separadas de administrador y emisor

La función de administrador de CA debe estar separada de la de los usuarios, que solo necesitan emitir certificados de entidad final. Si el administrador de CA y el emisor del certificado residen en

el mismo lugar Cuenta de AWS, puede limitar los permisos del emisor creando un usuario de IAM específico para ese fin.

Implementar la revocación gestionada de certificados

La revocación gestionada notifica automáticamente a los clientes de certificados cuando se ha revocado un certificado. Es posible que deba revocar un certificado si su información criptográfica se ha visto comprometida o si se emitió por error. Los clientes suelen negarse a aceptar los certificados revocados. Autoridad de certificación privada de AWS ofrece dos opciones estándar para la revocación gestionada: el Protocolo de estado de certificados en línea (OCSP) y las listas de revocación de certificados (). CRLs Para obtener más información, consulte Planifique el método de revocación AWS Private CA de su certificado.

Encienda AWS CloudTrail

Active el CloudTrail registro antes de crear y empezar a operar una CA privada. Con CloudTrail ella, puede recuperar un historial de llamadas a la AWS API de su cuenta para supervisar sus AWS despliegues. Este historial incluye las llamadas a la API realizadas desde los servicios de nivel superior Consola de administración de AWS AWS SDKs, los servicios y los AWS Command Line Interface servicios de nivel superior AWS . También puede identificar qué usuarios y cuentas llamaron a las operaciones de la API de PCA, la dirección IP de origen desde la que se realizaron las llamadas y el momento en que se efectuaron. Puede CloudTrail integrarse en las aplicaciones mediante la API, automatizar la creación de rutas para su organización, comprobar el estado de las rutas y controlar la forma en que los administradores activan y desactivan el CloudTrail inicio de sesión. Para obtener más información, consulte Crear un registro de seguimiento. Visite para Registro de llamadas a la AWS Private Certificate Authority API mediante AWS CloudTrail ver ejemplos de senderos para Autoridad de certificación privada de AWS operaciones.

Rotar la clave privada de la CA

Es recomendable actualizar periódicamente la clave privada de la CA privada. Puede actualizar una clave importando un nuevo certificado de entidad de certificación o puede reemplazar la entidad de certificación privada por una nueva entidad de certificación.

> **Note**
>
> Si reemplaza la propia CA, tenga en cuenta que el ARN de la CA cambia. Esto provocaría un error en la automatización que se basa en un ARN con codificación rígida.

Eliminar los no utilizados CAs

Puede eliminar permanentemente una CA privada. Tal vez quiera hacerlo si ya no la necesita o si desea reemplazarla por otra que tenga una nueva clave privada. Para eliminar de forma segura una CA, le recomendamos que siga el proceso que se describe en Eliminación de una CA privada.

> **Note**
>
> AWS le factura una CA hasta que la elimine.

Bloquee el acceso público a su CRLs

Autoridad de certificación privada de AWS recomienda utilizar la función Amazon S3 Block Public Access (BPA) en los buckets que contengan. CRLs Esto evita exponer innecesariamente los detalles de su PKI privada a posibles adversarios. El BPA es una de las mejores prácticas de S3 y está activado de forma predeterminada en los buckets nuevos. En algunos casos, se necesita una configuración adicional. Para obtener más información, consulte Habilite S3 Block Public Access (BPA) con CloudFront.

Prácticas recomendadas de aplicaciones de Amazon EKS

Cuando utilice certificados X.509 Autoridad de certificación privada de AWS para aprovisionar Amazon EKS, siga las recomendaciones para proteger los entornos multiusuario de las guías de prácticas recomendadas de Amazon EKS. Para obtener información general sobre la integración Autoridad de certificación privada de AWS con Kubernetes, consulte Proteja Kubernetes con AWS Private Certificate Authority.

AWS Private CA Úselo con el AWS SDK para Java

Puedes usar la AWS Private Certificate Authority API para interactuar con el servicio mediante programación mediante el envío de solicitudes HTTP. El servicio devuelve respuestas HTTP. Para obtener más información, consulte la Referencia de la API de AWS Private Certificate Authority.

Además de la API HTTP, puedes usar las herramientas AWS SDKs y las herramientas de línea de comandos para interactuar con ellas. Autoridad de certificación privada de AWS Es preferible utilizar este mecanismo a la API de HTTP. Para obtener más información, consulte Herramientas para Amazon Web Services. En los temas siguientes se muestra cómo utilizar AWS SDK para Java para programar la API de Autoridad de certificación privada de AWS .

El GetCertificateAuthorityCsrGetCertificate, y DescribeCertificateAuthorityAuditReportlas operaciones apoyan a los camareros. Puede usar las listas de espera para controlar la progresión del código en función de la presencia o el estado de determinados recursos. Para obtener más información, consulte los siguientes temas, así como Waiters AWS SDK para Java en el blog para AWS desarrolladores.

AWS Private CA Ejemplos de API

Los siguientes ejemplos de código muestran cómo utilizar determinadas acciones y tipos de datos de la AWS Private CA API con AWS SDK para Java.

Temas

- Crear y activar una CA raíz mediante programación
- Crear y activar una CA subordinada mediante programación
- CreateCertificateAuthority
- Utilización CreateCertificateAuthority para dar soporte a Active Directory
- CreateCertificateAuthorityAuditReport
- CreatePermission
- DeleteCertificateAuthority
- DeletePermission
- DeletePolicy
- DescribeCertificateAuthority
- DescribeCertificateAuthorityAuditReport

- GetCertificate
- GetCertificateAuthorityCertificate
- GetCertificateAuthorityCsr
- GetPolicy
- ImportCertificateAuthorityCertificate
- IssueCertificate
- ListCertificateAuthorities
- ListPermissions
- ListTags
- PutPolicy
- RestoreCertificateAuthority
- RevokeCertificate
- TagCertificateAuthorities
- UntagCertificateAuthority
- UpdateCertificateAuthority
- Cree CAs y certifique con nombres de asunto personalizados
- Cree certificados con extensiones personalizadas

Crear y activar una CA raíz mediante programación

En este ejemplo de Java, se muestra cómo activar una CA raíz mediante las siguientes acciones de la Autoridad de certificación privada de AWS API:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
```

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.samples.GetCertificateAuthorityCertificate;
import com.amazonaws.auth.AWSStaticCredentialsProvider;


import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Objects;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
```

```
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

public class RootCAActivation {
    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = "region"; // Substitute your region here, e.g. "us-
west-2"

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setOrganization("Example Organization");
        subject.setOrganizationalUnit("Example");
        subject.setCountry("US");
        subject.setState("Virginia");
        subject.setLocality("Arlington");
        subject.setCommonName("www.example.com");

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
 CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.RSA_2048);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);
        configCA.withSubject(subject);

        // Define a certificate revocation list configuration.
        CrlConfiguration crlConfigure = new CrlConfiguration();
        crlConfigure.withEnabled(true);
        crlConfigure.withExpirationInDays(365);
        crlConfigure.withCustomCname(null);
        crlConfigure.withS3BucketName("your-bucket-name");

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.ROOT;
```

```
        // ** Execute core code samples for Root CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCAArn = CreateCertificateAuthority(configCA, crlConfigure, CAtype,
client);
        String csr = GetCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = IssueCertificate(rootCAArn, csr, client);
        String rootCertificate = GetCertificate(rootCertificateArn, rootCAArn, client);
        ImportCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

    private static AWSACMPCA ClientBuilder(String endpointRegion) {
        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load the credentials from the credential profiles file. " +
                    "Please make sure that your credentials file is at the correct " +
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                    e);
        }

        String endpointProtocol = "https://acm-pca." + endpointRegion +
".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        return client;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CrlConfiguration crlConfigure, CertificateAuthorityType CAtype, AWSACMPCA
client) {
```

```
        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);

        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withRevocationConfiguration(revokeConfig);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);

        // Create the private CA.
        CreateCertificateAuthorityResult createCAResult = null;
        try {
            createCAResult = client.createCertificateAuthority(createCARequest);
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

        // Retrieve the ARN of the private CA.
        String rootCAArn = createCAResult.getCertificateAuthorityArn();
        System.out.println("Root CA Arn: " + rootCAArn);

        return rootCAArn;
    }

    private static String GetCertificateAuthorityCsr(String rootCAArn, AWSACMPCA
client) {

        // Create the CSR request object and set the CA ARN.
        GetCertificateAuthorityCsrRequest csrRequest = new
GetCertificateAuthorityCsrRequest();
        csrRequest.withCertificateAuthorityArn(rootCAArn);

        // Create waiter to wait on successful creation of the CSR file.
        Waiter<GetCertificateAuthorityCsrRequest> getCSRWaiter =
client.waiters().certificateAuthorityCSRCreated();
        try {
            getCSRWaiter.run(new WaiterParameters<>(csrRequest));
        } catch (WaiterUnrecoverableException e) {
```

```
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        // Retrieve and display the CSR;
        String csr = csrResult.getCsr();
        System.out.println(csr);

        return csr;
    }

    private static String IssueCertificate(String rootCAArn, String csr, AWSACMPCA
 client) {

        // Create a certificate request:
        IssueCertificateRequest issueRequest = new IssueCertificateRequest();

        // Set the CA ARN.
        issueRequest.withCertificateAuthorityArn(rootCAArn);

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/RootCACertificate/
V1");

        ByteBuffer csrByteBuffer = stringToByteBuffer(csr);
        issueRequest.setCsr(csrByteBuffer);
```

```
        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(3650L);
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

        // Set the idempotency token.
        issueRequest.setIdempotencyToken("1234");

        // Issue the certificate.
        IssueCertificateResult issueResult = null;
        try {
            issueResult = client.issueCertificate(issueRequest);
        } catch (LimitExceededException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Retrieve and display the certificate ARN.
        String rootCertificateArn = issueResult.getCertificateArn();
        System.out.println("Root Certificate Arn: " + rootCertificateArn);

        return rootCertificateArn;
    }

    private static String GetCertificate(String rootCertificateArn, String rootCAArn,
AWSACMPCA client) {

        // Create a request object.
        GetCertificateRequest certificateRequest = new GetCertificateRequest();
```

```
        // Set the certificate ARN.
        certificateRequest.withCertificateArn(rootCertificateArn);

        // Set the certificate authority ARN.
        certificateRequest.withCertificateAuthorityArn(rootCAArn);

        // Create waiter to wait on successful creation of the certificate file.
        Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
        try {
            getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the certificate and certificate chain.
        GetCertificateResult certificateResult = null;
        try {
            certificateResult = client.getCertificate(certificateRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

        // Get the certificate and certificate chain and display the result.
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
    }
```

```
    private static void ImportCertificateAuthorityCertificate(String rootCertificate,
String rootCAArn, AWSACMPCA client) {

        // Create the request object and set the signed certificate, chain and CA ARN.
        ImportCertificateAuthorityCertificateRequest importRequest =
            new ImportCertificateAuthorityCertificateRequest();

        ByteBuffer certByteBuffer = stringToByteBuffer(rootCertificate);
        importRequest.setCertificate(certByteBuffer);

        importRequest.setCertificateChain(null);

        // Set the certificate authority ARN.
        importRequest.withCertificateAuthorityArn(rootCAArn);

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Root CA certificate successfully imported.");
        System.out.println("Root CA activated successfully.");
    }

    private static ByteBuffer stringToByteBuffer(final String string) {
        if (Objects.isNull(string)) {
            return null;
        }
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
        return ByteBuffer.wrap(bytes);
```

```
    }
}
```

Crear y activar una CA subordinada mediante programación

En este ejemplo de Java, se muestra cómo activar una CA subordinada mediante las siguientes acciones de la Autoridad de certificación privada de AWS API:

- GetCertificateAuthorityCertificate
- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;


import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
```

```
import java.util.Objects;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateResult;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

public class SubordinateCAActivation {

    public static void main(String[] args) throws Exception {
        // Place your own Root CA ARN here.
        String rootCAArn = "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566";

        // Define the endpoint region for your sample.
```

```
        String endpointRegion = "region"; // Substitute your region here, e.g. "us-
west-2"

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setOrganization("Example Organization");
        subject.setOrganizationalUnit("Example");
        subject.setCountry("US");
        subject.setState("Virginia");
        subject.setLocality("Arlington");
        subject.setCommonName("www.example.com");

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
 CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.RSA_2048);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);
        configCA.withSubject(subject);

        // Define a certificate revocation list configuration.
        CrlConfiguration crlConfigure = new CrlConfiguration();
        crlConfigure.withEnabled(true);
        crlConfigure.withExpirationInDays(365);
        crlConfigure.withCustomCname(null);
        crlConfigure.withS3BucketName("your-bucket-name");

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.SUBORDINATE;

        // ** Execute core code samples for Subordinate CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCertificate = GetCertificateAuthorityCertificate(rootCAArn, client);
        String subordinateCAArn = CreateCertificateAuthority(configCA, crlConfigure,
 CAtype, client);
        String csr = GetCertificateAuthorityCsr(subordinateCAArn, client);
        String subordinateCertificateArn = IssueCertificate(rootCAArn, csr, client);
        String subordinateCertificate = GetCertificate(subordinateCertificateArn,
 rootCAArn, client);
        ImportCertificateAuthorityCertificate(subordinateCertificate, rootCertificate,
 subordinateCAArn, client);

    }

    private static AWSACMPCA ClientBuilder(String endpointRegion) {
```

```
        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load the credentials from the credential profiles file. " +
                    "Please make sure that your credentials file is at the correct " +
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                    e);
        }

        String endpointProtocol = "https://acm-pca." + endpointRegion +
".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        return client;
    }

    private static String GetCertificateAuthorityCertificate(String rootCAArn,
AWSACMPCA client) {
        // ** GetCertificateAuthorityCertificate **

        // Create a request object and set the certificate authority ARN,
        GetCertificateAuthorityCertificateRequest getCACertificateRequest =
        new GetCertificateAuthorityCertificateRequest();
        getCACertificateRequest.withCertificateAuthorityArn(rootCAArn);

        // Create a result object.
        GetCertificateAuthorityCertificateResult getCACertificateResult = null;
        try {
            getCACertificateResult =
client.getCertificateAuthorityCertificate(getCACertificateRequest);
        } catch (ResourceNotFoundException ex) {
```

```
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        }

        // Retrieve and display the certificate information.
        String rootCertificate = getCACertificateResult.getCertificate();
        System.out.println("Root CA Certificate / Certificate Chain:");
        System.out.println(rootCertificate);

        return rootCertificate;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CrlConfiguration crlConfigure, CertificateAuthorityType CAtype, AWSACMPCA
client) {
        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);

        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withRevocationConfiguration(revokeConfig);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);

        // Create the private CA.
        CreateCertificateAuthorityResult createCAResult = null;
        try {
            createCAResult = client.createCertificateAuthority(createCARequest);
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

        // Retrieve the ARN of the private CA.
        String subordinateCAArn = createCAResult.getCertificateAuthorityArn();
        System.out.println("Subordinate CA Arn: " + subordinateCAArn);
```

```
        return subordinateCAArn;
    }

    private static String GetCertificateAuthorityCsr(String subordinateCAArn, AWSACMPCA
client) {

        // Create the CSR request object and set the CA ARN.
        GetCertificateAuthorityCsrRequest csrRequest = new
GetCertificateAuthorityCsrRequest();
        csrRequest.withCertificateAuthorityArn(subordinateCAArn);

        // Create waiter to wait on successful creation of the CSR file.
        Waiter<GetCertificateAuthorityCsrRequest> getCSRWaiter =
client.waiters().certificateAuthorityCSRCreated();
        try {
            getCSRWaiter.run(new WaiterParameters<>(csrRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch(AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        // Retrieve and display the CSR;
        String csr = csrResult.getCsr();
        System.out.println("Subordinate CSR:");
        System.out.println(csr);
```

```
        return csr;
    }

    private static String IssueCertificate(String rootCAArn, String csr, AWSACMPCA
 client) {

        // Create a certificate request:
        IssueCertificateRequest issueRequest = new IssueCertificateRequest();

        // Set the issuing CA ARN.
        issueRequest.withCertificateAuthorityArn(rootCAArn);

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/
SubordinateCACertificate_PathLen0/V1");

        ByteBuffer csrByteBuffer = stringToByteBuffer(csr);
        issueRequest.setCsr(csrByteBuffer);

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(730L); // Approximately two years
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

        // Set the idempotency token.
        issueRequest.setIdempotencyToken("1234");

        // Issue the certificate.
        IssueCertificateResult issueResult = null;
        try {
            issueResult = client.issueCertificate(issueRequest);
        } catch (LimitExceededException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
```

```
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Retrieve and display the certificate ARN.
        String subordinateCertificateArn = issueResult.getCertificateArn();
        System.out.println("Subordinate Certificate Arn: " +
subordinateCertificateArn);

        return subordinateCertificateArn;
    }

    private static String GetCertificate(String subordinateCertificateArn, String
rootCAArn, AWSACMPCA client) {

        // Create a request object.
        GetCertificateRequest certificateRequest = new GetCertificateRequest();

        // Set the certificate ARN.
        certificateRequest.withCertificateArn(subordinateCertificateArn);

        // Set the certificate authority ARN.
        certificateRequest.withCertificateAuthorityArn(rootCAArn);

        // Create waiter to wait on successful creation of the certificate file.
        Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
        try {
            getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the certificate and certificate chain.
        GetCertificateResult certificateResult = null;
        try {
            certificateResult = client.getCertificate(certificateRequest);
```

```
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

        // Get the certificate and certificate chain and display the result.
        String subordinateCertificate = certificateResult.getCertificate();
        System.out.println("Subordinate CA Certificate:");
        System.out.println(subordinateCertificate);

        return subordinateCertificate;
    }

    private static void ImportCertificateAuthorityCertificate(String
subordinateCertificate, String rootCertificate, String subordinateCAArn, AWSACMPCA
client) {

        // Create the request object and set the signed certificate, chain and CA ARN.
        ImportCertificateAuthorityCertificateRequest importRequest =
            new ImportCertificateAuthorityCertificateRequest();

        ByteBuffer certByteBuffer = stringToByteBuffer(subordinateCertificate);
        importRequest.setCertificate(certByteBuffer);

        ByteBuffer rootCACertByteBuffer = stringToByteBuffer(rootCertificate);
        importRequest.setCertificateChain(rootCACertByteBuffer);

        // Set the certificate authority ARN.
        importRequest.withCertificateAuthorityArn(subordinateCAArn);

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
```

```
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
        System.out.println("Subordinate CA certificate successfully imported.");
        System.out.println("Subordinate CA activated successfully.");
    }

    private static ByteBuffer stringToByteBuffer(final String string) {
        if (Objects.isNull(string)) {
            return null;
        }
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
        return ByteBuffer.wrap(bytes);
    }
}
```

CreateCertificateAuthority

El siguiente ejemplo de Java muestra cómo utilizar la CreateCerticateAuthorityoperación.

La operación crea una entidad de certificación (CA) subordinada privada. Debe especificar la configuración de la CA, la configuración de revocación, el tipo de CA y un token de idempotencia opcional.

La configuración de la CA determina lo siguiente:

- El nombre del algoritmo y el tamaño de la clave que se utilizará para crear la clave privada de la entidad de certificación
- El tipo de algoritmo de firma que la CA utiliza para firmar su propia solicitud de firma de certificados y las CRLs respuestas del OCSP
- La información del sujeto de X.500

La configuración de la CRL determina lo siguiente:

- El periodo de caducidad de la CRL en días (periodo de validez de la CRL)
- El bucket de Amazon S3 que contendrá la CRL
- Un alias CNAME para el bucket de S3 que se va a incluir en los certificados de la CA

Si se ejecuta correctamente, la función devuelve el nombre de recurso de Amazon (ARN) de la CA.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;


import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.util.ArrayList;
import java.util.Objects;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;


public class CreateCertificateAuthority {

    public static void main(String[] args) throws Exception {
```

```
        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                   "Cannot load the credentials from the credential profiles file. " +
                   "Please make sure that your credentials file is at the correct " +
                   "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
 format.",
                   e);
        }

        // Define the endpoint for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
        String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setOrganization("Example Organization");
        subject.setOrganizationalUnit("Example");
        subject.setCountry("US");
        subject.setState("Virginia");
        subject.setLocality("Arlington");
        subject.setCommonName("www.example.com");

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
 CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.RSA_2048);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);
        configCA.withSubject(subject);
```

```
        // Define a certificate revocation list configuration.
        CrlConfiguration crlConfigure = new CrlConfiguration();
        crlConfigure.withEnabled(true);
        crlConfigure.withExpirationInDays(365);
        crlConfigure.withCustomCname(null);
        crlConfigure.withS3BucketName("your-bucket-name");

        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);

        // Define a certificate authority type: ROOT or SUBORDINATE
        CertificateAuthorityType CAtype = CertificateAuthorityType.<<SUBORDINATE>>;

        // Create a tag - method 1
        Tag tag1 = new Tag();
        tag1.withKey("PrivateCA");
        tag1.withValue("Sample");

        // Create a tag - method 2
        Tag tag2 = new Tag()
            .withKey("Purpose")
            .withValue("WebServices");

        // Add the tags to a collection.
        ArrayList<Tag> tags = new ArrayList<Tag>();
        tags.add(tag1);
        tags.add(tag2);

        // Create the request object.
        CreateCertificateAuthorityRequest req = new
CreateCertificateAuthorityRequest();
        req.withCertificateAuthorityConfiguration(configCA);
        req.withRevocationConfiguration(revokeConfig);
        req.withIdempotencyToken("123987");
        req.withCertificateAuthorityType(CAtype);
        req.withTags(tags);


        // Create the private CA.
        CreateCertificateAuthorityResult result = null;
        try {
            result = client.createCertificateAuthority(req);
        } catch (InvalidArgsException ex) {
```

```
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

        // Retrieve the ARN of the private CA.
        String arn = result.getCertificateAuthorityArn();
        System.out.println(arn);
    }
}
```

El resultado debería ser similar al siguiente:

```
arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
```

Utilización CreateCertificateAuthority para dar soporte a Active Directory

El siguiente ejemplo de Java muestra cómo utilizar la CreateCerticateAuthorityoperación para crear una CA que se pueda instalar en el NTAuth almacén empresarial de Microsoft Active Directory (AD).

La operación crea una entidad emisora de certificados (CA) raíz privada mediante identificadores de objeto personalizados (OIDs). Para obtener más información y un AWS CLI ejemplo de una operación equivalente, consulte Crear una CA para iniciar sesión en Active Directory.

Si se ejecuta correctamente, la función devuelve el nombre de recurso de Amazon (ARN) de la CA.

```
package com.amazonaws.samples.appstream;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.samples.GetCertificateAuthorityCertificate;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
```

```
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
```

```
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import org.bouncycastle.asn1.x509.SubjectPublicKeyInfo;
import org.bouncycastle.cert.jcajce.JcaX509ExtensionUtils;
import org.bouncycastle.openssl.PEMParser;
import org.bouncycastle.pkcs.PKCS10CertificationRequest;
import org.bouncycastle.util.io.pem.PemReader;

import lombok.SneakyThrows;

public class RootCAActivation {
    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "ap-
southeast-2"

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("2.5.4.3") // OID for Common Name
                .withValue("root CA"),
            new CustomAttribute()
                .withObjectIdentifier("0.9.2342.19200300.100.1.25") // OID for Domain
 Component
                .withValue("example"),
            new CustomAttribute()
                .withObjectIdentifier("0.9.2342.19200300.100.1.25") // OID for Domain
 Component
                .withValue("com")

        );

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);

        // Define the CA configuration.
```

```
        CertificateAuthorityConfiguration configCA = new
CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.EC_prime256v1);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
        configCA.withSubject(subject);

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.ROOT;

        // ** Execute core code samples for Root CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCAArn = CreateCertificateAuthority(configCA, CAtype, client);
        String csr = GetCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = IssueCertificate(rootCAArn, csr, client);
        String rootCertificate = GetCertificate(rootCertificateArn, rootCAArn, client);
        ImportCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

    private static AWSACMPCA ClientBuilder(String endpointRegion) {
        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load the credentials from the credential profiles file. " +
                    "Please make sure that your credentials file is at the correct " +
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                    e);
        }

        String endpointProtocol = "https://acm-pca." + endpointRegion +
".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();
```

```
        return client;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CertificateAuthorityType CAtype, AWSACMPCA client) {
        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);

        // Create the private CA.
        CreateCertificateAuthorityResult createCAResult = null;
        try {
            createCAResult = client.createCertificateAuthority(createCARequest);
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

        // Retrieve the ARN of the private CA.
        String rootCAArn = createCAResult.getCertificateAuthorityArn();
        System.out.println("Root CA Arn: " + rootCAArn);

        return rootCAArn;
    }

    private static String GetCertificateAuthorityCsr(String rootCAArn, AWSACMPCA
client) {

        // Create the CSR request object and set the CA ARN.
        GetCertificateAuthorityCsrRequest csrRequest = new
GetCertificateAuthorityCsrRequest();
        csrRequest.withCertificateAuthorityArn(rootCAArn);

        // Create waiter to wait on successful creation of the CSR file.
        Waiter<GetCertificateAuthorityCsrRequest> getCSRWaiter =
client.waiters().certificateAuthorityCSRCreated();
        try {
```

```
            getCSRWaiter.run(new WaiterParameters<>(csrRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        // Retrieve and display the CSR;
        String csr = csrResult.getCsr();
        System.out.println(csr);

        return csr;
    }

    private static String IssueCertificate(String rootCAArn, String csr, AWSACMPCA
 client) {

        // Create a certificate request:
        IssueCertificateRequest issueRequest = new IssueCertificateRequest();

        // Set the CA ARN.
        issueRequest.withCertificateAuthorityArn(rootCAArn);

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/RootCACertificate/
V1");
```

```
        ByteBuffer csrByteBuffer = stringToByteBuffer(csr);
        issueRequest.setCsr(csrByteBuffer);

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(3650L);
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

        // Set the idempotency token.
        issueRequest.setIdempotencyToken("1234");

        // Issue the certificate.
        IssueCertificateResult issueResult = null;
        try {
            issueResult = client.issueCertificate(issueRequest);
        } catch (LimitExceededException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Retrieve and display the certificate ARN.
        String rootCertificateArn = issueResult.getCertificateArn();
        System.out.println("Root Certificate Arn: " + rootCertificateArn);

        return rootCertificateArn;
    }

    private static String GetCertificate(String rootCertificateArn, String rootCAArn,
AWSACMPCA client) {

        // Create a request object.
```

```
        GetCertificateRequest certificateRequest = new GetCertificateRequest();

        // Set the certificate ARN.
        certificateRequest.withCertificateArn(rootCertificateArn);

        // Set the certificate authority ARN.
        certificateRequest.withCertificateAuthorityArn(rootCAArn);

        // Create waiter to wait on successful creation of the certificate file.
        Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
        try {
            getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the certificate and certificate chain.
        GetCertificateResult certificateResult = null;
        try {
            certificateResult = client.getCertificate(certificateRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

        // Get the certificate and certificate chain and display the result.
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
    }
```

```
   private static void ImportCertificateAuthorityCertificate(String rootCertificate,
String rootCAArn, AWSACMPCA client) {

        // Create the request object and set the signed certificate, chain and CA ARN.
        ImportCertificateAuthorityCertificateRequest importRequest =
            new ImportCertificateAuthorityCertificateRequest();

        ByteBuffer certByteBuffer = stringToByteBuffer(rootCertificate);
        importRequest.setCertificate(certByteBuffer);

        importRequest.setCertificateChain(null);

        // Set the certificate authority ARN.
        importRequest.withCertificateAuthorityArn(rootCAArn);

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Root CA certificate successfully imported.");
        System.out.println("Root CA activated successfully.");
    }

    private static ByteBuffer stringToByteBuffer(final String string) {
        if (Objects.isNull(string)) {
            return null;
        }
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
```

```
        return ByteBuffer.wrap(bytes);
    }
}
```

El resultado debería ser similar al siguiente:

```
arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
```

CreateCertificateAuthorityAuditReport

En el siguiente ejemplo de Java, se muestra cómo utilizar la CreateCertificateAuthorityAuditReportoperación.

La operación crea un informe de auditoría que aparece cada vez que se emite o revoca un certificado. El informe se guarda en el bucket de Amazon S3 que se especifique en la entrada. Puede generar un nuevo informe cada 30 minutos.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import
 com.amazonaws.services.acmpca.model.CreateCertificateAuthorityAuditReportRequest;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityAuditReportResult;

import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidStateException;

public class CreateCertificateAuthorityAuditReport {
```

```
   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
         credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
         throw new AmazonClientException("Cannot load your credentials from file.", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
         .build();

      // Create a request object and set the certificate authority ARN.
      CreateCertificateAuthorityAuditReportRequest req =
          new CreateCertificateAuthorityAuditReportRequest();

      // Set the certificate authority ARN.
      req.setCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Specify the S3 bucket name for your report.
      req.setS3BucketName("your-bucket-name");

      // Specify the audit response format.
      req.setAuditReportResponseFormat("JSON");

      // Create a result object.
      CreateCertificateAuthorityAuditReportResult result = null;
      try {
```

```
            result = client.createCertificateAuthorityAuditReport(req);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

        String ID = result.getAuditReportId();
        String S3Key = result.getS3Key();

        System.out.println(ID);
        System.out.println(S3Key);

    }
}
```

El resultado debería ser similar al siguiente:

```
58904752-7de3-4bdf-ba89-6953e48c3cc7
audit-report/16075838-061c-4f7a-b54b-49bbc111bcff/58904752-7de3-4bdf-
ba89-6953e48c3cc7.json
```

CreatePermission

En el siguiente ejemplo de Java, se muestra cómo utilizar la CreatePermissionoperación.

La operación asigna permisos de acceso desde una CA privada a un principal de AWS servicio designado. Se puede conceder permiso a los servicios para crear y recuperar certificados de una entidad de certificación privada, así como mostrar los permisos activos que la entidad de certificación privada ha concedido. Para renovar automáticamente los certificados a través de ACM, debe asignar todos los permisos posibles (`IssueCertificate`GetCertificate, y`ListPermissions`) de la CA al principal de servicio de ACM (). `acm.amazonaws.com` Para encontrar el ARN de una CA, llame a la ListCertificateAuthoritiesfunción.

Una vez creado un permiso, puede inspeccionarlo con la ListPermissionsfunción o eliminarlo con la DeletePermissionfunción.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.CreatePermissionRequest;
import com.amazonaws.services.acmpca.model.CreatePermissionResult;

import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.PermissionAlreadyExistsException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;

import java.util.ArrayList;

public class CreatePermission {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
         credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
         throw new AmazonClientException("Cannot load your credentials from file.", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
```

```
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
         .build();

      // Create a request object.
      CreatePermissionRequest req =
          new CreatePermissionRequest();

      //  Set the certificate authority ARN.
      req.setCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Set the permissions to give the user.
      ArrayList<String> permissions = new ArrayList<>();
      permissions.add("IssueCertificate");
      permissions.add("GetCertificate");
      permissions.add("ListPermissions");

      req.setActions(permissions);

      // Set the Principal.
      req.setPrincipal("acm.amazonaws.com");

      // Create a result object.
      CreatePermissionResult result = null;
      try {
         result = client.createPermission(req);
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (LimitExceededException ex) {
         throw ex;
      } catch (PermissionAlreadyExistsException ex) {
         throw ex;
      } catch (RequestFailedException ex) {
```

```
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        }
    }
}
```

DeleteCertificateAuthority

El siguiente ejemplo de Java muestra cómo utilizar la DeleteCertificateAuthorityoperación.

Esta operación elimina la entidad de certificación (CA) privada que creó mediante la CreateCertificateAuthorityoperación. La operación `DeleteCertificateAuthority` requiere que proporcione un ARN para que se elimine la entidad de certificación. Puede encontrar el ARN llamando a la ListCertificateAuthoritiesoperación. Si la CA privada tiene el estado `CREATING` o `PENDING_CERTIFICATE`, puede eliminarse inmediatamente. Sin embargo, si ya se ha importado el certificado, no se puede eliminar de forma inmediata. Primero debe deshabilitar la CA llamando a la UpdateCertificateAuthorityoperación y configurando el `Status` parámetro en. `DISABLED` A continuación, puede utilizar el parámetro `PermanentDeletionTimeInDays` en la operación `DeleteCertificateAuthority` para especificar el número de días, de 7 a 30. Durante dicho período, la entidad de certificación privada se puede restaurar al estado `disabled`. De forma predeterminada, si no se establece el parámetro `PermanentDeletionTimeInDays`, el periodo de restauración es de 30 días. Después de este período, la CA privada se elimina de forma permanente y no se puede restaurar. Para obtener más información, consulte Restauración de una CA.

Para ver un ejemplo de Java que muestra cómo utilizar la RestoreCertificateAuthorityoperación, consulteRestoreCertificateAuthority.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.DeleteCertificateAuthorityRequest;
```

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.RequestFailedException;

public class DeleteCertificateAuthority {

   public static void main(String[] args) throws Exception{

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
         credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
         throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
         .build();

      // Create a requrest object and set the ARN of the private CA to delete.
      DeleteCertificateAuthorityRequest req = new DeleteCertificateAuthorityRequest();

      // Set the certificate authority ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Set the recovery period.
      req.withPermanentDeletionTimeInDays(12);
```

```
      // Delete the CA.
      try {
         client.deleteCertificateAuthority(req);
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (RequestFailedException ex) {
         throw ex;
      }
   }
}
```

DeletePermission

El siguiente ejemplo de Java muestra cómo utilizar la DeletePermissionoperación.

La operación elimina los permisos que una entidad emisora de certificados privada ha delegado a una entidad de AWS servicio mediante la CreatePermissionsoperación. Para encontrar el ARN de una CA, llame a la ListCertificateAuthoritiesfunción. Puede inspeccionar los permisos otorgados por una CA llamando a la ListPermissionsfunción.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.DeletePermissionRequest;
import com.amazonaws.services.acmpca.model.DeletePermissionResult;

import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
```

```
public class DeletePermission {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
         credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
         throw new AmazonClientException("Cannot load your credentials from file.", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
         .build();

      // Create a request object.
      DeletePermissionRequest req =
          new DeletePermissionRequest();

      //  Set the certificate authority ARN.
      req.setCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Set the AWS service principal.
      req.setPrincipal("acm.amazonaws.com");

      // Create a result object.
      DeletePermissionResult result = null;
      try {
         result = client.deletePermission(req);
```

```
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (RequestFailedException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      }
   }
}
```

DeletePolicy

El siguiente ejemplo de Java muestra cómo utilizar la DeletePolicyoperación.

La operación elimina la política basada en recursos asociada a una CA privada. Se utiliza una política basada en recursos para permitir el uso compartido de CA entre cuentas. Puede encontrar el ARN de una CA privada convocando la ListCertificateAuthoritiesacción.

Las acciones de la API relacionadas incluyen PutPolicyy. GetPolicy

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.DeletePolicyRequest;
import com.amazonaws.services.acmpca.model.DeletePolicyResult;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.LockoutPreventedException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
```

```
public class DeletePolicy {

    public static void main(String[] args) throws Exception {

        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load your credentials from file.",
 e);
        }

        // Define the endpoint for your sample.
        String endpointRegion = "us-west-2";  // Substitute your Region here, e.g. "us-
west-2"
        String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        // Create the request object.
        DeletePolicyRequest req = new DeletePolicyRequest();

        // Set the resource ARN.
        req.withResourceArn("arn:aws:acm-pca:us-west-2:111122223333:certificate-
authority/11223344-44ee-aa22-bb33-4cd2d13f1f18");

        // Retrieve a list of your CAs.
        DeletePolicyResult result = null;
        try {
            result = client.deletePolicy(req);
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
```

```
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (LockoutPreventedException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (AWSACMPCAException ex) {
            throw ex;
        }
    }
}
```

DescribeCertificateAuthority

El siguiente ejemplo de Java muestra cómo utilizar la DescribeCertificateAuthorityoperación.

Esta operación muestra información sobre la entidad de certificación (CA) privada. Debe especificar el nombre de recurso de Amazon (ARN) de la CA privada. El resultado contiene el estado de la CA. Puede ser uno de los siguientes:

- `CREATING`— Autoridad de certificación privada de AWS está creando su entidad de certificación privada.
- `PENDING_CERTIFICATE`: el certificado está pendiente. Debe utilizar la CA raíz o subordinada en las instalaciones para firmar la CSR de la CA privada e importarla en PCA.
- `ACTIVE`: la CA privada está activa.
- `DISABLED`: la CA privada se ha deshabilitado.
- `EXPIRED`: el certificado de la CA privada ha caducado.
- `FAILED`: no se puede crear la CA privada.
- `DELETED`: la CA privada se encuentra en el periodo de restauración. Una vez transcurrido este periodo, se eliminará permanentemente.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
```

```
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.CertificateAuthority;
import com.amazonaws.services.acmpca.model.DescribeCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.DescribeCertificateAuthorityResult;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArnException;

public class DescribeCertificateAuthority {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
         credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
         throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
         .build();

      // Create a request object
```

```
      DescribeCertificateAuthorityRequest req = new
 DescribeCertificateAuthorityRequest();

      // Set the certificate authority ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Create a result object.
      DescribeCertificateAuthorityResult result = null;
      try {
         result = client.describeCertificateAuthority(req);
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      }

      // Retrieve and display information about the CA.
      CertificateAuthority PCA = result.getCertificateAuthority();
      String strPCA = PCA.toString();
      System.out.println(strPCA);
   }
}
```

DescribeCertificateAuthorityAuditReport

En el siguiente ejemplo de Java, se muestra cómo utilizar la DescribeCertificateAuthorityAuditReportoperación.

La operación muestra información sobre un informe de auditoría específico que creó al llamar a la CreateCertificateAuthorityAuditReportoperación. Los datos de auditoría se crean cada vez que se usa la clave privada de la entidad de certificación (CA). La clave privada se utiliza cuando se emite un certificado, se firma una CRL o se revoca un certificado.

```
package com.amazonaws.samples;

import java.util.Date;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
```

```
import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import
 com.amazonaws.services.acmpca.model.DescribeCertificateAuthorityAuditReportRequest;
import
 com.amazonaws.services.acmpca.model.DescribeCertificateAuthorityAuditReportResult;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

public class DescribeCertificateAuthorityAuditReport {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
         credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
         throw new AmazonClientException("Cannot load your credentials from file.", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
```

```
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
         .build();

      // Create a request object.
      DescribeCertificateAuthorityAuditReportRequest req =
            new DescribeCertificateAuthorityAuditReportRequest();

      // Set the certificate authority ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Set the audit report ID.
      req.withAuditReportId("11111111-2222-3333-4444-555555555555");

      // Create waiter to wait on successful creation of the audit report file.
      Waiter<DescribeCertificateAuthorityAuditReportRequest> waiter =
 client.waiters().auditReportCreated();
      try {
         waiter.run(new WaiterParameters<>(req));
      } catch (WaiterUnrecoverableException e) {
          //Explicit short circuit when the recourse transitions into
          //an undesired state.
      } catch (WaiterTimedOutException e) {
          //Failed to transition into desired state even after polling.
      } catch (AWSACMPCAException e) {
          //Unexpected service exception.
      }

      // Create a result object.
      DescribeCertificateAuthorityAuditReportResult result = null;
      try {
         result = client.describeCertificateAuthorityAuditReport(req);
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidArgsException ex) {
         throw ex;
      }

      String status = result.getAuditReportStatus();
      String S3Bucket = result.getS3BucketName();
      String S3Key = result.getS3Key();
      Date createdAt = result.getCreatedAt();

      System.out.println(status);
```

```
        System.out.println(S3Bucket);
        System.out.println(S3Key);
        System.out.println(createdAt);
    }
}
```

El resultado debería ser similar al siguiente:

```
SUCCESS
your-audit-report-bucket-name
audit-report/a4119411-8153-498a-a607-2cb77b858043/25211c3d-f2fe-479f-b437-
fe2b3612bc45.json
Tue Jan 16 13:07:58 PST 2018
```

GetCertificate

En el siguiente ejemplo de Java, se muestra cómo utilizar la GetCertificateoperación.

La operación recupera un certificado de la entidad de certificación privada. El ARN del certificado se devuelve cuando se llama a la IssueCertificateoperación. Debe especificar tanto el ARN de la entidad de certificación privada como el ARN del certificado emitido al llamar a la operación `GetCertificate`. El certificado se puede recuperar si tiene el estado `ISSUED`. Puede llamar a la CreateCertificateAuthorityAuditReportoperación para crear un informe que contenga información sobre todos los certificados emitidos y revocados por su entidad emisora de certificados privada.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.RequestFailedException ;
```

```
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import com.amazonaws.services.acmpca.model.AWSACMPCAException;

public class GetCertificate {

   public static void main(String[] args) throws Exception{

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
         credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
         throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

      // Create a client.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
         .build();

      // Create a request object.
      GetCertificateRequest req = new GetCertificateRequest();

      // Set the certificate ARN.
```

```
      req.withCertificateArn("arn:aws:acm-pca:region:account:certificate-
authority/CA_ID/certificate/certificate_ID");

      // Set the certificate authority ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Create waiter to wait on successful creation of the certificate file.
      Waiter<GetCertificateRequest> waiter = client.waiters().certificateIssued();
      try {
         waiter.run(new WaiterParameters<>(req));
      } catch (WaiterUnrecoverableException e) {
          //Explicit short circuit when the recourse transitions into
          //an undesired state.
      } catch (WaiterTimedOutException e) {
          //Failed to transition into desired state even after polling.
      } catch (AWSACMPCAException e) {
          //Unexpected service exception.
      }

      // Retrieve the certificate and certificate chain.
      GetCertificateResult result = null;
      try {
         result = client.getCertificate(req);
      } catch (RequestInProgressException ex) {
         throw ex;
      } catch (RequestFailedException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      }

      // Get the certificate and certificate chain and display the result.
      String strCert = result.getCertificate();
      System.out.println(strCert);
   }
}
```

El resultado debería ser una cadena de certificados similar a la siguiente con la entidad de certificación (CA) y el certificado especificados.

```
-----BEGIN CERTIFICATE----- base64-encoded certificate -----END CERTIFICATE-----

-----BEGIN CERTIFICATE----- base64-encoded certificate -----END CERTIFICATE-----

-----BEGIN CERTIFICATE----- base64-encoded certificate -----END CERTIFICATE-----
```

GetCertificateAuthorityCertificate

En el siguiente ejemplo de Java, se muestra cómo utilizar la GetCertificateAuthorityCertificateoperación.

Esta operación recupera el certificado y la cadena de certificados de la entidad de certificación (CA) privada. Tanto el certificado como la cadena están codificadas en base64 con formato PEM. La cadena no incluye el certificado de CA. Cada uno de los certificados de la cadena firma el certificado anterior a él.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateResult;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidArnException;

public class GetCertificateAuthorityCertificate {

   public static void main(String[] args) throws Exception {
```

```
      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
         credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
         throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

      // Create a request object
      GetCertificateAuthorityCertificateRequest req =
            new GetCertificateAuthorityCertificateRequest();

      // Set the certificate authority ARN,
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Create a result object.
      GetCertificateAuthorityCertificateResult result = null;
      try {
         result = client.getCertificateAuthorityCertificate(req);
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      }
```

```
        // Retrieve and display the certificate information.
        String strPcaCert = result.getCertificate();
        System.out.println(strPcaCert);
        String strPCACChain = result.getCertificateChain();
        System.out.println(strPCACChain);
    }
}
```

El resultado debería ser un certificado y una cadena similares a los siguientes con la entidad de certificación (CA) especificada.

```
-----BEGIN CERTIFICATE----- base64-encoded certificate -----END CERTIFICATE-----

-----BEGIN CERTIFICATE----- base64-encoded certificate -----END CERTIFICATE-----
```

GetCertificateAuthorityCsr

El siguiente ejemplo de Java muestra cómo utilizar la GetCertificateAuthorityCsroperación.

Esta operación recupera la solicitud de firma de certificado (CSR) de la entidad de certificación (CA) privada. La CSR se crea al llamar a la CreateCertificateAuthorityoperación. Transfiere la CSR a la infraestructura X.509 en las instalaciones y fírmela utilizando la CA raíz o una CA subordinada. A continuación, vuelva a importar el certificado firmado a ACM PCA mediante una llamada a la operación. ImportCertificateAuthorityCertificate La CSR que se devuelve es una cadena codificada en base64 en el formato PEM.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
```

```
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

public class GetCertificateAuthorityCsr {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
         credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
         throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
         .build();

      // Create the request object and set the CA ARN.
      GetCertificateAuthorityCsrRequest req = new GetCertificateAuthorityCsrRequest();
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Create waiter to wait on successful creation of the CSR file.
```

```
      Waiter<GetCertificateAuthorityCsrRequest> waiter =
 client.waiters().certificateAuthorityCSRCreated();
      try {
         waiter.run(new WaiterParameters<>(req));
      } catch (WaiterUnrecoverableException e) {
          //Explicit short circuit when the recourse transitions into
          //an undesired state.
      } catch (WaiterTimedOutException e) {
          //Failed to transition into desired state even after polling.
      } catch (AWSACMPCAException e) {
          //Unexpected service exception.
      }

      // Retrieve the CSR.
      GetCertificateAuthorityCsrResult result = null;
      try {
         result = client.getCertificateAuthorityCsr(req);
      } catch (RequestInProgressException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (RequestFailedException ex) {
         throw ex;
      }

      // Retrieve and display the CSR;
      String Csr = result.getCsr();
      System.out.println(Csr);
   }
}
```

El resultado debería ser similar al siguiente con la entidad de certificación (CA) que especifique. La solicitud de firma de certificado (CSR) está codificada en base64 con el formato PEM. Guárdela en un archivo local, transfiérala a la infraestructura en las instalaciones de X.509 y fírmela utilizando la entidad de certificación raíz o una entidad de certificación subordinada.

```
-----BEGIN CERTIFICATE REQUEST----- base64-encoded request -----END CERTIFICATE
 REQUEST-----
```

GetPolicy

El siguiente ejemplo de Java muestra cómo utilizar la GetPolicyoperación.

La operación recupera la política basada en recursos asociada a una CA privada. Se utiliza una política basada en recursos para permitir el uso compartido de CA entre cuentas. Puede encontrar el ARN de una CA privada convocando la ListCertificateAuthoritiesacción.

Las acciones de la API relacionadas incluyen PutPolicyy. DeletePolicy

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.GetPolicyRequest;
import com.amazonaws.services.acmpca.model.GetPolicyResult;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;

public class GetPolicy {

    public static void main(String[] args) throws Exception {

        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load your credentials from file.",
 e);
        }
```

```
        // Define the endpoint for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
        String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        // Create the request object.
        GetPolicyRequest req = new GetPolicyRequest();

        // Set the resource ARN.
        req.withResourceArn("arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566");

        // Retrieve a list of your CAs.
        GetPolicyResult result= null;
        try {
            result = client.getPolicy(req);
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (AWSACMPCAException ex) {
            throw ex;
        }

        // Display the policy.
        System.out.println(result.getPolicy());
    }
}
```

ImportCertificateAuthorityCertificate

En el siguiente ejemplo de Java, se muestra cómo utilizar la ImportCertificateAuthorityCertificateoperación.

Esta operación importa su certificado de CA privado firmado a Autoridad de certificación privada de AWS. Antes de poder llamar a esta operación, debe crear la autoridad de certificación privada mediante la llamada a la CreateCertificateAuthorityoperación. A continuación, debe generar una solicitud de firma de certificado (CSR) llamando a la GetCertificateAuthorityCsroperación. Transfiera la CSR a la CA en las instalaciones y utilice el certificado raíz o un certificado subordinado para firmarla. Cree una cadena de certificados y copie el certificado firmado y la cadena de certificados en el directorio de trabajo.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.RequestFailedException;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Objects;

public class ImportCertificateAuthorityCertificate {
```

```
   public static ByteBuffer stringToByteBuffer(final String string) {
      if (Objects.isNull(string)) {
          return null;
      }
      byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
      return ByteBuffer.wrap(bytes);
      }

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
         .build();

      // Create the request object and set the signed certificate, chain and CA ARN.
      ImportCertificateAuthorityCertificateRequest req =
            new ImportCertificateAuthorityCertificateRequest();

      // Set the signed certificate.
      String strCertificate =
            "-----BEGIN CERTIFICATE-----\n" +
            "base64-encoded certificate\n" +
            "-----END CERTIFICATE-----\n";
      ByteBuffer certByteBuffer = stringToByteBuffer(strCertificate);
      req.setCertificate(certByteBuffer);
```

```
      // Set the certificate chain.
      String strCertificateChain =
            "-----BEGIN CERTIFICATE-----\n" +
            "base64-encoded certificate\n" +
            "-----END CERTIFICATE-----\n";
      ByteBuffer chainByteBuffer = stringToByteBuffer(strCertificateChain);
      req.setCertificateChain(chainByteBuffer);

      // Set the certificate authority ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Import the certificate.
      try {
         client.importCertificateAuthorityCertificate(req);
      } catch (CertificateMismatchException ex) {
         throw ex;
      } catch (MalformedCertificateException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (RequestInProgressException ex) {
         throw ex;
      } catch (ConcurrentModificationException ex) {
         throw ex;
      } catch (RequestFailedException ex) {
         throw ex;
      }
   }
}
```

IssueCertificate

En el siguiente ejemplo de Java, se muestra cómo utilizar la IssueCertificateoperación.

Esta operación utiliza la entidad de certificación privada (Certificate Authority, CA) para emitir un certificado de entidad final. Esta operación devuelve el nombre de recurso de Amazon (ARN) del certificado. Puede recuperar el certificado llamando al ARN GetCertificatey especificándolo.

> **Note**
>
> La IssueCertificateoperación requiere que especifique una plantilla de certificado. En este ejemplo, se utiliza la plantilla `EndEntityCertificate/V1`. Para obtener información sobre las plantillas disponibles, consulte Utilice plantillas de certificados AWS Private CA.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Objects;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;

public class IssueCertificate {
   public static ByteBuffer stringToByteBuffer(final String string) {
      if (Objects.isNull(string)) {
          return null;
      }
      byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
      return ByteBuffer.wrap(bytes);
```

```
      }

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
          .withEndpointConfiguration(endpoint)
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
          .build();

      // Create a certificate request:
      IssueCertificateRequest req = new IssueCertificateRequest();

      // Set the CA ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Specify the certificate signing request (CSR) for the certificate to be signed
 and issued.
      String strCSR =
      "-----BEGIN CERTIFICATE REQUEST-----\n" +
      "base64-encoded certificate\n" +
      "-----END CERTIFICATE REQUEST-----\n";
      ByteBuffer csrByteBuffer = stringToByteBuffer(strCSR);
      req.setCsr(csrByteBuffer);

      // Specify the template for the issued certificate.
```

```
      req.withTemplateArn("arn:aws:acm-pca:::template/EndEntityCertificate/V1");

      // Set the signing algorithm.
      req.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);

      // Set the validity period for the certificate to be issued.
      Validity validity = new Validity();
      validity.withValue(<<3650L>>);
      validity.withType("DAYS");
      req.withValidity(validity);

      // Set the idempotency token.
      req.setIdempotencyToken("1234");

      // Issue the certificate.
      IssueCertificateResult result = null;
      try {
         result = client.issueCertificate(req);
      } catch (LimitExceededException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidArgsException ex) {
         throw ex;
      } catch (MalformedCSRException ex) {
         throw ex;
      }

      // Retrieve and display the certificate ARN.
      String arn = result.getCertificateArn();
      System.out.println(arn);
   }
}
```

El resultado debería ser similar al siguiente:

```
arn:aws:acm-pca:region:account:certificate-authority/CA_ID/certificate/certificate_ID
```

ListCertificateAuthorities

El siguiente ejemplo de Java muestra cómo utilizar la ListCertificateAuthoritiesoperación.

Esta operación muestra las autoridades de certificación privadas (CAs) que creó mediante la CreateCertificateAuthorityoperación.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ListCertificateAuthoritiesRequest;
import com.amazonaws.services.acmpca.model.ListCertificateAuthoritiesResult;
import com.amazonaws.services.acmpca.model.InvalidNextTokenException;

public class ListCertificateAuthorities {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from file.",
 e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
```

```
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        // Create the request object.
        ListCertificateAuthoritiesRequest req = new ListCertificateAuthoritiesRequest();
        req.withMaxResults(10);

        // Retrieve a list of your CAs.
        ListCertificateAuthoritiesResult result= null;
        try {
            result = client.listCertificateAuthorities(req);
        } catch (InvalidNextTokenException ex) {
            throw ex;
        }

        // Display the CA list.
        System.out.println(result.getCertificateAuthorities());
    }
}
```

Si hay entidades de certificación para mostrar, el resultado debería ser similar al siguiente:

```
[{
 Arn: arn: aws: acm-pca: region: account: certificate-
authority/12345678-1234-1234-1234-123456789012,
 CreatedAt: TueNov0712: 05: 39PST2017,
 LastStateChangeAt: WedJan1012: 35: 39PST2018,
 Type: SUBORDINATE,
 Serial: 4109,
 Status: DISABLED,
 NotBefore: TueNov0712: 19: 15PST2017,
 NotAfter: FriNov0513: 19: 15PDT2027,
 CertificateAuthorityConfiguration: {
  KeyType: RSA2048,
  SigningAlgorithm: SHA256WITHRSA,
  Subject: {
   Organization: ExampleCorp,
   OrganizationalUnit: HR,
```

```
    State: Washington,
    CommonName: www.example.com,
    Locality: Seattle,

   }
  },
  RevocationConfiguration: {
   CrlConfiguration: {
    Enabled: true,
    ExpirationInDays: 3650,
    CustomCname: your-custom-name,
    S3BucketName: your-bucket-name
   }
  }
 },
 {
  Arn: arn: aws: acm-pca: region: account>: certificate-
authority/12345678-1234-1234-1234-123456789012,
  CreatedAt: WedSep1312: 54: 52PDT2017,
  LastStateChangeAt: WedSep1312: 54: 52PDT2017,
  Type: SUBORDINATE,
  Serial: 4100,
  Status: ACTIVE,
  NotBefore: WedSep1314: 11: 19PDT2017,
  NotAfter: SatSep1114: 11: 19PDT2027,
  CertificateAuthorityConfiguration: {
   KeyType: RSA2048,
   SigningAlgorithm: SHA256WITHRSA,
   Subject: {
    Country: US,
    Organization: ExampleCompany,
    OrganizationalUnit: Sales,
    State: Washington,
    CommonName: www.example.com,
    Locality: Seattle,

   }
  },
  RevocationConfiguration: {
   CrlConfiguration: {
    Enabled: false,
    ExpirationInDays: 5,
    CustomCname: your-custom-name,
    S3BucketName: your-bucket-name
```

```
  }
 }
},
{
 Arn: arn: aws: acm-pca: region: account>: certificate-
authority/12345678-1234-1234-1234-123456789012,
 CreatedAt: FriJan1213: 57: 11PST2018,
 LastStateChangeAt: FriJan1213: 57: 11PST2018,
 Type: SUBORDINATE,
 Status: PENDING_CERTIFICATE,
 CertificateAuthorityConfiguration: {
  KeyType: RSA2048,
  SigningAlgorithm: SHA256WITHRSA,
  Subject: {
   Country: US,
   Organization: Examples-R-Us Ltd.,
   OrganizationalUnit: corporate,
   State: WA,
   CommonName: www.examplesrus.com,
   Locality: Seattle,

  }
 },
 RevocationConfiguration: {
  CrlConfiguration: {
   Enabled: true,
   ExpirationInDays: 365,
   CustomCname: your-custom-name,
   S3BucketName: your-bucket-name
  }
 }
},
{
 Arn: arn: aws: acm-pca: region: account>: certificate-
authority/12345678-1234-1234-1234-123456789012,
 CreatedAt: FriJan0511: 14: 21PST2018,
 LastStateChangeAt: FriJan0511: 14: 21PST2018,
 Type: SUBORDINATE,
 Serial: 4116,
 Status: ACTIVE,
 NotBefore: FriJan0512: 12: 56PST2018,
 NotAfter: MonJan0312: 12: 56PST2028,
 CertificateAuthorityConfiguration: {
  KeyType: RSA2048,
```

```
  SigningAlgorithm: SHA256WITHRSA,
  Subject: {
   Country: US,
   Organization: ExamplesLLC,
   OrganizationalUnit: CorporateOffice,
   State: WA,
   CommonName: www.example.com,
   Locality: Seattle,

  }
 },
 RevocationConfiguration: {
  CrlConfiguration: {
   Enabled: true,
   ExpirationInDays: 3650,
   CustomCname: your-custom-name,
   S3BucketName: your-bucket-name
  }
 }
}]
```

ListPermissions

El siguiente ejemplo de Java muestra cómo utilizar la ListPermissionsoperación.

Esta operación enumera los permisos, si los hay, que ha asignado la entidad de certificación privada. Los permisos`IssueCertificate`, incluidos los permisos `GetCertificate`y `ListPermissions`, pueden asignarse a un AWS responsable de servicio con la CreatePermissionoperación y revocarse con la DeletePermissionsoperación.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ListPermissionsRequest;
import com.amazonaws.services.acmpca.model.ListPermissionsResult;
```

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidNextTokenException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RequestFailedException;

public class ListPermissions {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
         .withEndpointConfiguration(endpoint)
         .withCredentials(new AWSStaticCredentialsProvider(credentials))
         .build();

      // Create a request object and set the CA ARN.
      ListPermissionsRequest req = new ListPermissionsRequest();
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // List the tags.
      ListPermissionsResult result = null;
      try {
         result = client.listPermissions(req);
```

```
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch(RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        }

        // Retrieve and display the permissions.
        System.out.println(result);
    }
}
```

Si la entidad de certificación privada designada ha asignado permisos a una entidad principal de servicio, el resultado debería ser similar al siguiente:

```
[{
       Arn: arn:aws:acm-
pca:region:account:permission/12345678-1234-1234-1234-123456789012,
       CreatedAt: WedFeb0317: 05: 39PST2019,
       Prinicpal: acm.amazonaws.com,
       Permissions: {
            ISSUE_CERTIFICATE,
            GET_CERTIFICATE,
            DELETE,CERTIFICATE
       },
       SourceAccount: account
}]
```

ListTags

El siguiente ejemplo de Java muestra cómo utilizar la ListTagsoperación.

Esta operación muestra las etiquetas, si las hay, que están asociadas con la entidad de certificación privada. Las etiquetas son etiquetas que puede utilizar para identificar y organizar su CAs. Cada etiqueta consta de una clave y un valor opcional. Llame a la TagCertificateAuthorityoperación para añadir una o más etiquetas a su CA. Llame a la UntagCertificateAuthorityoperación para eliminar las etiquetas.

```
package com.amazonaws.samples;
```

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ListTagsRequest;
import com.amazonaws.services.acmpca.model.ListTagsResult;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArnException;

public class ListTags {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
          .withEndpointConfiguration(endpoint)
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
          .build();
```

```
      // Create a request object and set the CA ARN.
      ListTagsRequest req = new ListTagsRequest();
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // List the tags
      ListTagsResult result = null;
      try {
          result = client.listTags(req);
      } catch (InvalidArnException ex) {
          throw ex;
      } catch (ResourceNotFoundException ex) {
          throw ex;
      }

      // Retrieve and display the tags.
      System.out.println(result);
   }
}
```

Si hay etiquetas para mostrar, el resultado debería ser similar al siguiente:

```
{Tags: [{Key: Admin,Value: Alice}, {Key: Purpose,Value: WebServices}],}
```

PutPolicy

En el siguiente ejemplo de Java, se muestra cómo utilizar la PutPolicyoperación.

La operación se asocia a una política basada en recursos a una CA privada, lo que permite compartir entre cuentas. Cuando una política lo autorice, un titular que resida en otra AWS cuenta puede emitir y renovar certificados privados de entidad final mediante una entidad emisora de certificados privada que no sea de su propiedad. Puede encontrar el ARN de una CA privada convocando la ListCertificateAuthoritiesacción. Para ver ejemplos de políticas, consulte la guía Autoridad de certificación privada de AWS sobre Políticas basadas en recursos.

Una vez que una política esté asociada a una CA, puede inspeccionarla con la GetPolicyacción o eliminarla con la DeletePolicyacción.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
```

```
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.PutPolicyRequest;
import com.amazonaws.services.acmpca.model.PutPolicyResult;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.LockoutPreventedException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class PutPolicy {

    public static void main(String[] args) throws Exception {

        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load your credentials from file.",
 e);
        }

        // Define the endpoint for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
        String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
        EndpointConfiguration endpoint =
```

```
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        // Create the request object.
        PutPolicyRequest req = new PutPolicyRequest();

        // Set the resource ARN.
        req.withResourceArn("arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566");

        // Import and set the policy.
        // Note: This code assumes the file "ShareResourceWithAccountPolicy.json" is in
 a folder titled policy.
        String policy = new String(Files.readAllBytes(Paths.get("policy",
 "ShareResourceWithAccountPolicy.json")));
        req.withPolicy(policy);

        // Retrieve a list of your CAs.
        PutPolicyResult result = null;
        try {
            result = client.putPolicy(req);
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LockoutPreventedException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (AWSACMPCAException ex) {
            throw ex;
        }
```

```
    }
}
```

RestoreCertificateAuthority

En el siguiente ejemplo de Java, se muestra cómo utilizar la RestoreCertificateAuthorityoperación. Las CA privadas se puedan restaurar en cualquier momento durante el periodo de restauración. En la actualidad, este período puede durar entre 7 y 30 días a partir de la fecha de eliminación y puede especificarse al eliminar la CA. Para obtener más información, consulte Restauración de una CA. Consulte también el ejemplo de Java DeleteCertificateAuthority.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.RestoreCertificateAuthorityRequest;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;

public class RestoreCertificateAuthority {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from file.",
 e);
      }
```

```
      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
          .withEndpointConfiguration(endpoint)
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
          .build();

      // Create the request object.
      RestoreCertificateAuthorityRequest req = new
 RestoreCertificateAuthorityRequest();

      // Set the certificate authority ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Restore the CA.
      try {
          client.restoreCertificateAuthority(req);
      } catch (InvalidArnException ex) {
          throw ex;
      } catch (InvalidStateException ex) {
          throw ex;
      } catch (ResourceNotFoundException ex) {
          throw ex;
      }
   }
}
```

RevokeCertificate

En el siguiente ejemplo de Java, se muestra cómo utilizar la RevokeCertificateoperación.

Esta operación revoca un certificado que usted emitió al llamar a la IssueCertificateoperación. Si habilitó una lista de revocación de certificados (CRL) al crear o actualizar su CA privada, la información sobre los certificados revocados se incluye en la CRL. Autoridad de certificación privada

de AWS escribe la CRL en un bucket de Amazon S3 que especifique. Para obtener más información, consulte la CrlConfigurationestructura.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.RevokeCertificateRequest;
import com.amazonaws.services.acmpca.model.RevocationReason;

import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestAlreadyProcessedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;

public class RevokeCertificate {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
```

```
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        // Create a request object.
        RevokeCertificateRequest req = new RevokeCertificateRequest();

        // Set the certificate authority ARN.
        req.setCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

        // Set the certificate serial number.
        req.setCertificateSerial("79:3f:0d:5b:6a:04:12:5e:2c:9c:fb:52:37:35:98:fe");

        // Set the RevocationReason.
        req.withRevocationReason(RevocationReason.<<KEY_COMPROMISE>>);

        // Revoke the certificate.
        try {
            client.revokeCertificate(req);
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestAlreadyProcessedException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
    }
}
```

TagCertificateAuthorities

En el siguiente ejemplo de Java, se muestra cómo utilizar la TagCertificateAuthorityoperación.

Esta operación agrega una o varias etiquetas a la entidad de certificación privada. Las etiquetas son etiquetas que puede utilizar para identificar y organizar AWS los recursos. Cada etiqueta consta de una clave y un valor opcional. Cuando se llama a esta operación, se especifica la entidad de certificación privada por su nombre de recurso de Amazon (ARN). Especifique la etiqueta utilizando un par clave-valor. Para identificar una característica específica de esa entidad de certificación (CA), puede aplicar una etiqueta a una sola CA privada. O bien, para filtrar una relación común entre ellos CAs, puedes aplicar la misma etiqueta a varios privados CAs. Para eliminar una o más etiquetas, utilice la UntagCertificateAuthorityoperación. Llame a la ListTagsoperación para ver qué etiquetas están asociadas a su CA.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.TagCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.Tag;

import java.util.ArrayList;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidTagException;
import com.amazonaws.services.acmpca.model.TooManyTagsException;

public class TagCertificateAuthorities {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
```

```
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
          .withEndpointConfiguration(endpoint)
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
          .build();

      // Create a tag - method 1
      Tag tag1 = new Tag();
      tag1.withKey("Administrator");
      tag1.withValue("Bob");

      // Create a tag - method 2
      Tag tag2 = new Tag()
          .withKey("Purpose")
          .withValue("WebServices");

      // Add the tags to a collection.
      ArrayList<Tag> tags = new ArrayList<Tag>();
      tags.add(tag1);
      tags.add(tag2);

      // Create a request object and specify the certificate authority ARN.
      TagCertificateAuthorityRequest req = new TagCertificateAuthorityRequest();
      req.setCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");
      req.setTags(tags);

      // Add a tag
      try {
```

```
            client.tagCertificateAuthority(req);
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidTagException ex) {
            throw ex;
        } catch (TooManyTagsException ex) {
            throw ex;
        }
    }
}
```

UntagCertificateAuthority

El siguiente ejemplo de Java muestra cómo utilizar la UntagCertificateAuthorityoperación.

Esta operación elimina una o varias etiquetas de la entidad de certificación privada. Las etiquetas se componen de un par clave-valor. Si no se especifica la parte del valor de la etiqueta al llamar a esta operación, la etiqueta se eliminará independientemente del valor que tenga. Si se especifica un valor, la etiqueta solamente se eliminará si está asociada con el valor especificado. Para añadir etiquetas a una CA privada, utilice la TagCertificateAuthorityoperación. Llame a la ListTagsoperación para ver qué etiquetas están asociadas a su CA.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.util.ArrayList;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.UntagCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.Tag;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
```

```
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidTagException;

public class UntagCertificateAuthority {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
          .withEndpointConfiguration(endpoint)
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
          .build();

      // Create a Tag object with the tag to delete.
      Tag tag = new Tag();
      tag.withKey("Administrator");
      tag.withValue("Bob");

      // Add the tags to a collection.
      ArrayList<Tag> tags = new ArrayList<Tag>();
      tags.add(tag);

      // Create a request object and specify the certificate authority ARN.
      UntagCertificateAuthorityRequest req = new UntagCertificateAuthorityRequest();
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");
      req.withTags(tags);
```

```
        // Delete the tag
        try {
            client.untagCertificateAuthority(req);
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidTagException ex) {
            throw ex;
        }
    }
}
```

UpdateCertificateAuthority

En el siguiente ejemplo de Java, se muestra cómo utilizar la UpdateCertificateAuthorityoperación.

La operación actualiza el estado o la configuración de una entidad de certificación (CA) privada. Para poder actualizar una CA, esta debe tener el estado `ACTIVE` o `DISABLED`. Puede deshabilitar una CA privada que tenga el estado `ACTIVE` o activar una CA que tenga el estado `DISABLED`.

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.UpdateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CertificateAuthorityStatus;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
```

```
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;

public class UpdateCertificateAuthority {

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from file.",
 e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
          .withEndpointConfiguration(endpoint)
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
          .build();

      // Create the request object.
      UpdateCertificateAuthorityRequest req = new UpdateCertificateAuthorityRequest();

      // Set the ARN of the private CA that you want to update.
      req.setCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Define the certificate revocation list configuration. If you do not want to
      // update the CRL configuration, leave the CrlConfiguration structure alone and
      // do not set it on your UpdateCertificateAuthorityRequest object.
      CrlConfiguration crlConfigure = new CrlConfiguration();
      crlConfigure.withEnabled(true);
      crlConfigure.withExpirationInDays(365);
```

```
        crlConfigure.withCustomCname("your-custom-name");
        crlConfigure.withS3BucketName("your-bucket-name");

        // Set the CRL configuration onto your UpdateCertificateAuthorityRequest object.
        // If you do not want to change your CRL configuration, do not use the
        // setCrlConfiguration method.
        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);
        req.setRevocationConfiguration(revokeConfig);

        // Set the status.
        req.withStatus(CertificateAuthorityStatus.<<ACTIVE>>);

        // Create the result object.
        try {
            client.updateCertificateAuthority(req);
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        }
    }
}
```

Cree CAs y certifique con nombres de asunto personalizados

El CustomAttributeobjeto permite a los administradores pasar identificadores de objetos personalizados (OIDs) a certificados privados CAs y certificados. OIDs Los personalizados se pueden utilizar para crear jerarquías especializadas de nombres de sujetos que reflejen la estructura y las necesidades de su organización. Los certificados personalizados se deben crear con una de las plantillas `ApiPassthrough`. Para obtener más información acerca de las plantillas, consulte AWS Private CA variedades de plantillas. Para obtener más información sobre el uso de atributos personalizados, consulte Emita certificados privados de entidad final y Cree una CA privada en AWS Private CA.

No se puede utilizar `StandardAttributes` con `CustomAttributes`. Sin embargo, puede aprobar el estándar OIDs como parte de un. `CustomAttributes` El nombre de asunto predeterminado OIDs se muestra en la siguiente tabla:

Nombre del asunto	ID de objeto
País	2.5.4.6
CommonName	2.5.4.3
DistinguishedNameQualifier	2.5.4,46
GenerationQualifier	2.5.4,44
GivenName	2.5.4,42
Iniciales	2.5.4,43
Localidad	2.5.4.7
Organización	2.5.4,10
OrganizationalUnit	2.5.4,11
Seudónimo	2.5.4,65
SerialNumber	2.5.4.5
Estado	2.5.4.8
Apellido	2.5.4.4
Título	2.5.4.12

Temas

- Cree CA con CustomAttribute
- Emita un certificado con CustomAttribute

Cree CA con CustomAttribute

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;


import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;


public class CreateCertificateAuthorityWithCustomAttributes {

    public static void main(String[] args) throws Exception {

        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
```

```
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                   "Cannot load the credentials from the credential profiles file. " +
                   "Please make sure that your credentials file is at the correct " +
                   "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
 format.",
                   e);
        }

        // Define the endpoint for your sample.
        String endpointRegion = "us-west-2";  // Substitute your region here, e.g. "us-
west-2"
        String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("2.5.4.6") // Country
                .withValue("US"),
            new CustomAttribute()
                .withObjectIdentifier("2.5.4.3") // CommonName
                .withValue("CommonName"),
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1") // CustomOID
                .withValue("ABCDEFG"),
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1") // CustomOID
                .withValue("BCDEFGH")
        );

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
```

```
        subject.setCustomAttributes(customAttributes);

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.RSA_2048);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);
        configCA.withSubject(subject);

        // Define a certificate revocation list configuration.
        CrlConfiguration crlConfigure = new CrlConfiguration();
        crlConfigure.withEnabled(true);
        crlConfigure.withExpirationInDays(365);
        crlConfigure.withCustomCname(null);
        crlConfigure.withS3BucketName("your-bucket-name");

        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);

        // Define a certificate authority type: ROOT or SUBORDINATE
        CertificateAuthorityType caType = CertificateAuthorityType.SUBORDINATE;

        // Create a tag - method 1
        Tag tag1 = new Tag();
        tag1.withKey("PrivateCA");
        tag1.withValue("Sample");

        // Create a tag - method 2
        Tag tag2 = new Tag()
            .withKey("Purpose")
            .withValue("WebServices");

        // Add the tags to a collection.
        ArrayList<Tag> tags = new ArrayList<Tag>();
        tags.add(tag1);
        tags.add(tag2);

        // Create the request object.
        CreateCertificateAuthorityRequest req = new
CreateCertificateAuthorityRequest();
        req.withCertificateAuthorityConfiguration(configCA);
        req.withRevocationConfiguration(revokeConfig);
        req.withIdempotencyToken("1234");
        req.withCertificateAuthorityType(caType);
```

```
        req.withTags(tags);


        // Create the private CA.
        CreateCertificateAuthorityResult result = null;
        try {
            result = client.createCertificateAuthority(req);
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

        // Retrieve the ARN of the private CA.
        String arn = result.getCertificateAuthorityArn();
        System.out.println(arn);
    }
}
```

Emita un certificado con CustomAttribute

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;
import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CustomAttribute;
```

```
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;

public class IssueCertificateWithCustomAttributes {
   private static ByteBuffer stringToByteBuffer(final String string) {
      if (Objects.isNull(string)) {
          return null;
      }
      byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
      return ByteBuffer.wrap(bytes);
   }

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
      String endpointRegion = "us-west-2";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
```

```
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

      // Create a certificate request:
      IssueCertificateRequest req = new IssueCertificateRequest();

      // Set the CA ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
         "certificate-authority/12345678-1234-1234-1234-123456789012");

      // Specify the certificate signing request (CSR) for the certificate to be signed
 and issued.
      String strCSR =
      "-----BEGIN CERTIFICATE REQUEST-----\n" +
      "base64-encoded CSR\n" +
      "-----END CERTIFICATE REQUEST-----\n";
      ByteBuffer csrByteBuffer = stringToByteBuffer(strCSR);
      req.setCsr(csrByteBuffer);

      // Specify the template for the issued certificate.
      req.withTemplateArn("arn:aws:acm-pca:::template/
EndEntityCertificate_APIPassthrough/V1");

      // Set the signing algorithm.
      req.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);

      // Set the validity period for the certificate to be issued.
      Validity validity = new Validity();
      validity.withValue(100L);
      validity.withType("DAYS");
      req.withValidity(validity);

      // Set the idempotency token.
      req.setIdempotencyToken("1234");

      // Define custom attributes
      List<CustomAttribute> customAttributes = Arrays.asList(
         new CustomAttribute()
               .withObjectIdentifier("2.5.4.6") // Country
               .withValue("US"),
         new CustomAttribute()
               .withObjectIdentifier("2.5.4.3") // CommonName
               .withValue("CommonName"),
```

```
            new CustomAttribute()
                  .withObjectIdentifier("1.3.6.1.4.1") // CustomOID
                  .withValue("ABCDEFG"),
            new CustomAttribute()
                  .withObjectIdentifier("1.3.6.1.4.1") // CustomOID
                  .withValue("BCDEFGH")
      );

      // Define certificate subject.
      ASN1Subject subject = new ASN1Subject();
      subject.setCustomAttributes(customAttributes);

      // Add subject to api-passthrough
      ApiPassthrough apiPassthrough = new ApiPassthrough();
      apiPassthrough.setSubject(subject);
      req.setApiPassthrough(apiPassthrough);

      // Issue the certificate.
      IssueCertificateResult result = null;
      try {
         result = client.issueCertificate(req);
      } catch (LimitExceededException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidArgsException ex) {
         throw ex;
      } catch (MalformedCSRException ex) {
         throw ex;
      }

      // Retrieve and display the certificate ARN.
      String arn = result.getCertificateArn();
      System.out.println(arn);
   }
}
```

Cree certificados con extensiones personalizadas

El CustomExtensionobjeto permite a los administradores establecer extensiones X.509 personalizadas en los certificados privados. Los certificados personalizados se deben crear con una de las ApiPassthrough plantillas. Para obtener más información acerca de las plantillas, consulte AWS Private CA variedades de plantillas. Para obtener más información acerca de las extensiones personalizadas, consulte Emita certificados privados de entidad final.

Temas

- Active una CA subordinada con la extensión NameConstraints
- Emita un certificado con la extensión de la instrucción QC

Active una CA subordinada con la extensión NameConstraints

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.Objects;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;
import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;
```

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateResult;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;

import org.bouncycastle.asn1.x509.GeneralName;
import org.bouncycastle.asn1.x509.GeneralSubtree;
import org.bouncycastle.asn1.x509.NameConstraints;

import lombok.SneakyThrows;

public class SubordinateCAActivationWithNameConstraints {
   public static void main(String[] args) throws Exception {
      // Place your own Root CA ARN here.
      String rootCAArn = "arn:aws:acm-pca:region:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012";
```

```
      // Define the endpoint region for your sample.
      String endpointRegion = "us-west-2";  // Substitute your region here, e.g. "us-
west-2"

      // Define a CA subject.
      ASN1Subject subject = new ASN1Subject();
      subject.setOrganization("Example Organization");
      subject.setOrganizationalUnit("Example");
      subject.setCountry("US");
      subject.setState("Virginia");
      subject.setLocality("Arlington");
      subject.setCommonName("SubordinateCA");

      // Define the CA configuration.
      CertificateAuthorityConfiguration configCA = new
 CertificateAuthorityConfiguration();
      configCA.withKeyAlgorithm(KeyAlgorithm.RSA_2048);
      configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);
      configCA.withSubject(subject);

      // Define a certificate revocation list configuration.
      CrlConfiguration crlConfigure = new CrlConfiguration();
      crlConfigure.withEnabled(true);
      crlConfigure.withExpirationInDays(365);
      crlConfigure.withCustomCname(null);
      crlConfigure.withS3BucketName("your-bucket-name");

      // Define a certificate authority type
      CertificateAuthorityType caType = CertificateAuthorityType.SUBORDINATE;

      // ** Execute core code samples for Subordinate CA activation in sequence **
      AWSACMPCA client = ClientBuilder(endpointRegion);
      String rootCertificate = GetCertificateAuthorityCertificate(rootCAArn, client);
      String subordinateCAArn = CreateCertificateAuthority(configCA, crlConfigure,
 caType, client);
      String csr = GetCertificateAuthorityCsr(subordinateCAArn, client);
      String subordinateCertificateArn = IssueCertificate(rootCAArn, csr, client);
      String subordinateCertificate = GetCertificate(subordinateCertificateArn,
 rootCAArn, client);
      ImportCertificateAuthorityCertificate(subordinateCertificate, rootCertificate,
 subordinateCAArn, client);
   }
```

```
  private static AWSACMPCA ClientBuilder(String endpointRegion) {
      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException(
                  "Cannot load the credentials from the credential profiles file. " +
                  "Please make sure that your credentials file is at the correct " +
                  "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                  e);
      }

      String endpointProtocol = "https://acm-pca." + endpointRegion +
".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
          .withEndpointConfiguration(endpoint)
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
          .build();

      return client;
  }

  private static String GetCertificateAuthorityCertificate(String rootCAArn, AWSACMPCA
client) {
      // ** GetCertificateAuthorityCertificate **

      // Create a request object and set the certificate authority ARN,
      GetCertificateAuthorityCertificateRequest getCACertificateRequest =
           new GetCertificateAuthorityCertificateRequest();
      getCACertificateRequest.withCertificateAuthorityArn(rootCAArn);

      // Create a result object.
      GetCertificateAuthorityCertificateResult getCACertificateResult = null;
      try {
          getCACertificateResult =
client.getCertificateAuthorityCertificate(getCACertificateRequest);
```

```
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        }

        // Retrieve and display the certificate information.
        String rootCertificate = getCACertificateResult.getCertificate();
        System.out.println("Root CA Certificate / Certificate Chain:");
        System.out.println(rootCertificate);

        return rootCertificate;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CrlConfiguration crlConfigure, CertificateAuthorityType caType, AWSACMPCA
client) {
        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);

        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withRevocationConfiguration(revokeConfig);
        createCARequest.withIdempotencyToken("1234");
        createCARequest.withCertificateAuthorityType(caType);

        // Create the private CA.
        CreateCertificateAuthorityResult createCAResult = null;
        try {
            createCAResult = client.createCertificateAuthority(createCARequest);
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

        // Retrieve the ARN of the private CA.
        String subordinateCAArn = createCAResult.getCertificateAuthorityArn();
```

```
        System.out.println("Subordinate CA Arn: " + subordinateCAArn);

        return subordinateCAArn;
    }

    private static String GetCertificateAuthorityCsr(String subordinateCAArn, AWSACMPCA
client) {

        // Create the CSR request object and set the CA ARN.
        GetCertificateAuthorityCsrRequest csrRequest = new
GetCertificateAuthorityCsrRequest();
        csrRequest.withCertificateAuthorityArn(subordinateCAArn);

        // Create waiter to wait on successful creation of the CSR file.
        Waiter<GetCertificateAuthorityCsrRequest> getCSRWaiter =
client.waiters().certificateAuthorityCSRCreated();
        try {
            getCSRWaiter.run(new WaiterParameters<>(csrRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch(AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        // Retrieve and display the CSR;
        String csr = csrResult.getCsr();
        System.out.println("Subordinate CSR:");
```

```
        System.out.println(csr);

        return csr;
    }

    private static String IssueCertificate(String rootCAArn, String csr, AWSACMPCA
 client) {

        // Create a certificate request:
        IssueCertificateRequest issueRequest = new IssueCertificateRequest();

        // Set the issuing CA ARN.
        issueRequest.withCertificateAuthorityArn(rootCAArn);

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/
SubordinateCACertificate_PathLen0_APIPassthrough/V1");

        ByteBuffer csrByteBuffer = stringToByteBuffer(csr);
        issueRequest.setCsr(csrByteBuffer);

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(100L);
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

        // Set the idempotency token.
        issueRequest.setIdempotencyToken("1234");

        // Generate Base64 encoded Nameconstraints extension value
        String base64EncodedExtValue = getNameConstraintExtensionValue();

        // Generate custom extension
        CustomExtension customExtension = new CustomExtension();
        customExtension.setCritical(true);
        customExtension.setObjectIdentifier("2.5.29.30"); // NameConstraints Extension
 OID
        customExtension.setValue(base64EncodedExtValue);

        // Add custom extension to api-passthrough
```

```
        ApiPassthrough apiPassthrough = new ApiPassthrough();
        Extensions extensions = new Extensions();
        extensions.setCustomExtensions(Arrays.asList(customExtension));
        apiPassthrough.setExtensions(extensions);
        issueRequest.setApiPassthrough(apiPassthrough);

        // Issue the certificate.
        IssueCertificateResult issueResult = null;
        try {
            issueResult = client.issueCertificate(issueRequest);
        } catch (LimitExceededException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Retrieve and display the certificate ARN.
        String subordinateCertificateArn = issueResult.getCertificateArn();
        System.out.println("Subordinate Certificate Arn: " + subordinateCertificateArn);

        return subordinateCertificateArn;
    }

    @SneakyThrows
    private static String getNameConstraintExtensionValue() {
       // Generate Base64 encoded Nameconstraints extension value
       GeneralSubtree dnsPrivate = new GeneralSubtree(new
GeneralName(GeneralName.dNSName, ".private"));
       GeneralSubtree dnsLocal = new GeneralSubtree(new GeneralName(GeneralName.dNSName,
".local"));
       GeneralSubtree dnsCorp = new GeneralSubtree(new GeneralName(GeneralName.dNSName,
".corp"));
       GeneralSubtree dnsSecretCorp = new GeneralSubtree(new
GeneralName(GeneralName.dNSName, ".secret.corp"));
       GeneralSubtree dnsExample = new GeneralSubtree(new
GeneralName(GeneralName.dNSName, ".example.com"));
```

```
      GeneralSubtree[] permittedSubTree = new GeneralSubtree[] { dnsPrivate, dnsLocal,
dnsCorp };
      GeneralSubtree[] excludedSubTree = new GeneralSubtree[] { dnsSecretCorp,
dnsExample };
      NameConstraints nameConstraints = new NameConstraints(permittedSubTree,
excludedSubTree);

      return new String(Base64.getEncoder().encode(nameConstraints.getEncoded()));
  }

  private static String GetCertificate(String subordinateCertificateArn, String
rootCAArn, AWSACMPCA client) {

       // Create a request object.
       GetCertificateRequest certificateRequest = new GetCertificateRequest();

       // Set the certificate ARN.
       certificateRequest.withCertificateArn(subordinateCertificateArn);

       // Set the certificate authority ARN.
       certificateRequest.withCertificateAuthorityArn(rootCAArn);

       // Create waiter to wait on successful creation of the certificate file.
       Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
       try {
           getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
       } catch (WaiterUnrecoverableException e) {
           //Explicit short circuit when the recourse transitions into
           //an undesired state.
       } catch (WaiterTimedOutException e) {
           //Failed to transition into desired state even after polling.
       } catch (AWSACMPCAException e) {
           //Unexpected service exception.
       }

       // Retrieve the certificate and certificate chain.
       GetCertificateResult certificateResult = null;
       try {
           certificateResult = client.getCertificate(certificateRequest);
       } catch (RequestInProgressException ex) {
           throw ex;
       } catch (RequestFailedException ex) {
           throw ex;
```

```
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

        // Get the certificate and certificate chain and display the result.
        String subordinateCertificate = certificateResult.getCertificate();
        System.out.println("Subordinate CA Certificate:");
        System.out.println(subordinateCertificate);

        return subordinateCertificate;
    }

    private static void ImportCertificateAuthorityCertificate(String
subordinateCertificate, String rootCertificate, String subordinateCAArn, AWSACMPCA
client) {

        // Create the request object and set the signed certificate, chain and CA ARN.
        ImportCertificateAuthorityCertificateRequest importRequest =
            new ImportCertificateAuthorityCertificateRequest();

        ByteBuffer certByteBuffer = stringToByteBuffer(subordinateCertificate);
        importRequest.setCertificate(certByteBuffer);

        ByteBuffer rootCACertByteBuffer = stringToByteBuffer(rootCertificate);
        importRequest.setCertificateChain(rootCACertByteBuffer);

        // Set the certificate authority ARN.
        importRequest.withCertificateAuthorityArn(subordinateCAArn);

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
```

```
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
        System.out.println("Subordinate CA certificate successfully imported.");
        System.out.println("Subordinate CA activated successfully.");
    }

    private static ByteBuffer stringToByteBuffer(final String string) {
        if (Objects.isNull(string)) {
            return null;
        }
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
        return ByteBuffer.wrap(bytes);
    }
}
```

Emita un certificado con la extensión de la instrucción QC

```
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.Objects;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
```

```
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;

import org.bouncycastle.asn1.ASN1EncodableVector;
import org.bouncycastle.asn1.ASN1ObjectIdentifier;
import org.bouncycastle.asn1.DERSequence;
import org.bouncycastle.asn1.DERUTF8String;
import org.bouncycastle.asn1.x509.qualified.ETSIQCObjectIdentifiers;
import org.bouncycastle.asn1.x509.qualified.QCStatement;

import lombok.SneakyThrows;

public class IssueCertificateWithQCStatement {
   private static ByteBuffer stringToByteBuffer(final String string) {
      if (Objects.isNull(string)) {
          return null;
      }
      byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
      return ByteBuffer.wrap(bytes);
   }

   @SneakyThrows
   private static String generateQCStatementBase64ExtValue() {
      DERSequence qcTypeSeq = new DERSequence(ETSIQCObjectIdentifiers.id_etsi_qct_web);
      QCStatement qcType = new QCStatement(ETSIQCObjectIdentifiers.id_etsi_qcs_QcType,
 qcTypeSeq);

      ASN1EncodableVector pspAIVector = new ASN1EncodableVector(2);
      pspAIVector.add(new ASN1ObjectIdentifier("0.4.0.19495.1.3"));
      pspAIVector.add(new DERUTF8String("PSP_AI"));
      DERSequence pspAISeq = new DERSequence(pspAIVector);

      ASN1EncodableVector pspASVector = new ASN1EncodableVector(2);
      pspASVector.add(new ASN1ObjectIdentifier("0.4.0.19495.1.1"));
      pspASVector.add(new DERUTF8String("PSP_AS"));
      DERSequence pspASSeq = new DERSequence(pspASVector);
```

```
      ASN1EncodableVector pspPIVector = new ASN1EncodableVector(2);
      pspPIVector.add(new ASN1ObjectIdentifier("0.4.0.19495.1.2"));
      pspPIVector.add(new DERUTF8String("PSP_PI"));
      DERSequence pspPISeq = new DERSequence(pspPIVector);

      ASN1EncodableVector pspICVector = new ASN1EncodableVector(2);
      pspICVector.add(new ASN1ObjectIdentifier("0.4.0.19495.1.4"));
      pspICVector.add(new DERUTF8String("PSP_IC"));
      DERSequence pspICSeq = new DERSequence(pspICVector);

      ASN1EncodableVector pspSeqVector = new ASN1EncodableVector(4);
      pspSeqVector.add(pspPISeq);
      pspSeqVector.add(pspICSeq);
      pspSeqVector.add(pspASSeq);
      pspSeqVector.add(pspAISeq);
      DERSequence pspSeq = new DERSequence(pspSeqVector);

      ASN1EncodableVector pspVector = new ASN1EncodableVector(3);
      pspVector.add(pspSeq);
      pspVector.add(new DERUTF8String("Your Financial Authority"));
      pspVector.add(new DERUTF8String("AB-CD"));
      DERSequence psp = new DERSequence(pspVector);
      QCStatement qcPSP = new QCStatement(new ASN1ObjectIdentifier("0.4.0.19495.2"),
psp);

      DERSequence qcSeq = new DERSequence(new QCStatement[] { qcType, qcPSP });

      byte[] qcExtValueInBytes = qcSeq.getEncoded();
      return Base64.getEncoder().encodeToString(qcExtValueInBytes);
   }

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from disk", e);
      }

      // Define the endpoint for your sample.
```

```
      String endpointRegion = "us-west-2";  // Substitute your region here, e.g. "us-
west-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
          .withEndpointConfiguration(endpoint)
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
          .build();

      // Create a certificate request:
      IssueCertificateRequest req = new IssueCertificateRequest();

      // Set the CA ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
         "certificate-authority/12345678-1234-1234-1234-123456789012");

      // Specify the certificate signing request (CSR) for the certificate to be signed
 and issued.
      String strCSR =
      "-----BEGIN CERTIFICATE REQUEST-----\n" +
      "base64-encoded CSR\n" +
      "-----END CERTIFICATE REQUEST-----\n";
      ByteBuffer csrByteBuffer = stringToByteBuffer(strCSR);
      req.setCsr(csrByteBuffer);

      // Specify the template for the issued certificate.
      req.withTemplateArn("arn:aws:acm-pca:::template/
EndEntityCertificate_APIPassthrough/V1");

      // Set the signing algorithm.
      req.withSigningAlgorithm(SigningAlgorithm.SHA256WITHRSA);

      // Set the validity period for the certificate to be issued.
      Validity validity = new Validity();
      validity.withValue(30L);
      validity.withType("DAYS");
      req.withValidity(validity);

      // Set the idempotency token.
      req.setIdempotencyToken("1234");
```

```
      // Generate Base64 encoded extension value for QC Statement
      String base64EncodedExtValue = generateQCStatementBase64ExtValue();

      // Generate custom extension
      CustomExtension customExtension = new CustomExtension();
      customExtension.setObjectIdentifier("1.3.6.1.5.5.7.1.3"); // QC Statement
 Extension OID
      customExtension.setValue(base64EncodedExtValue);

      // Add custom extension to api-passthrough
      ApiPassthrough apiPassthrough = new ApiPassthrough();
      Extensions extensions = new Extensions();
      extensions.setCustomExtensions(Arrays.asList(customExtension));
      apiPassthrough.setExtensions(extensions);
      req.setApiPassthrough(apiPassthrough);

      // Issue the certificate.
      IssueCertificateResult result = null;
      try {
         result = client.issueCertificate(req);
      } catch (LimitExceededException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidArgsException ex) {
         throw ex;
      } catch (MalformedCSRException ex) {
         throw ex;
      }

      // Retrieve and display the certificate ARN.
      String arn = result.getCertificateArn();
      System.out.println(arn);
   }
}
```

Se utiliza Autoridad de certificación privada de AWS para implementar los certificados Matter

Puede usar la AWS Private Certificate Authority API para crear certificados que cumplan con el estándar de conectividad Matter. Matter especifica las configuraciones de certificados que mejoran la seguridad y la coherencia de los dispositivos de Internet de las cosas (IoT) en múltiples plataformas de ingeniería. Para obtener más información de Matter, consulte buildwithmatter.com.

Matter 1.2, publicado en octubre de 2023, admite la revocación del DAC mediante listas de revocación de certificados (). CRLs Para ayudarlo a cumplir con el estándar Matter actual, al habilitar la revocación de la CRL para CAs esa emisión de certificados Matter, en el `CrlConfiguration` objeto, en la estructura, establecido en. `CrlDistributionPointExtensionConfiguration` `OmitExtension` `true`

Por lo general, CAs incrusta el punto de distribución de la CRL (CDP) en los certificados que emiten para que las partes que confían en ellos y realizan la validación de la cadena de certificados puedan obtener la CRL y comprobar el estado del certificado. En Matter, el URI del CDP no se escribe en los certificados. En su lugar, los usuarios lo consultan CDPs en Matter Distributed Compliance Ledger (DCL), el almacén de datos de Matter de confianza. Debe cargar el URI del CDP en la DCL de Matter para poder detectarlo durante la validación. DACs Para obtener más información sobre cómo determinar el URI del CDP, consulte. Determinación del URI del punto de distribución (CDP) de la CRL Para obtener más información sobre Matter, consulte la página de inicio estándar de Matter.

Temas

- Activar una autoridad de certificación de productos (PAA)
- Activar un intermedio de certificación de producto (PAI)
- Crear un certificado de atestación de dispositivos (DAC)
- Active una CA raíz para los certificados operativos de nodo (NOC).
- Activar una CA subordinada para los certificados operativos de nodo (NOC)
- Crear un certificado operativo de nodo (NOC)

Activar una autoridad de certificación de productos (PAA)

En este ejemplo de Java se muestra cómo utilizar la Definición de Root CACertificate APIPassthrough _/V1 plantilla para crear e instalar un certificado Matter Root CA (PAA) para la certificación de productos. La extensión AuthorityKeyIdentifier (AKI) es opcional para. PAAs

Para configurar una AKI, debe generar un valor de AKI codificado en Base64 y pasarlo por un. CustomExtension

El ejemplo llama a las siguientes acciones de API: Autoridad de certificación privada de AWS

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Si tiene problemas, consulte Solucione los errores de los AWS Private CA certificados compatibles con Matter en la sección Resolución de problemas.

```
package com.amazonaws.samples.matter;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.samples.GetCertificateAuthorityCertificate;
import com.amazonaws.auth.AWSStaticCredentialsProvider;


import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;
```

```
import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CrlDistributionPointExtensionConfiguration;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;
```

```
import org.bouncycastle.asn1.x509.SubjectPublicKeyInfo;
import org.bouncycastle.cert.jcajce.JcaX509ExtensionUtils;
import org.bouncycastle.openssl.PEMParser;
import org.bouncycastle.pkcs.PKCS10CertificationRequest;
import org.bouncycastle.util.io.pem.PemReader;

import lombok.SneakyThrows;

public class ProductAttestationAuthorityActivation {

    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "ap-
southeast-2"

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("2.5.4.3") // CommonName
                .withValue("Matter Test PAA"),
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1.37244.2.1") // Vendor ID
                .withValue("FFF1")
        );

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
 CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.EC_prime256v1);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
        configCA.withSubject(subject);

        // Define a CRL distribution point extension configuration
        CrlDistributionPointExtensionConfiguration CDPConfigure = new
 CrlDistributionPointExtensionConfiguration();
        CDPConfigure.withOmitExtension(true);

        // Define a certificate revocation list configuration.
        CrlConfiguration crlConfigure = new CrlConfiguration();
        crlConfigure.withEnabled(true);
```

```
        crlConfigure.withExpirationInDays(365);
        crlConfigure.withCustomCname(null);
        crlConfigure.withS3BucketName("your-bucket-name");
        crlConfigure.withS3ObjectAcl("BUCKET_OWNER_FULL_CONTROL");
        crlConfigure.withCrlDistributionPointExtensionConfiguration(CDPConfigure);

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.ROOT;

        // ** Execute core code samples for Root CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCAArn = CreateCertificateAuthority(configCA, crlConfigure, CAtype,
client);
        String csr = GetCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = IssueCertificate(rootCAArn, csr, client);
        String rootCertificate = GetCertificate(rootCertificateArn, rootCAArn, client);
        ImportCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

    private static AWSACMPCA ClientBuilder(String endpointRegion) {
        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load the credentials from the credential profiles file. " +
                    "Please make sure that your credentials file is at the correct " +
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                    e);
        }

        String endpointProtocol = "https://acm-pca." + endpointRegion +
".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
```

```
            .build();

        return client;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CrlConfiguration crlConfigure, CertificateAuthorityType CAtype, AWSACMPCA
client) {
        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);

        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);
        createCARequest.withRevocationConfiguration(revokeConfig);

        // Create the private CA.
        CreateCertificateAuthorityResult createCAResult = null;
        try {
            createCAResult = client.createCertificateAuthority(createCARequest);
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

        // Retrieve the ARN of the private CA.
        String rootCAArn = createCAResult.getCertificateAuthorityArn();
        System.out.println("Product Attestation Authority (PAA) Arn: " + rootCAArn);

        return rootCAArn;
    }

    private static String GetCertificateAuthorityCsr(String rootCAArn, AWSACMPCA
client) {

        // Create the CSR request object and set the CA ARN.
        GetCertificateAuthorityCsrRequest csrRequest = new
GetCertificateAuthorityCsrRequest();
```

```
        csrRequest.withCertificateAuthorityArn(rootCAArn);

        // Create waiter to wait on successful creation of the CSR file.
        Waiter<GetCertificateAuthorityCsrRequest> getCSRWaiter =
client.waiters().certificateAuthorityCSRCreated();
        try {
            getCSRWaiter.run(new WaiterParameters<>(csrRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        // Retrieve and display the CSR;
        String csr = csrResult.getCsr();
        System.out.println(csr);

        return csr;
    }

    @SneakyThrows
    private static String generateAuthorityKeyIdentifier(final String csrPEM) {
        PKCS10CertificationRequest csr = getPKCS10CertificationRequest(csrPEM);
        SubjectPublicKeyInfo spki = csr.getSubjectPublicKeyInfo();

        JcaX509ExtensionUtils extensionUtils = new JcaX509ExtensionUtils();
```

```
        byte[] akiBytes =
 extensionUtils.createAuthorityKeyIdentifier(spki).getEncoded();

        return Base64.getEncoder().encodeToString(akiBytes);
    }

    @SneakyThrows
    private static PKCS10CertificationRequest getPKCS10CertificationRequest(final
 String csrPEM) {
        ByteArrayInputStream bais = new ByteArrayInputStream(csrPEM.getBytes());
        PemReader pemReader = new PemReader(new InputStreamReader(bais));
        PEMParser parser = new PEMParser(pemReader);
        Object o = parser.readObject();
        if (o instanceof PKCS10CertificationRequest) {
            return (PKCS10CertificationRequest) o;
        }
        return null;
    }

    private static String IssueCertificate(String rootCAArn, String csr, AWSACMPCA
 client) {

        // Create a certificate request:
        IssueCertificateRequest issueRequest = new IssueCertificateRequest();

        // Set the CA ARN.
        issueRequest.withCertificateAuthorityArn(rootCAArn);

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/
RootCACertificate_APIPassthrough/V1");

        ByteBuffer csrByteBuffer = stringToByteBuffer(csr);
        issueRequest.setCsr(csrByteBuffer);

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(3650L);
        validity.withType("DAYS");
        issueRequest.withValidity(validity);
```

```
        // Set the idempotency token.
        issueRequest.setIdempotencyToken("1234");

        // Generate Base64 encoded extension value for AuthorityKeyIdentifier
        String base64EncodedExtValue = generateAuthorityKeyIdentifier(csr);

        // Generate custom extension
        CustomExtension customExtension = new CustomExtension();
        customExtension.setObjectIdentifier("2.5.29.35"); // AuthorityKeyIdentifier
Extension OID
        customExtension.setValue(base64EncodedExtValue);

        // Add custom extension to api-passthrough
        ApiPassthrough apiPassthrough = new ApiPassthrough();
        Extensions extensions = new Extensions();
        extensions.setCustomExtensions(Arrays.asList(customExtension));
        apiPassthrough.setExtensions(extensions);
        issueRequest.setApiPassthrough(apiPassthrough);

        // Issue the certificate.
        IssueCertificateResult issueResult = null;
        try {
            issueResult = client.issueCertificate(issueRequest);
        } catch (LimitExceededException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Retrieve and display the certificate ARN.
        String rootCertificateArn = issueResult.getCertificateArn();
        System.out.println("Product Attestation Authority (PAA) Certificate Arn: " +
rootCertificateArn);

        return rootCertificateArn;
    }
```

```
   private static String GetCertificate(String rootCertificateArn, String rootCAArn,
AWSACMPCA client) {

       // Create a request object.
       GetCertificateRequest certificateRequest = new GetCertificateRequest();

       // Set the certificate ARN.
       certificateRequest.withCertificateArn(rootCertificateArn);

       // Set the certificate authority ARN.
       certificateRequest.withCertificateAuthorityArn(rootCAArn);

       // Create waiter to wait on successful creation of the certificate file.
       Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
       try {
           getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
       } catch (WaiterUnrecoverableException e) {
           //Explicit short circuit when the recourse transitions into
           //an undesired state.
       } catch (WaiterTimedOutException e) {
           //Failed to transition into desired state even after polling.
       } catch (AWSACMPCAException e) {
           //Unexpected service exception.
       }

       // Retrieve the certificate and certificate chain.
       GetCertificateResult certificateResult = null;
       try {
           certificateResult = client.getCertificate(certificateRequest);
       } catch (RequestInProgressException ex) {
           throw ex;
       } catch (RequestFailedException ex) {
           throw ex;
       } catch (ResourceNotFoundException ex) {
           throw ex;
       } catch (InvalidArnException ex) {
           throw ex;
       } catch (InvalidStateException ex) {
           throw ex;
       }

       // Get the certificate and certificate chain and display the result.
```

```
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
    }

    private static void ImportCertificateAuthorityCertificate(String rootCertificate,
String rootCAArn, AWSACMPCA client) {

        // Create the request object and set the signed certificate, chain and CA ARN.
        ImportCertificateAuthorityCertificateRequest importRequest =
            new ImportCertificateAuthorityCertificateRequest();

        ByteBuffer certByteBuffer = stringToByteBuffer(rootCertificate);
        importRequest.setCertificate(certByteBuffer);

        importRequest.setCertificateChain(null);

        // Set the certificate authority ARN.
        importRequest.withCertificateAuthorityArn(rootCAArn);

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Product Attestation Authority (PAA) certificate
successfully imported.");
        System.out.println("Product Attestation Authority (PAA) activated
successfully.");
```

```
    }

    private static ByteBuffer stringToByteBuffer(final String string) {
        if (Objects.isNull(string)) {
            return null;
        }
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
        return ByteBuffer.wrap(bytes);
    }
}
```

Activar un intermedio de certificación de producto (PAI)

En este ejemplo de Java se muestra cómo utilizar la BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1definición plantilla para crear e instalar un certificado de CA (PAI) de Matter Subordinate para la certificación de productos. Debe generar un KeyUsage valor codificado en Base64 y pasarlo por un. CustomExtension

El ejemplo llama a las siguientes acciones de API: Autoridad de certificación privada de AWS

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate
- GetCertificateAuthorityCertificate

Si tiene problemas, consulte Solucione los errores de los AWS Private CA certificados compatibles con Matter en la sección Resolución de problemas.

```
package com.amazonaws.samples.matter;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;


import com.amazonaws.services.acmpca.AWSACMPCA;
```

```
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CrlDistributionPointExtensionConfiguration;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateResult;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
```

```
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import lombok.SneakyThrows;

public class ProductAttestationIntermediateActivation {

    public static void main(String[] args) throws Exception {
        // Place your own Root CA ARN here.
        String paaArn = "arn:aws:acm-pca:region:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012";

        // Define the endpoint region for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "ap-
southeast-2"

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("2.5.4.3") // CommonName
                .withValue("Matter Test PAI"),
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1.37244.2.1") // Vendor ID
                .withValue("FFF1"),
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1.37244.2.2") // Product ID
                .withValue("8000")
        );

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);
```

```
        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.EC_prime256v1);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
        configCA.withSubject(subject);

        // Define a CRL distribution point extension configuration
        CrlDistributionPointExtensionConfiguration CDPConfigure = new
CrlDistributionPointExtensionConfiguration();
        CDPConfigure.withOmitExtension(true);

        // Define a certificate revocation list configuration.
        CrlConfiguration crlConfigure = new CrlConfiguration();
        crlConfigure.withEnabled(true);
        crlConfigure.withExpirationInDays(365);
        crlConfigure.withCustomCname(null);
        crlConfigure.withS3BucketName("your-bucket-name");
        crlConfigure.withS3ObjectAcl("BUCKET_OWNER_FULL_CONTROL");
        crlConfigure.withCrlDistributionPointConfiguration(CDPConfigure);

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.SUBORDINATE;

        // ** Execute core code samples for Subordinate CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCertificate = GetCertificateAuthorityCertificate(paaArn, client);
        String subordinateCAArn = CreateCertificateAuthority(configCA, crlConfigure,
CAtype, client);
        String csr = GetCertificateAuthorityCsr(subordinateCAArn, client);
        String subordinateCertificateArn = IssueCertificate(paaArn, csr, client);
        String subordinateCertificate = GetCertificate(subordinateCertificateArn,
paaArn, client);
        ImportCertificateAuthorityCertificate(subordinateCertificate, rootCertificate,
subordinateCAArn, client);

    }

    private static AWSACMPCA ClientBuilder(String endpointRegion) {
        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
```

```
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load the credentials from the credential profiles file. " +
                    "Please make sure that your credentials file is at the correct " +
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                    e);
        }

        String endpointProtocol = "https://acm-pca." + endpointRegion +
".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        return client;
    }

    private static String GetCertificateAuthorityCertificate(String rootCAArn,
AWSACMPCA client) {
        // ** GetCertificateAuthorityCertificate **

        // Create a request object and set the certificate authority ARN,
        GetCertificateAuthorityCertificateRequest getCACertificateRequest =
        new GetCertificateAuthorityCertificateRequest();
        getCACertificateRequest.withCertificateAuthorityArn(rootCAArn);

        // Create a result object.
        GetCertificateAuthorityCertificateResult getCACertificateResult = null;
        try {
            getCACertificateResult =
client.getCertificateAuthorityCertificate(getCACertificateRequest);
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
```

```
            throw ex;
        }

        // Retrieve and display the certificate information.
        String rootCertificate = getCACertificateResult.getCertificate();
        System.out.println("Product Attestation Authority (PAA) Certificate /
Certificate Chain:");
        System.out.println(rootCertificate);

        return rootCertificate;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CrlConfiguration crlConfigure, CertificateAuthorityType CAtype, AWSACMPCA
client) {
        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);

        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);
        createCARequest.withRevocationConfiguration(revokeConfig);

        // Create the private CA.
        CreateCertificateAuthorityResult createCAResult = null;
        try {
            createCAResult = client.createCertificateAuthority(createCARequest);
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

        // Retrieve the ARN of the private CA.
        String subordinateCAArn = createCAResult.getCertificateAuthorityArn();
        System.out.println("Product Attestation Intermediate (PAI) Arn: " +
subordinateCAArn);

        return subordinateCAArn;
```

```
    }

    private static String GetCertificateAuthorityCsr(String subordinateCAArn, AWSACMPCA
client) {

        // Create the CSR request object and set the CA ARN.
        GetCertificateAuthorityCsrRequest csrRequest = new
GetCertificateAuthorityCsrRequest();
        csrRequest.withCertificateAuthorityArn(subordinateCAArn);

        // Create waiter to wait on successful creation of the CSR file.
        Waiter<GetCertificateAuthorityCsrRequest> getCSRWaiter =
client.waiters().certificateAuthorityCSRCreated();
        try {
            getCSRWaiter.run(new WaiterParameters<>(csrRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch(AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        // Retrieve and display the CSR;
        String csr = csrResult.getCsr();
        System.out.println("Subordinate CSR:");
        System.out.println(csr);

        return csr;
```

```
    }

    private static String IssueCertificate(String rootCAArn, String csr, AWSACMPCA
 client) {

        // Create a certificate request:
        IssueCertificateRequest issueRequest = new IssueCertificateRequest();

        // Set the issuing CA ARN.
        issueRequest.withCertificateAuthorityArn(rootCAArn);

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/
BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1");

        ByteBuffer csrByteBuffer = stringToByteBuffer(csr);
        issueRequest.setCsr(csrByteBuffer);

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(730L); // Approximately two years
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

        // Set the idempotency token.
        issueRequest.setIdempotencyToken("1234");

        ApiPassthrough apiPassthrough = new ApiPassthrough();

        // Generate Base64 encoded extension value for ExtendedKeyUsage
        String base64EncodedKUValue = generateKeyUsageValue();

        // Generate custom extension
        CustomExtension customKeyUsageExtension = new CustomExtension();
        customKeyUsageExtension.setObjectIdentifier("2.5.29.15");
        customKeyUsageExtension.setValue(base64EncodedKUValue);
        customKeyUsageExtension.setCritical(true);

        // Set KeyUsage extension to api passthrough
        Extensions extensions = new Extensions();
        extensions.setCustomExtensions(Arrays.asList(customKeyUsageExtension));
```

```
        apiPassthrough.setExtensions(extensions);
        issueRequest.setApiPassthrough(apiPassthrough);

        // Issue the certificate.
        IssueCertificateResult issueResult = null;
        try {
            issueResult = client.issueCertificate(issueRequest);
        } catch (LimitExceededException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Retrieve and display the certificate ARN.
        String subordinateCertificateArn = issueResult.getCertificateArn();
        System.out.println("Subordinate Certificate Arn: " +
subordinateCertificateArn);

        return subordinateCertificateArn;
    }

    @SneakyThrows
    private static String generateKeyUsageValue() {
       KeyUsage keyUsage = new KeyUsage(X509KeyUsage.keyCertSign |
X509KeyUsage.cRLSign);
       byte[] kuBytes = keyUsage.getEncoded();
       return Base64.getEncoder().encodeToString(kuBytes);
    }

    private static String GetCertificate(String subordinateCertificateArn, String
rootCAArn, AWSACMPCA client) {

        // Create a request object.
        GetCertificateRequest certificateRequest = new GetCertificateRequest();

        // Set the certificate ARN.
```

```
        certificateRequest.withCertificateArn(subordinateCertificateArn);

        // Set the certificate authority ARN.
        certificateRequest.withCertificateAuthorityArn(rootCAArn);

        // Create waiter to wait on successful creation of the certificate file.
        Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
        try {
            getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the certificate and certificate chain.
        GetCertificateResult certificateResult = null;
        try {
            certificateResult = client.getCertificate(certificateRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

        // Get the certificate and certificate chain and display the result.
        String subordinateCertificate = certificateResult.getCertificate();
        System.out.println("Subordinate CA Certificate:");
        System.out.println(subordinateCertificate);

        return subordinateCertificate;
    }
```

```
   private static void ImportCertificateAuthorityCertificate(String
subordinateCertificate, String rootCertificate, String subordinateCAArn, AWSACMPCA
client) {

       // Create the request object and set the signed certificate, chain and CA ARN.
       ImportCertificateAuthorityCertificateRequest importRequest =
           new ImportCertificateAuthorityCertificateRequest();

       ByteBuffer certByteBuffer = stringToByteBuffer(subordinateCertificate);
       importRequest.setCertificate(certByteBuffer);

       ByteBuffer rootCACertByteBuffer = stringToByteBuffer(rootCertificate);
       importRequest.setCertificateChain(rootCACertByteBuffer);

       // Set the certificate authority ARN.
       importRequest.withCertificateAuthorityArn(subordinateCAArn);

       // Import the certificate.
       try {
           client.importCertificateAuthorityCertificate(importRequest);
       } catch (CertificateMismatchException ex) {
           throw ex;
       } catch (MalformedCertificateException ex) {
           throw ex;
       } catch (InvalidArnException ex) {
           throw ex;
       } catch (ResourceNotFoundException ex) {
           throw ex;
       } catch (RequestInProgressException ex) {
           throw ex;
       } catch (ConcurrentModificationException ex) {
           throw ex;
       } catch (RequestFailedException ex) {
           throw ex;
       }
       System.out.println("Product Attestation Intermediate (PAI) certificate
successfully imported.");
       System.out.println("Product Attestation Intermediate (PAI) activated
successfully.");
   }

   private static ByteBuffer stringToByteBuffer(final String string) {
       if (Objects.isNull(string)) {
           return null;
```

```
        }
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
        return ByteBuffer.wrap(bytes);
    }
}
```

Crear un certificado de atestación de dispositivos (DAC)

En este ejemplo de Java se muestra cómo utilizar la plantilla BlankEndEntityCertificate_ CriticalBasicConstraints _ APIPassthrough /V1 para crear un certificado de atestación de dispositivos de Matter. Debe generar un KeyUsage valor codificado en Base64 y pasarlo por un. CustomExtension

El ejemplo llama a la siguiente acción de API: Autoridad de certificación privada de AWS

- IssueCertificate

Si tiene problemas, consulte Solucione los errores de los AWS Private CA certificados compatibles con Matter en la sección Resolución de problemas.

```
package com.amazonaws.samples.matter;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
```

```
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;

import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

import lombok.SneakyThrows;

public class IssueDeviceAttestationCertificate {
   public static ByteBuffer stringToByteBuffer(final String string) {
      if (Objects.isNull(string)) {
          return null;
      }
      byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
      return ByteBuffer.wrap(bytes);
   }

   @SneakyThrows
   private static String generateKeyUsageValue() {
      KeyUsage keyUsage = new KeyUsage(X509KeyUsage.digitalSignature);
      byte[] kuBytes = keyUsage.getEncoded();
      return Base64.getEncoder().encodeToString(kuBytes);
   }

   public static void main(String[] args) throws Exception {

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
      } catch (Exception e) {
          throw new AmazonClientException("Cannot load your credentials from disk", e);
```

```
      }

      // Define the endpoint for your sample.
      String endpointRegion = "region";  // Substitute your region here, e.g. "ap-
southeast-2"
      String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
      EndpointConfiguration endpoint =
          new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
          .withEndpointConfiguration(endpoint)
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
          .build();

      // Create a certificate request:
      IssueCertificateRequest req = new IssueCertificateRequest();

      // Set the CA ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:region:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012");

      // Specify the certificate signing request (CSR) for the certificate to be signed
 and issued.
      String strCSR =
      "-----BEGIN CERTIFICATE REQUEST-----\n" +
      "base64-encoded certificate\n" +
      "-----END CERTIFICATE REQUEST-----\n";
      ByteBuffer csrByteBuffer = stringToByteBuffer(strCSR);
      req.setCsr(csrByteBuffer);

      // Specify the template for the issued certificate.
      req.withTemplateArn("arn:aws:acm-pca:::template/
BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough/V1");

      // Set the signing algorithm.
      req.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

      // Set the validity period for the certificate to be issued.
      Validity validity = new Validity();
      validity.withValue(10L);
      validity.withType("DAYS");
      req.withValidity(validity);
```

```
      // Set the idempotency token.
      req.setIdempotencyToken("1234");

      // Define custom attributes
      List<CustomAttribute> customAttributes = Arrays.asList(
         new CustomAttribute()
            .withObjectIdentifier("2.5.4.3")
            .withValue("Matter Test DAC 0001"),
         new CustomAttribute()
            .withObjectIdentifier("1.3.6.1.4.1.37244.2.1")
            .withValue("FFF1"),
         new CustomAttribute()
            .withObjectIdentifier("1.3.6.1.4.1.37244.2.2")
            .withValue("8000")
      );

      // Define a cert subject.
      ASN1Subject subject = new ASN1Subject();
      subject.setCustomAttributes(customAttributes);

      ApiPassthrough apiPassthrough = new ApiPassthrough();
      apiPassthrough.setSubject(subject);

      // Generate Base64 encoded extension value for ExtendedKeyUsage
      String base64EncodedKUValue = generateKeyUsageValue();

      // Generate custom extension
      CustomExtension customKeyUsageExtension = new CustomExtension();
      customKeyUsageExtension.setObjectIdentifier("2.5.29.15"); // KeyUsage Extension
OID
      customKeyUsageExtension.setValue(base64EncodedKUValue);
      customKeyUsageExtension.setCritical(true);

      Extensions extensions = new Extensions();
      extensions.setCustomExtensions(Arrays.asList(customKeyUsageExtension));
      apiPassthrough.setExtensions(extensions);
      req.setApiPassthrough(apiPassthrough);

      // Issue the certificate.
      IssueCertificateResult result = null;
      try {
         result = client.issueCertificate(req);
      } catch (LimitExceededException ex) {
```

```
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Retrieve and display the certificate ARN.
        String arn = result.getCertificateArn();
        System.out.println(arn);
    }
}
```

Active una CA raíz para los certificados operativos de nodo (NOC).

Este ejemplo de Java muestra cómo usar la Definición de Root CACertificate APIPassthrough _/ V1 plantilla para crear e instalar un certificado de CA Matter Root para emitirlo NOCs. La extensión AuthorityKeyIdentifier (AKI) es opcional para los certificados NOC Root CA. Para configurar una AKI, debe generar un valor de AKI codificado en Base64 y pasarlo por una. CustomExtension

El ejemplo llama a las siguientes acciones de API: Autoridad de certificación privada de AWS

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Si tiene problemas, consulte Solucione los errores de los AWS Private CA certificados compatibles con Matter en la sección Resolución de problemas.

```
package com.amazonaws.samples.matter;

import com.amazonaws.auth.AWSCredentials;
```

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.samples.GetCertificateAuthorityCertificate;
import com.amazonaws.auth.AWSStaticCredentialsProvider;


import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;
```

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import org.bouncycastle.asn1.x509.SubjectPublicKeyInfo;
import org.bouncycastle.cert.jcajce.JcaX509ExtensionUtils;
import org.bouncycastle.openssl.PEMParser;
import org.bouncycastle.pkcs.PKCS10CertificationRequest;
import org.bouncycastle.util.io.pem.PemReader;

import lombok.SneakyThrows;

public class RootCAActivation {
    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "ap-
southeast-2"

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1.37244.1.4")
                .withValue("CACACACA00000001")
        );

        // Define a CA subject.
```

```
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.EC_prime256v1);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
        configCA.withSubject(subject);

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.ROOT;

        // ** Execute core code samples for Root CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCAArn = CreateCertificateAuthority(configCA, CAtype, client);
        String csr = GetCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = IssueCertificate(rootCAArn, csr, client);
        String rootCertificate = GetCertificate(rootCertificateArn, rootCAArn, client);
        ImportCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

    private static AWSACMPCA ClientBuilder(String endpointRegion) {
        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load the credentials from the credential profiles file. " +
                    "Please make sure that your credentials file is at the correct " +
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                    e);
        }

        String endpointProtocol = "https://acm-pca." + endpointRegion +
".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
endpointRegion);

        // Create a client that you can use to make requests.
```

```
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        return client;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CertificateAuthorityType CAtype, AWSACMPCA client) {
        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);

        // Create the private CA.
        CreateCertificateAuthorityResult createCAResult = null;
        try {
            createCAResult = client.createCertificateAuthority(createCARequest);
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

        // Retrieve the ARN of the private CA.
        String rootCAArn = createCAResult.getCertificateAuthorityArn();
        System.out.println("Root CA Arn: " + rootCAArn);

        return rootCAArn;
    }

    private static String GetCertificateAuthorityCsr(String rootCAArn, AWSACMPCA
client) {

        // Create the CSR request object and set the CA ARN.
        GetCertificateAuthorityCsrRequest csrRequest = new
GetCertificateAuthorityCsrRequest();
        csrRequest.withCertificateAuthorityArn(rootCAArn);
```

```
        // Create waiter to wait on successful creation of the CSR file.
        Waiter<GetCertificateAuthorityCsrRequest> getCSRWaiter =
client.waiters().certificateAuthorityCSRCreated();
        try {
            getCSRWaiter.run(new WaiterParameters<>(csrRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        // Retrieve and display the CSR;
        String csr = csrResult.getCsr();
        System.out.println(csr);

        return csr;
    }

    @SneakyThrows
    private static String generateAuthorityKeyIdentifier(final String csrPEM) {
        PKCS10CertificationRequest csr = getPKCS10CertificationRequest(csrPEM);
        SubjectPublicKeyInfo spki = csr.getSubjectPublicKeyInfo();

        JcaX509ExtensionUtils extensionUtils = new JcaX509ExtensionUtils();
        byte[] akiBytes =
extensionUtils.createAuthorityKeyIdentifier(spki).getEncoded();
```

```
        return Base64.getEncoder().encodeToString(akiBytes);
    }

    @SneakyThrows
    private static PKCS10CertificationRequest getPKCS10CertificationRequest(final
 String csrPEM) {
        ByteArrayInputStream bais = new ByteArrayInputStream(csrPEM.getBytes());
        PemReader pemReader = new PemReader(new InputStreamReader(bais));
        PEMParser parser = new PEMParser(pemReader);
        Object o = parser.readObject();
        if (o instanceof PKCS10CertificationRequest) {
            return (PKCS10CertificationRequest) o;
        }
        return null;
    }

    private static String IssueCertificate(String rootCAArn, String csr, AWSACMPCA
 client) {

        // Create a certificate request:
        IssueCertificateRequest issueRequest = new IssueCertificateRequest();

        // Set the CA ARN.
        issueRequest.withCertificateAuthorityArn(rootCAArn);

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/
RootCACertificate_APIPassthrough/V1");

        ByteBuffer csrByteBuffer = stringToByteBuffer(csr);
        issueRequest.setCsr(csrByteBuffer);

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(3650L);
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

        // Set the idempotency token.
        issueRequest.setIdempotencyToken("1234");
```

```
        // Generate Base64 encoded extension value for AuthorityKeyIdentifier
        String base64EncodedExtValue = generateAuthorityKeyIdentifier(csr);

        // Generate custom extension
        CustomExtension customExtension = new CustomExtension();
        customExtension.setObjectIdentifier("2.5.29.35"); // AuthorityKeyIdentifier
Extension OID
        customExtension.setValue(base64EncodedExtValue);

        // Add custom extension to api-passthrough
        ApiPassthrough apiPassthrough = new ApiPassthrough();
        Extensions extensions = new Extensions();
        extensions.setCustomExtensions(Arrays.asList(customExtension));
        apiPassthrough.setExtensions(extensions);
        issueRequest.setApiPassthrough(apiPassthrough);

        // Issue the certificate.
        IssueCertificateResult issueResult = null;
        try {
            issueResult = client.issueCertificate(issueRequest);
        } catch (LimitExceededException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Retrieve and display the certificate ARN.
        String rootCertificateArn = issueResult.getCertificateArn();
        System.out.println("Root Certificate Arn: " + rootCertificateArn);

        return rootCertificateArn;
    }

    private static String GetCertificate(String rootCertificateArn, String rootCAArn,
AWSACMPCA client) {
```

```
        // Create a request object.
        GetCertificateRequest certificateRequest = new GetCertificateRequest();

        // Set the certificate ARN.
        certificateRequest.withCertificateArn(rootCertificateArn);

        // Set the certificate authority ARN.
        certificateRequest.withCertificateAuthorityArn(rootCAArn);

        // Create waiter to wait on successful creation of the certificate file.
        Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
        try {
            getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the certificate and certificate chain.
        GetCertificateResult certificateResult = null;
        try {
            certificateResult = client.getCertificate(certificateRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

        // Get the certificate and certificate chain and display the result.
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
```

```
    }

    private static void ImportCertificateAuthorityCertificate(String rootCertificate,
String rootCAArn, AWSACMPCA client) {

        // Create the request object and set the signed certificate, chain and CA ARN.
        ImportCertificateAuthorityCertificateRequest importRequest =
            new ImportCertificateAuthorityCertificateRequest();

        ByteBuffer certByteBuffer = stringToByteBuffer(rootCertificate);
        importRequest.setCertificate(certByteBuffer);

        importRequest.setCertificateChain(null);

        // Set the certificate authority ARN.
        importRequest.withCertificateAuthorityArn(rootCAArn);

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Root CA certificate successfully imported.");
        System.out.println("Root CA activated successfully.");
    }

    private static ByteBuffer stringToByteBuffer(final String string) {
        if (Objects.isNull(string)) {
            return null;
        }
```

```
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
        return ByteBuffer.wrap(bytes);
    }
}
```

Activar una CA subordinada para los certificados operativos de nodo (NOC)

En este ejemplo de Java se muestra cómo utilizar la BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1definición plantilla para emitir e instalar un certificado de CA de Matter Subordinate que se va a emitir. NOCs Debe generar un KeyUsage valor codificado en Base64 y pasarlo por un. CustomExtension

El ejemplo llama a las siguientes acciones de API: Autoridad de certificación privada de AWS

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate
- GetCertificateAuthorityCertificate

Si se producen problemas, consulta Solucione los errores de los AWS Private CA certificados compatibles con Matter la sección de solución de problemas.

```
package com.amazonaws.samples.matter;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;


import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
```

```
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateResult;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;
```

```
import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import lombok.SneakyThrows;

public class IntermediateCAActivation {

    public static void main(String[] args) throws Exception {
        // Place your own Root CA ARN here.
        String rootCAArn = "arn:aws:acm-pca:region:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012";

        // Define the endpoint region for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "ap-
southeast-2"

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1.37244.1.3")
                .withValue("CACACACA00000003")
        );

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
 CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.EC_prime256v1);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
        configCA.withSubject(subject);

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.SUBORDINATE;

        // ** Execute core code samples for Subordinate CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCertificate = GetCertificateAuthorityCertificate(rootCAArn, client);
        String subordinateCAArn = CreateCertificateAuthority(configCA, CAtype, client);
```

```
        String csr = GetCertificateAuthorityCsr(subordinateCAArn, client);
        String subordinateCertificateArn = IssueCertificate(rootCAArn, csr, client);
        String subordinateCertificate = GetCertificate(subordinateCertificateArn,
rootCAArn, client);
        ImportCertificateAuthorityCertificate(subordinateCertificate, rootCertificate,
subordinateCAArn, client);

    }

    private static AWSACMPCA ClientBuilder(String endpointRegion) {
        // Get your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load the credentials from the credential profiles file. " +
                    "Please make sure that your credentials file is at the correct " +
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                    e);
        }

        String endpointProtocol = "https://acm-pca." + endpointRegion +
".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        return client;
    }

    private static String GetCertificateAuthorityCertificate(String rootCAArn,
AWSACMPCA client) {
        // ** GetCertificateAuthorityCertificate **

        // Create a request object and set the certificate authority ARN,
```

```
        GetCertificateAuthorityCertificateRequest getCACertificateRequest =
        new GetCertificateAuthorityCertificateRequest();
        getCACertificateRequest.withCertificateAuthorityArn(rootCAArn);

        // Create a result object.
        GetCertificateAuthorityCertificateResult getCACertificateResult = null;
        try {
            getCACertificateResult =
client.getCertificateAuthorityCertificate(getCACertificateRequest);
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        }

        // Get and display the certificate information.
        String rootCertificate = getCACertificateResult.getCertificate();
        System.out.println("Root CA Certificate / Certificate Chain:");
        System.out.println(rootCertificate);

        return rootCertificate;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CertificateAuthorityType CAtype, AWSACMPCA client) {
        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);

        // Create the private CA.
        CreateCertificateAuthorityResult createCAResult = null;
        try {
            createCAResult = client.createCertificateAuthority(createCARequest);
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
```

```
        }

        // Retrieve the ARN of the private CA.
        String subordinateCAArn = createCAResult.getCertificateAuthorityArn();
        System.out.println("Subordinate CA Arn: " + subordinateCAArn);

        return subordinateCAArn;
    }

    private static String GetCertificateAuthorityCsr(String subordinateCAArn, AWSACMPCA
client) {

        // Create the CSR request object and set the CA ARN.
        GetCertificateAuthorityCsrRequest csrRequest = new
GetCertificateAuthorityCsrRequest();
        csrRequest.withCertificateAuthorityArn(subordinateCAArn);

        // Create waiter to wait on successful creation of the CSR file.
        Waiter<GetCertificateAuthorityCsrRequest> getCSRWaiter =
client.waiters().certificateAuthorityCSRCreated();
        try {
            getCSRWaiter.run(new WaiterParameters<>(csrRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch(AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Get the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
```

```
        // Get and display the CSR;
        String csr = csrResult.getCsr();
        System.out.println("Subordinate CSR:");
        System.out.println(csr);

        return csr;
    }

    private static String IssueCertificate(String rootCAArn, String csr, AWSACMPCA
 client) {

        // Create a certificate request:
        IssueCertificateRequest issueRequest = new IssueCertificateRequest();

        // Set the issuing CA ARN.
        issueRequest.withCertificateAuthorityArn(rootCAArn);

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/
BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1");

        ByteBuffer csrByteBuffer = stringToByteBuffer(csr);
        issueRequest.setCsr(csrByteBuffer);

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(730L); // Approximately two years
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

        // Set the idempotency token.
        issueRequest.setIdempotencyToken("1234");

        ApiPassthrough apiPassthrough = new ApiPassthrough();

        // Generate base64 encoded extension value for ExtendedKeyUsage
        String base64EncodedKUValue = generateKeyUsageValue();

        // Generate custom extension
        CustomExtension customKeyUsageExtension = new CustomExtension();
```

```
        customKeyUsageExtension.setObjectIdentifier("2.5.29.15");
        customKeyUsageExtension.setValue(base64EncodedKUValue);
        customKeyUsageExtension.setCritical(true);

        // Set KeyUsage extension to api passthrough
        Extensions extensions = new Extensions();
        extensions.setCustomExtensions(Arrays.asList(customKeyUsageExtension));
        apiPassthrough.setExtensions(extensions);
        issueRequest.setApiPassthrough(apiPassthrough);

        // Issue the certificate.
        IssueCertificateResult issueResult = null;
        try {
            issueResult = client.issueCertificate(issueRequest);
        } catch (LimitExceededException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Get and display the certificate ARN.
        String subordinateCertificateArn = issueResult.getCertificateArn();
        System.out.println("Subordinate Certificate Arn: " +
subordinateCertificateArn);

        return subordinateCertificateArn;
    }

    @SneakyThrows
    private static String generateKeyUsageValue() {
       KeyUsage keyUsage = new KeyUsage(X509KeyUsage.keyCertSign |
X509KeyUsage.cRLSign);
       byte[] kuBytes = keyUsage.getEncoded();
       return Base64.getEncoder().encodeToString(kuBytes);
    }
```

```
   private static String GetCertificate(String subordinateCertificateArn, String
rootCAArn, AWSACMPCA client) {

       // Create a request object.
       GetCertificateRequest certificateRequest = new GetCertificateRequest();

       // Set the certificate ARN.
       certificateRequest.withCertificateArn(subordinateCertificateArn);

       // Set the certificate authority ARN.
       certificateRequest.withCertificateAuthorityArn(rootCAArn);

       // Create waiter to wait on successful creation of the certificate file.
       Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
       try {
           getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
       } catch (WaiterUnrecoverableException e) {
           //Explicit short circuit when the recourse transitions into
           //an undesired state.
       } catch (WaiterTimedOutException e) {
           //Failed to transition into desired state even after polling.
       } catch (AWSACMPCAException e) {
           //Unexpected service exception.
       }

       // Get the certificate and certificate chain.
       GetCertificateResult certificateResult = null;
       try {
           certificateResult = client.getCertificate(certificateRequest);
       } catch (RequestInProgressException ex) {
           throw ex;
       } catch (RequestFailedException ex) {
           throw ex;
       } catch (ResourceNotFoundException ex) {
           throw ex;
       } catch (InvalidArnException ex) {
           throw ex;
       } catch (InvalidStateException ex) {
           throw ex;
       }

       // Get the certificate and certificate chain and display the result.
       String subordinateCertificate = certificateResult.getCertificate();
```

```
        System.out.println("Subordinate CA Certificate:");
        System.out.println(subordinateCertificate);

        return subordinateCertificate;
    }

    private static void ImportCertificateAuthorityCertificate(String
subordinateCertificate, String rootCertificate, String subordinateCAArn, AWSACMPCA
client) {

        // Create the request object and set the signed certificate, chain and CA ARN.
        ImportCertificateAuthorityCertificateRequest importRequest =
            new ImportCertificateAuthorityCertificateRequest();

        ByteBuffer certByteBuffer = stringToByteBuffer(subordinateCertificate);
        importRequest.setCertificate(certByteBuffer);

        ByteBuffer rootCACertByteBuffer = stringToByteBuffer(rootCertificate);
        importRequest.setCertificateChain(rootCACertByteBuffer);

        // Set the certificate authority ARN.
        importRequest.withCertificateAuthorityArn(subordinateCAArn);

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
        System.out.println("Subordinate CA certificate successfully imported.");
        System.out.println("Subordinate CA activated successfully.");
    }
```

```
    private static ByteBuffer stringToByteBuffer(final String string) {
        if (Objects.isNull(string)) {
            return null;
        }
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
        return ByteBuffer.wrap(bytes);
    }
}
```

Crear un certificado operativo de nodo (NOC)

En este ejemplo de Java, se muestra cómo utilizar la plantilla BlankEndEntityCertificate_ CriticalBasicConstraints _ APIPassthrough /V1 para crear un certificado operativo de Matter Node. Debe generar un KeyUsage valor codificado en Base64 y pasarlo por un. CustomExtension

El ejemplo llama a la siguiente acción de API: Autoridad de certificación privada de AWS

- IssueCertificate

Si tiene problemas, consulte Solucione los errores de los AWS Private CA certificados compatibles con Matter en la sección Resolución de problemas.

```
package com.amazonaws.samples.matter;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
```

```
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;

import org.bouncycastle.asn1.x509.ExtendedKeyUsage;
import org.bouncycastle.asn1.x509.KeyPurposeId;
import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

import lombok.SneakyThrows;

public class IssueNodeOperatingCertificate {
   public static ByteBuffer stringToByteBuffer(final String string) {
      if (Objects.isNull(string)) {
          return null;
      }
      byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
      return ByteBuffer.wrap(bytes);
   }

   @SneakyThrows
   private static String generateExtendedKeyUsageValue() {
      KeyPurposeId[] keyPurposeIds = new KeyPurposeId[]
 { KeyPurposeId.id_kp_clientAuth, KeyPurposeId.id_kp_serverAuth };
      ExtendedKeyUsage eku = new ExtendedKeyUsage(keyPurposeIds);
      byte[] ekuBytes = eku.getEncoded();
      return Base64.getEncoder().encodeToString(ekuBytes);
   }

   @SneakyThrows
   private static String generateKeyUsageValue() {
```

```
        KeyUsage keyUsage = new KeyUsage(X509KeyUsage.digitalSignature);
        byte[] kuBytes = keyUsage.getEncoded();
        return Base64.getEncoder().encodeToString(kuBytes);
    }

    public static void main(String[] args) throws Exception {

        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load your credentials from disk", e);
        }

        // Define the endpoint for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "ap-
southeast-2"
        String endpointProtocol = "https://acm-pca." + endpointRegion +
 ".amazonaws.com/";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        // Create a certificate request:
        IssueCertificateRequest req = new IssueCertificateRequest();

        // Set the CA ARN.
        req.withCertificateAuthorityArn("arn:aws:acm-pca:region:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012");

        // Specify the certificate signing request (CSR) for the certificate to be signed
 and issued.
        String strCSR =
        "-----BEGIN CERTIFICATE REQUEST-----\n" +
        "base64-encoded certificate\n" +
        "-----END CERTIFICATE REQUEST-----\n";
        ByteBuffer csrByteBuffer = stringToByteBuffer(strCSR);
```

```
      req.setCsr(csrByteBuffer);

      // Specify the template for the issued certificate.
      req.withTemplateArn("arn:aws:acm-pca:::template/
BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough/V1");

      // Set the signing algorithm.
      req.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

      // Set the validity period for the certificate to be issued.
      Validity validity = new Validity();
      validity.withValue(10L);
      validity.withType("DAYS");
      req.withValidity(validity);

      // Set the idempotency token.
      req.setIdempotencyToken("1234");

      // Define custom attributes
      List<CustomAttribute> customAttributes = Arrays.asList(
         new CustomAttribute()
            .withObjectIdentifier("1.3.6.1.4.1.37244.1.1")
            .withValue("DEDEDEDE00010001"),
         new CustomAttribute()
            .withObjectIdentifier("1.3.6.1.4.1.37244.1.5")
            .withValue("FAB000000000001D")
      );

      // Define a cert subject.
      ASN1Subject subject = new ASN1Subject();
      subject.setCustomAttributes(customAttributes);

      ApiPassthrough apiPassthrough = new ApiPassthrough();
      apiPassthrough.setSubject(subject);

      // Generate Base64 encoded extension value for ExtendedKeyUsage
      String base64EncodedKUValue = generateKeyUsageValue();

      // Generate custom extension
      CustomExtension customKeyUsageExtension = new CustomExtension();
      customKeyUsageExtension.setObjectIdentifier("2.5.29.15");
      customKeyUsageExtension.setValue(base64EncodedKUValue);
      customKeyUsageExtension.setCritical(true);
```

```
      // Generate Base64 encoded extension value for ExtendedKeyUsage
      String base64EncodedEKUValue = generateExtendedKeyUsageValue();

      CustomExtension customExtendedKeyUsageExtension = new CustomExtension();
      customExtendedKeyUsageExtension.setObjectIdentifier("2.5.29.37"); //
 ExtendedKeyUsage Extension OID
      customExtendedKeyUsageExtension.setValue(base64EncodedEKUValue);
      customExtendedKeyUsageExtension.setCritical(true);

      // Set KeyUsage and ExtendedKeyUsage extension to api-passthrough
      Extensions extensions = new Extensions();
      extensions.setCustomExtensions(Arrays.asList(customKeyUsageExtension,
 customExtendedKeyUsageExtension));
      apiPassthrough.setExtensions(extensions);
      req.setApiPassthrough(apiPassthrough);

      // Issue the certificate.
      IssueCertificateResult result = null;
      try {
         result = client.issueCertificate(req);
      } catch (LimitExceededException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidArgsException ex) {
         throw ex;
      } catch (MalformedCSRException ex) {
         throw ex;
      }

      // Retrieve and display the certificate ARN.
      String arn = result.getCertificateArn();
      System.out.println(arn);
   }
}
```

Úselo Autoridad de certificación privada de AWS para implementar certificados mDL

Puede utilizar la AWS Private Certificate Authority API para crear certificados que se ajusten a la norma ISO/IEC para permisos de conducir móviles (mDL). Esta norma establece las especificaciones de interfaz para la implementación de un permiso de conducir en asociación con un dispositivo móvil, incluidas las configuraciones de los certificados.

Temas

- Activar un certificado de autoridad emisora de certificados (IACA)
- Crear un certificado de firmante de documentos

Activar un certificado de autoridad emisora de certificados (IACA)

En este ejemplo de Java se muestra cómo utilizar la BlankRootCACertificate_ PathLen 0_ /V1 definición APIPassthrough plantilla para crear e instalar un certificado de autoridad certificadora (IACA) de la autoridad emisora de certificados (IACA) que cumpla con la norma ISO/IEC mDL. Debe generar valores codificados en base64 para`KeyUsage`, y, y pasarlos. `IssuerAlternativeName` `CRLDistributionPoint` `CustomExtensions`

> **Note**
>
> El certificado de enlace de la IACA establece una ruta de confianza desde el antiguo certificado raíz de la IACA hasta el nuevo certificado raíz de la IACA. La autoridad emisora puede generar y distribuir un certificado de enlace de la IACA durante el proceso de cambio de clave de la IACA. No puede emitir un certificado de enlace de la IACA mediante un certificado raíz de la IACA con un conjunto. `pathLen=0`

El ejemplo llama a las siguientes acciones de Autoridad de certificación privada de AWS API:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

```
package com.amazonaws.samples.mdl;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.Objects;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
```

```
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import org.bouncycastle.asn1.x509.GeneralNames;
import org.bouncycastle.asn1.x509.GeneralName;
import org.bouncycastle.asn1.x509.CRLDistPoint;
import org.bouncycastle.asn1.x509.DistributionPoint;
import org.bouncycastle.asn1.x509.DistributionPointName;
import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

import lombok.SneakyThrows;

public class IssuingAuthorityCertificateAuthorityActivation {
    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = null;  // Substitute your region here, e.g. "ap-
southeast-2"
        if (endpointRegion == null) throw new Exception("Region cannot be null");

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject()
                .withCountry("US") // mDL spec requires ISO 3166-1-alpha-2 country code
 e.g. "US"
                .withCommonName("mDL Test IACA");

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
 CertificateAuthorityConfiguration()
                .withKeyAlgorithm(KeyAlgorithm.EC_prime256v1)
                .withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA)
                .withSubject(subject);

        // Define a certificate authority type
```

```
        CertificateAuthorityType CAType = CertificateAuthorityType.ROOT;

        // Execute core code samples for Root CA activation in sequence
        AWSACMPCA client = buildClient(endpointRegion);
        String rootCAArn = createCertificateAuthority(configCA, CAType, client);
        String csr = getCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = issueCertificate(rootCAArn, csr, client);
        String rootCertificate = getCertificate(rootCertificateArn, rootCAArn, client);
        importCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

    private static AWSACMPCA buildClient(String endpointRegion) {
        // Get your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load your credentials from disk",
e);
        }

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
                .withRegion(endpointRegion)
                .withCredentials(new AWSStaticCredentialsProvider(credentials))
                .build();

        return client;
    }

    private static String createCertificateAuthority(CertificateAuthorityConfiguration
configCA, CertificateAuthorityType CAtype, AWSACMPCA client) {
        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest()
                .withCertificateAuthorityConfiguration(configCA)
                .withIdempotencyToken("123987")
                .withCertificateAuthorityType(CAtype);

        // Create the private CA.
        CreateCertificateAuthorityResult createCAResult = null;
        try {
            createCAResult = client.createCertificateAuthority(createCARequest);
```

```
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

        // Get the ARN of the private CA.
        String rootCAArn = createCAResult.getCertificateAuthorityArn();
        System.out.println("Issuing Authority Certificate Authority (IACA) Arn: " +
rootCAArn);

        return rootCAArn;
    }

    private static String getCertificateAuthorityCsr(String rootCAArn, AWSACMPCA
client) {

        // Create the CSR request object and set the CA ARN.
        GetCertificateAuthorityCsrRequest csrRequest = new
GetCertificateAuthorityCsrRequest()
                .withCertificateAuthorityArn(rootCAArn);

        // Create waiter to wait on successful creation of the CSR file.
        Waiter<GetCertificateAuthorityCsrRequest> getCSRWaiter =
client.waiters().certificateAuthorityCSRCreated();
        try {
            getCSRWaiter.run(new WaiterParameters<>(csrRequest));
        } catch (WaiterUnrecoverableException e) {
            // Explicit short circuit when the recourse transitions into
            // an undesired state.
        } catch (WaiterTimedOutException e) {
            // Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            // Unexpected service exception.
        }

        // Get the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
```

```
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        // Get and display the CSR;
        String csr = csrResult.getCsr();
        System.out.println("CSR:");
        System.out.println(csr);

        return csr;
    }

    @SneakyThrows
    private static String issueCertificate(String rootCAArn, String csr, AWSACMPCA
 client) {
        IssueCertificateRequest issueRequest = new IssueCertificateRequest()
                .withCertificateAuthorityArn(rootCAArn)
                .withTemplateArn("arn:aws:acm-pca:::template/
BlankRootCACertificate_PathLen0_APIPassthrough/V1")
                .withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA)
                .withIdempotencyToken("1234");

        // Set the CSR.
        ByteBuffer csrByteBuffer = stringToByteBuffer(csr);
        issueRequest.setCsr(csrByteBuffer);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity()
                .withValue(3650L)
                .withType("DAYS");
        issueRequest.setValidity(validity);

        // Generate base64 encoded extension value for KeyUsage
        KeyUsage keyUsage = new KeyUsage(X509KeyUsage.keyCertSign +
 X509KeyUsage.cRLSign);
        byte[] kuBytes = keyUsage.getEncoded();
        String base64EncodedKUValue = Base64.getEncoder().encodeToString(kuBytes);

        CustomExtension keyUsageCustomExtension = new CustomExtension()
                .withObjectIdentifier("2.5.29.15") // KeyUsage Extension OID
```

```
                .withValue(base64EncodedKUValue)
                .withCritical(true);

        // Generate base64 encoded extension value for IssuerAlternativeName
        GeneralNames issuerAlternativeName = new GeneralNames(new
 GeneralName(GeneralName.uniformResourceIdentifier, "https://issuer-alternative-
name.com"));
        String base64EncodedIANValue =
 Base64.getEncoder().encodeToString(issuerAlternativeName.getEncoded());

        CustomExtension ianCustomExtension = new CustomExtension()
                .withValue(base64EncodedIANValue)
                .withObjectIdentifier("2.5.29.18"); // IssuerAlternativeName Extension
 OID

        // Generate base64 encoded extension value for CRLDistributionPoint
        CRLDistPoint crlDistPoint = new CRLDistPoint(new DistributionPoint[]{new
 DistributionPoint(new DistributionPointName(
                new GeneralNames(new GeneralName(GeneralName.uniformResourceIdentifier,
 "dummycrl.crl"))), null, null)});
        String base64EncodedCDPValue =
 Base64.getEncoder().encodeToString(crlDistPoint.getEncoded());

        CustomExtension cdpCustomExtension = new CustomExtension()
                .withValue(base64EncodedCDPValue)
                .withObjectIdentifier("2.5.29.31"); // CRLDistributionPoint Extension
 OID

        // Add custom extension to api-passthrough
        Extensions extensions = new Extensions()
                .withCustomExtensions(Arrays.asList(keyUsageCustomExtension,
 ianCustomExtension, cdpCustomExtension));
        ApiPassthrough apiPassthrough = new ApiPassthrough()
                .withExtensions(extensions);
        issueRequest.setApiPassthrough(apiPassthrough);

        // Issue the certificate.
        IssueCertificateResult issueResult = null;
        try {
            issueResult = client.issueCertificate(issueRequest);
        } catch (LimitExceededException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
```

```
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Get and display the certificate ARN.
        String rootCertificateArn = issueResult.getCertificateArn();
        System.out.println("mDL IACA Certificate Arn: " + rootCertificateArn);

        return rootCertificateArn;
    }

    private static String getCertificate(String rootCertificateArn, String rootCAArn,
AWSACMPCA client) {

        // Create a request object.
        GetCertificateRequest certificateRequest = new GetCertificateRequest()
                .withCertificateArn(rootCertificateArn)
                .withCertificateAuthorityArn(rootCAArn);

        // Create waiter to wait on successful creation of the certificate file.
        Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
        try {
            getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
        } catch (WaiterUnrecoverableException e) {
            // Explicit short circuit when the recourse transitions into
            // an undesired state.
        } catch (WaiterTimedOutException e) {
            // Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            // Unexpected service exception.
        }

        // Get the certificate and certificate chain.
        GetCertificateResult certificateResult = null;
        try {
            certificateResult = client.getCertificate(certificateRequest);
        } catch (RequestInProgressException ex) {
```

```
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

        // Get the certificate and certificate chain and display the result.
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
    }

    private static void importCertificateAuthorityCertificate(String rootCertificate,
String rootCAArn, AWSACMPCA client) {

        // Create the request object and set the signed certificate, chain and CA ARN.
        ImportCertificateAuthorityCertificateRequest importRequest =
                new ImportCertificateAuthorityCertificateRequest()
                        .withCertificateChain(null)
                        .withCertificateAuthorityArn(rootCAArn);

        ByteBuffer certByteBuffer = stringToByteBuffer(rootCertificate);
        importRequest.setCertificate(certByteBuffer);

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
```

```
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Root CA certificate successfully imported.");
        System.out.println("Root CA activated successfully.");
    }

    private static ByteBuffer stringToByteBuffer(final String string) {
        if (Objects.isNull(string)) {
            return null;
        }
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
        return ByteBuffer.wrap(bytes);
    }
}
```

Crear un certificado de firmante de documentos

En este ejemplo de Java se muestra cómo utilizar la plantilla BlankEndEntityCertificateAPIPassthrough_/V1 para crear un certificado de firma de documentos que cumpla con la norma ISO/IEC mDL. Debe generar valores codificados en base64 para, y pasarlos. `KeyUsage` `IssuerAlternativeName` `CRLDistributionPoint` `CustomExtensions`

El ejemplo llama a la siguiente acción de API: Autoridad de certificación privada de AWS

- IssueCertificate

```
package com.amazonaws.samples.mdl;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.Objects;
```

```
import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.ExtendedKeyUsage;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;

import org.bouncycastle.asn1.x509.GeneralNames;
import org.bouncycastle.asn1.x509.GeneralName;
import org.bouncycastle.asn1.x509.CRLDistPoint;
import org.bouncycastle.asn1.x509.DistributionPoint;
import org.bouncycastle.asn1.x509.DistributionPointName;
import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

public class IssueDocumentSignerCertificate {
    public static ByteBuffer stringToByteBuffer(final String string) {
        if (Objects.isNull(string)) {
            return null;
        }
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
        return ByteBuffer.wrap(bytes);
    }

    public static void main(String[] args) throws Exception {

        // Get your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
```

```
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load your credentials from disk",
 e);
        }

        // Create a client that you can use to make requests.
        String endpointRegion = null;  // Substitute your region here, e.g. "ap-
southeast-2"
        if (endpointRegion == null) throw new Exception("Region cannot be null");

        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
                .withRegion(endpointRegion)
                .withCredentials(new AWSStaticCredentialsProvider(credentials))
                .build();

        // Create a certificate request:
        String caArn = null;
        if (caArn == null) throw new Exception("Certificate authority ARN cannot be
 null");

        IssueCertificateRequest req = new IssueCertificateRequest()
                .withCertificateAuthorityArn(caArn)
                .withTemplateArn("arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APIPassthrough/V1")
                .withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA)
                .withIdempotencyToken("1234");

        // Specify the certificate signing request (CSR) for the certificate to be
 signed and issued.
        // Format: "-----BEGIN CERTIFICATE REQUEST-----\n" +
        //         "base64-encoded certificate\n" +
        //         "-----END CERTIFICATE REQUEST-----\n";
        String strCSR = null;
        if (strCSR == null) throw new Exception("CSR string cannot be null");

        ByteBuffer csrByteBuffer = stringToByteBuffer(strCSR);
        req.setCsr(csrByteBuffer);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity()
                .withValue(365L)
                .withType("DAYS");
        req.setValidity(validity);
```

```
        // Define a cert subject.
        ASN1Subject subject = new ASN1Subject()
                .withCountry("US") // mDL spec requires ISO 3166-1-alpha-2 country code
 e.g. "US"
                .withCommonName("mDL Test DS");

        ApiPassthrough apiPassthrough = new ApiPassthrough()
                .withSubject(subject);

        // Generate base64 encoded extension value for KeyUsage
        KeyUsage keyUsage = new KeyUsage(X509KeyUsage.digitalSignature);
        byte[] kuBytes = keyUsage.getEncoded();
        String base64EncodedKUValue = Base64.getEncoder().encodeToString(kuBytes);

        CustomExtension customKeyUsageExtension = new CustomExtension()
                .withObjectIdentifier("2.5.29.15") // KeyUsage Extension OID
                .withValue(base64EncodedKUValue)
                .withCritical(true);

        // Generate base64 encoded extension value for IssuerAlternativeName
        GeneralNames issuerAlternativeName = new GeneralNames(new
 GeneralName(GeneralName.uniformResourceIdentifier, "https://issuer-alternative-
name.com"));
        String base64EncodedIANValue =
 Base64.getEncoder().encodeToString(issuerAlternativeName.getEncoded());

        CustomExtension ianCustomExtension = new CustomExtension()
                .withValue(base64EncodedIANValue)
                .withObjectIdentifier("2.5.29.18"); // IssuerAlternativeName Extension
 OID

        // Generate base64 encoded extension value for CRLDistributionPoint
        CRLDistPoint crlDistPoint = new CRLDistPoint(new DistributionPoint[]{new
 DistributionPoint(new DistributionPointName(
                new GeneralNames(new GeneralName(GeneralName.uniformResourceIdentifier,
 "dummycrl.crl"))), null, null)});
        String base64EncodedCDPValue =
 Base64.getEncoder().encodeToString(crlDistPoint.getEncoded());

        CustomExtension cdpCustomExtension = new CustomExtension()
                .withValue(base64EncodedCDPValue)
                .withObjectIdentifier("2.5.29.31"); // CRLDistributionPoint Extension
 OID
```

```
        // Generate EKU
        ExtendedKeyUsage eku = new ExtendedKeyUsage()
                .withExtendedKeyUsageObjectIdentifier("1.0.18013.5.1.2"); // EKU value
 reserved for mDL DS

        // Set KeyUsage, ExtendedKeyUsage, IssuerAlternativeName, CRL Distribution
 Point extensions to api-passthrough
        Extensions extensions = new Extensions()
                .withCustomExtensions(Arrays.asList(customKeyUsageExtension,
 ianCustomExtension, cdpCustomExtension))
                .withExtendedKeyUsage(Arrays.asList(eku));
        apiPassthrough.setExtensions(extensions);
        req.setApiPassthrough(apiPassthrough);

        // Issue the certificate.
        IssueCertificateResult result = null;
        try {
            result = client.issueCertificate(req);
        } catch (LimitExceededException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Get and display the certificate ARN.
        String arn = result.getCertificateArn();
        System.out.println("mDL DS Certificate Arn: " + arn);
    }
}
```

Diseñe su solución para AWS Private CA

Autoridad de certificación privada de AWS le ofrece un control total y basado en la nube sobre la PKI (infraestructura de clave pública) privada de su organización, desde los certificados de una entidad emisora de certificados (CA) raíz, pasando por los subordinados CAs, hasta los certificados de la entidad final. Una planificación exhaustiva es esencial para una PKI que sea segura, se pueda mantener, sea ampliable y adecuada a las necesidades de su organización. Esta sección proporciona orientación sobre el diseño de una jerarquía de entidades de certificación, la administración de los ciclos de vida de los certificados de entidad final privada y de entidad de certificación privada y la aplicación de las prácticas recomendadas para la seguridad.

En esta sección se describe cómo prepararse Autoridad de certificación privada de AWS para su uso antes de crear una entidad de certificación (CA) privada. También se explica la opción de permitir la revocación mediante el Protocolo de estado de certificados en línea (OCSP) o una lista de revocación de certificados (CRL).

Además, debe determinar si su organización prefiere alojar sus credenciales de CA raíz privada en las instalaciones y no en ellas AWS. En ese caso, debe configurar y proteger una PKI privada autogestionada antes de utilizarla. Autoridad de certificación privada de AWS En este escenario, se crea una CA subordinada Autoridad de certificación privada de AWS respaldada por una CA principal externa a. Autoridad de certificación privada de AWS Para obtener más información, consulte Instalación de un certificado de entidad de certificación subordinada firmado por una entidad de certificación principal externa.

Temas

- Diseñe una jerarquía de CA
- Administre el ciclo de vida de la CA privada
- Planifique el método de revocación AWS Private CA de su certificado
- AWS Private CA Comprenda los modos de CA
- Planifique la resiliencia en AWS Private CA

Diseñe una jerarquía de CA

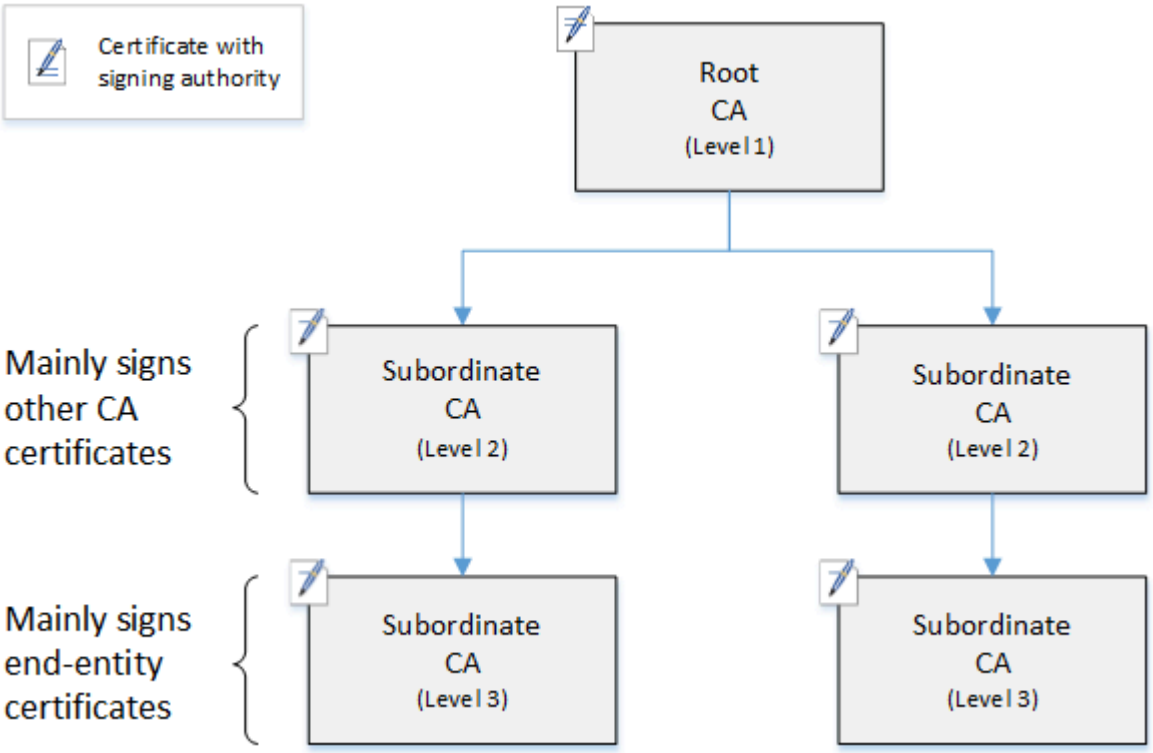
Con Autoridad de certificación privada de AWSél, puede crear una jerarquía de autoridades de certificación de hasta cinco niveles. La entidad de certificación raíz, en la parte superior de un árbol de jerarquía, puede tener cualquier número de ramas. La CA raíz puede tener hasta cuatro niveles

de subordinados CAs en cada rama. También puede crear varias jerarquías, cada una con su propia raíz.

Una jerarquía de entidades de certificación bien diseñada ofrece los siguientes beneficios:

- Controles de seguridad detallados adecuados para cada entidad de certificación
- División de tareas administrativas para mejorar el equilibrio de carga y la seguridad
- Se utiliza CAs con un fideicomiso revocable y limitado para las operaciones diarias
- Períodos de validez y límites de ruta de certificado

El siguiente diagrama ilustra una jerarquía de entidad de certificación simple de tres niveles.



Cada CA del árbol está respaldada por un certificado X.509 v3 con autoridad de firma (simbolizada por el icono). pen-and-paper Esto significa que, por lo tanto CAs, pueden firmar otros certificados que estén subordinados a ellos. Cuando una entidad de certificación firma un certificado de entidad de certificación de nivel inferior, confiere una autoridad limitada y revocable al certificado firmado. La entidad de certificación raíz del nivel 1 firma certificados de entidad de certificación subordinados de alto nivel en el nivel 2. Estos CAs, a su vez, firman los certificados del nivel 3 que utilizan los administradores de la PKI (infraestructura de clave pública) que administran los certificados de la entidad final. CAs

La seguridad en una jerarquía de entidades de certificación debe configurarse para que sea más fuerte en la parte superior del árbol. Esta disposición protege el certificado de entidad de certificación raíz y su clave privada. La entidad emisora de certificados raíz garantiza la confianza de todos los certificados subordinados CAs y de la entidad final situados debajo de ella. Aunque se pueden producir daños localizados al poner en riesgo un certificado de entidad final, la puesta en riesgo de la raíz destruye la confianza en toda la PKI. La raíz y la subordinada de alto nivel CAs se utilizan con poca frecuencia (normalmente para firmar otros certificados de CA). Por consiguiente, se controlan y auditan rigurosamente para garantizar un menor riesgo de peligro. En los niveles inferiores de la jerarquía, la seguridad es menos restrictiva. Este enfoque permite las tareas administrativas rutinarias de emitir y revocar certificados de entidad final para usuarios, alojamientos de equipos y servicios de software.

> **Note**
>
> El uso de una entidad de certificación raíz para firmar un certificado subordinado es un evento poco habitual que se produce en solo algunas circunstancias:
>
> - Cuando se crea la PKI
> - Cuando se debe reemplazar una entidad de certificación de alto nivel
> - Cuando se debe configurar un respondedor de lista de revocación de certificados (CRL) o del protocolo de estado de certificados en línea (OCSP)
>
> La raíz y otros protocolos de control de acceso de alto nivel CAs requieren procesos operativos y protocolos de control de acceso muy seguros.

Temas

- Valide los certificados de la entidad final
- Planifique la estructura de una jerarquía de CA
- Establezca restricciones de longitud en la ruta de certificación

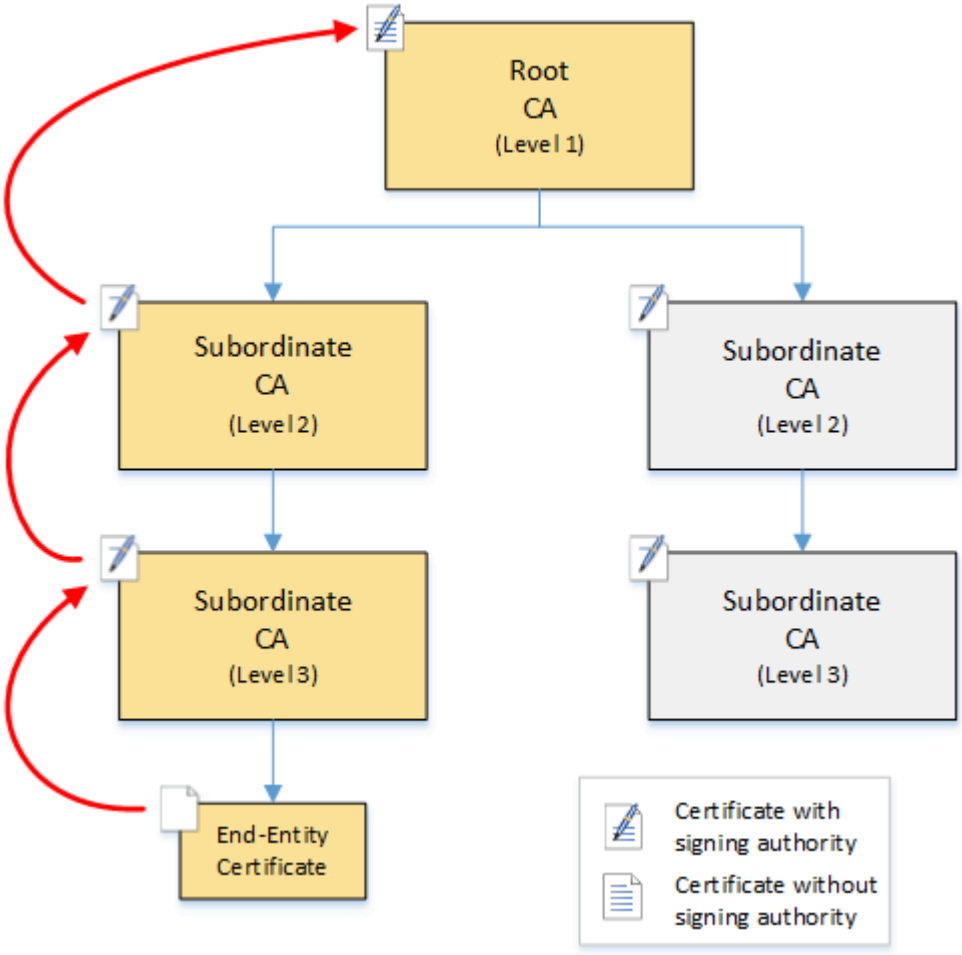
Valide los certificados de la entidad final

La confianza en los certificados de la entidad final proviene de una ruta de certificación que va desde la entidad emisora subordinada CAs hasta una entidad emisora raíz. Cuando un navegador web u otro cliente se presenta con un certificado de entidad final, intenta crear una cadena de

confianza. Por ejemplo, podría comprobar que el nombre distintivo del emisor del certificado y el nombre distintivo del sujeto coinciden con los campos correspondientes del certificado de entidad de certificación emisor. La coincidencia continuaría en cada nivel sucesivo de la jerarquía hasta que el cliente alcance una raíz de confianza contenida en su almacén de confianza.

El almacén de confianza es una biblioteca de datos de confianza CAs que contiene el navegador o el sistema operativo. Para una PKI privada, la TI de su organización debe asegurarse de que cada navegador o sistema haya agregado previamente la entidad de certificación raíz privada a su almacén de confianza. De lo contrario, la ruta de certificación no se puede validar, lo que da lugar a errores de cliente.

El siguiente diagrama muestra la ruta de validación que sigue un explorador cuando se presenta un certificado X.509 de entidad final. Tenga en cuenta que el certificado de entidad final carece de autoridad de firma y sólo sirve para autenticar a la entidad que lo posee.



El explorador inspecciona el certificado de entidad final. El explorador encuentra que el certificado ofrece una firma de entidad de certificación subordinada (nivel 3) como credencial de confianza.

Los certificados del subordinado CAs deben incluirse en el mismo archivo PEM. Alternativamente, también pueden estar en un archivo independiente que contenga los certificados que componen la cadena de confianza. Al encontrarlos, el navegador comprueba el certificado de entidad de certificación subordinada (nivel 3) y encuentra que ofrece una firma de entidad de certificación subordinada (nivel 2). A su vez, la entidad de certificación subordinada (nivel 2) ofrece una firma de la entidad de certificación raíz (nivel 1) como credencial de confianza. Si el explorador encuentra una copia del certificado de entidad de certificación raíz privada preinstalado en su almacén de confianza, valida el certificado de entidad final como de confianza.

Normalmente, el explorador también comprueba cada certificado con una lista de revocación de certificados (CRL). Se rechaza un certificado caducado, revocado o mal configurado y se produce un error en la validación.

Planifique la estructura de una jerarquía de CA

En general, la jerarquía de entidad de certificación debe reflejar la estructura de la organización. Apunte a una longitud de la ruta (es decir, número de niveles de entidades de certificación) no mayor de lo necesario para delegar roles administrativos y de seguridad. Agregar una entidad de certificación a la jerarquía significa aumentar el número de certificados en la ruta de certificación, lo que aumenta el tiempo de validación. Mantener la longitud de la ruta al mínimo también reduce la cantidad de certificados que se envían desde el servidor al cliente al validar un certificado de entidad final.

En teoría, una CA raíz, que no tiene ningún pathLenConstraintparámetro, puede autorizar niveles ilimitados de subordinados. CAs Una CA subordinada puede tener tantos subordinados secundarios CAs como permita su configuración interna. Autoridad de certificación privada de AWS Las jerarquías gestionadas admiten rutas de certificación de CA con una profundidad de hasta cinco niveles.

Las estructuras de entidad de certificación bien diseñadas tienen varios beneficios:

- Controles administrativos independientes para distintas dependencias orgánicas
- La capacidad de delegar el acceso a un subordinado CAs
- Una estructura jerárquica que protege los niveles superiores CAs con controles de seguridad adicionales

Dos estructuras de entidades de certificación comunes logran todo esto:

- Dos niveles de entidades de certificación: entidad de certificación raíz y entidad de certificación subordinada

  Esta es la estructura de entidad de certificación más simple que permite políticas de administración, control y seguridad separadas para la entidad de certificación raíz y una entidad de certificación subordinada. Puede mantener controles y políticas restrictivos para la entidad de certificación raíz mientras permite un acceso más permisivo para la entidad de certificación subordinada. Este último se utiliza para la emisión masiva de certificados de entidad final.

- Tres niveles de entidades de certificación: entidad de certificación raíz y dos capas de entidad de certificación subordinada

  De forma similar a la anterior, esta estructura agrega una capa de entidad de certificación adicional para separar aún más la entidad de certificación raíz de las operaciones de entidades de certificación de bajo nivel. La capa de CA intermedia solo se usa para firmar a las entidades subordinadas CAs que se encargan de emitir los certificados de la entidad final.

Las estructuras de entidades de certificación menos comunes incluyen las siguientes:

- Cuatro o más niveles de CA

  Aunque menos comunes que las jerarquías de tres niveles, las jerarquías de entidad de certificación con cuatro o más niveles son posibles y pueden ser necesarias para permitir la delegación administrativa.

- Un nivel de entidad de certificación: solo entidad de certificación raíz

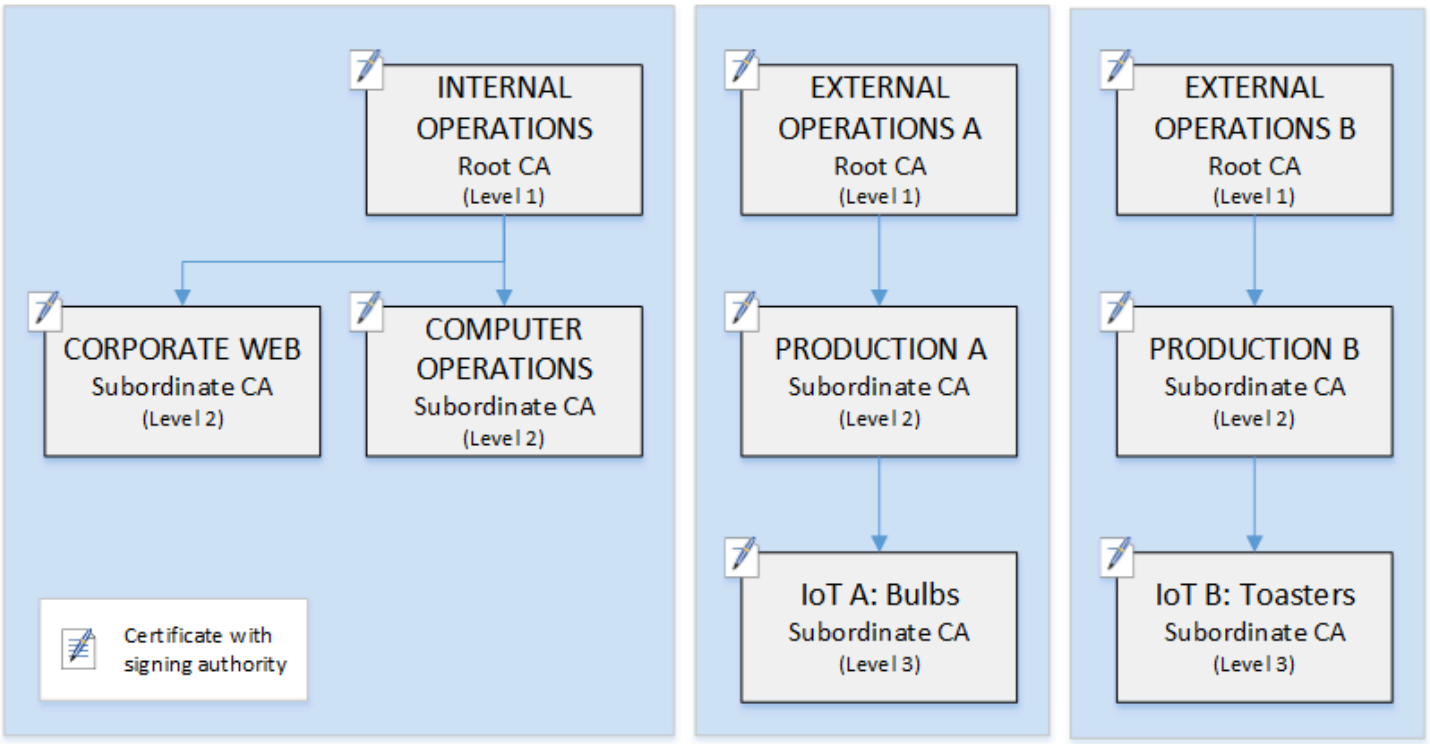
  Esta estructura se utiliza habitualmente para el desarrollo y las pruebas cuando no se requiere una cadena de confianza completa. Su uso en producción es atípico. Además, infringe la práctica recomendada de mantener políticas de seguridad separadas para la CA raíz y la CAs que emite los certificados de la entidad final.

  Sin embargo, si ya está emitiendo certificados directamente desde una CA raíz, puede migrarlos a Autoridad de certificación privada de AWS. Esto proporciona ventajas de seguridad y control sobre el uso de una entidad de certificación raíz administrada con OpenSSL u otro software.

Ejemplo de una PKI privada para un fabricante

En este ejemplo, una hipotética compañía de tecnología fabrica dos productos de Internet de las cosas (IoT), una bombilla inteligente y una tostadora inteligente. Durante la producción, cada dispositivo obtiene un certificado de entidad final para que pueda comunicarse de forma segura a través de Internet con el fabricante. La PKI de la compañía también garantiza su infraestructura informática, incluyendo el sitio web interno y varios servicios informáticos autoalojados que ejecutan operaciones financieras y comerciales.

En consecuencia, la jerarquía de entidad de certificación modela de cerca estos aspectos administrativos y operativos del negocio.



Esta jerarquía contiene tres raíces, una para operaciones internas y dos para operaciones externas (una entidad de certificación raíz para cada línea de productos). También ilustra múltiples longitudes de ruta de certificación, con dos niveles de entidades de certificación para operaciones internas y tres niveles para operaciones externas.

El uso de capas de CA raíz separadas CAs y capas de CA subordinadas adicionales por parte de las operaciones externas es una decisión de diseño que responde a las necesidades empresariales y de seguridad. Con múltiples árboles de entidades de certificación, la PKI está preparada para el futuro contra reorganizaciones corporativas, desinversiones o adquisiciones. Cuando se producen cambios, toda una jerarquía de entidad de certificación raíz puede moverse limpiamente con la división que

protege. Además, al contar con dos niveles de CA subordinados, las raíces CAs tienen un alto nivel de aislamiento respecto del nivel 3, CAs que se encarga de firmar en masa los certificados de miles o millones de artículos fabricados.

En el lado interno, las operaciones de la web corporativa y de la computadora interna completan una jerarquía de dos niveles. Estos niveles permiten a los administradores web y a los ingenieros de operaciones administrar la emisión de certificados de forma independiente para sus propios dominios de trabajo. La compartimentación de PKI en dominios funcionales diferenciados es una práctica recomendada de seguridad y protege a cada uno de un peligro que pudiera afectar al otro. Los administradores web emiten certificados de entidad final para que los utilicen los navegadores web de toda la empresa, autenticando y cifrando las comunicaciones en el sitio web interno. Los ingenieros de operaciones emiten certificados de entidad final que autentican los alojamientos del centro de datos y los servicios informáticos entre sí. Este sistema contribuye a proteger la información confidencial cifrándola en la LAN.

Establezca restricciones de longitud en la ruta de certificación

La estructura de una jerarquía de entidad de certificación se define y aplica mediante la extensión de restricciones básicas que contiene cada certificado. La extensión define dos restricciones:

- `cA`: si el certificado define una CA. Si este valor es falso (el valor predeterminado), el certificado es un certificado de entidad final.
- `pathLenConstraint`— El número máximo de subordinados de nivel inferior CAs que pueden existir en una cadena de confianza válida. El certificado de la entidad final no se cuenta porque no es un certificado de CA.

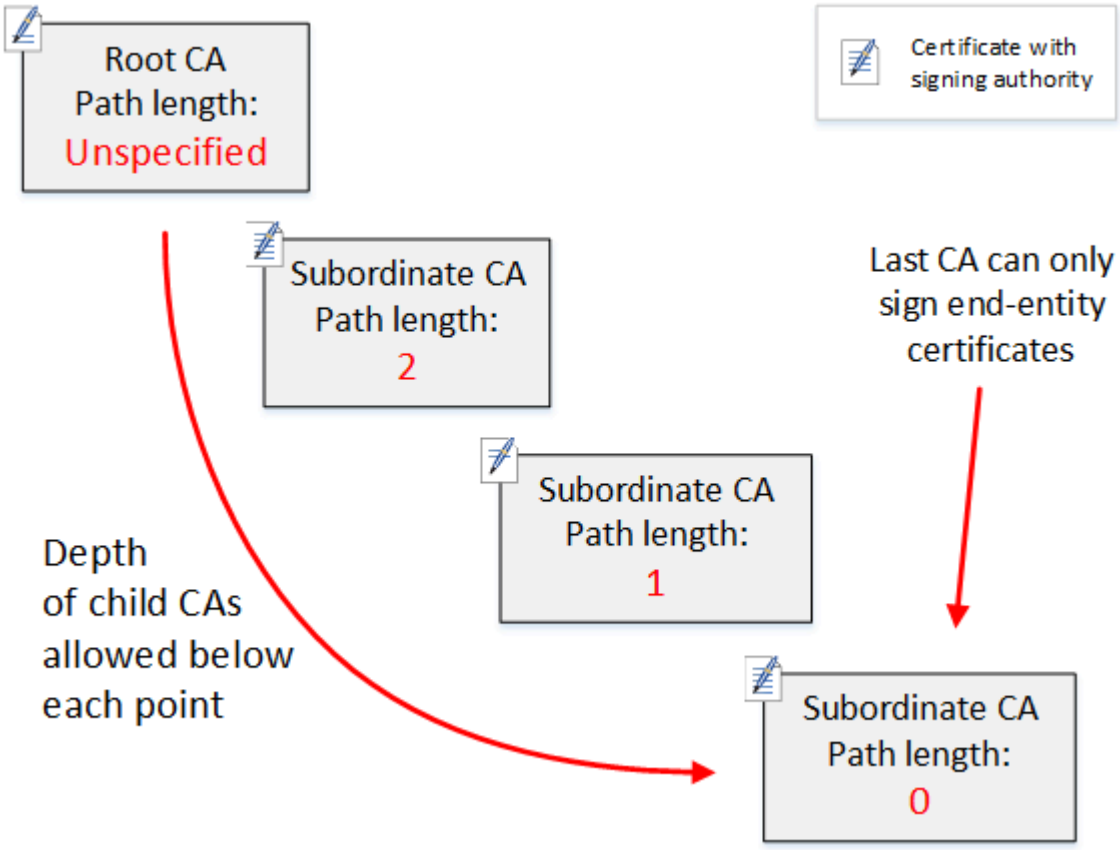
Un certificado de entidad de certificación raíz necesita la máxima flexibilidad y no incluye una restricción de longitud de ruta. Esto permite que la raíz defina una ruta de certificación de cualquier profundidad.

> **Note**
>
> Autoridad de certificación privada de AWS limita la ruta de certificación a cinco niveles.

Los subordinados CAs tienen `pathLenConstraint` valores iguales o superiores a cero, según su ubicación en la jerarquía y las características deseadas. Por ejemplo, en una jerarquía con tres

CAs, no se especifica ninguna restricción de ruta para la CA raíz. La primera CA subordinada tiene una longitud de ruta de 1 y, por lo tanto, puede denominarse secundaria. CAs Cada uno de estos elementos secundarios CAs debe tener necesariamente un `pathLenConstraint` valor de cero. Esto significa que pueden firmar certificados de entidad final pero no pueden emitir certificados de entidad de certificación adicionales. Limitar el poder de crear nuevos CAs es un importante control de seguridad.

El siguiente diagrama ilustra esta propagación de entidad de certificación limitada en la jerarquía.



En esta jerarquía de cuatro niveles, la raíz no está restringida (como siempre). Sin embargo, la primera CA subordinada tiene un `pathLenConstraint` valor de 2, lo que impide que su hijo CAs profundice más de dos niveles. Por consiguiente, para una ruta de certificación válida, el valor de restricción debe disminuir a cero en los dos niveles siguientes. Si un explorador web encuentra que un certificado de entidad final de esta rama tiene una longitud de ruta mayor de cuatro, la validación devuelve un error. Dicho certificado podría ser el resultado de una entidad de certificación creada accidentalmente, de una entidad de certificación configurada incorrectamente o de una emisión no autorizada.

Administre la longitud de la ruta con plantillas

Autoridad de certificación privada de AWS proporciona plantillas para emitir certificados raíz, subordinados y de entidad final. Estas plantillas encapsulan las prácticas recomendadas para los valores básicos de restricciones, incluida la longitud de la ruta. Las plantillas incluyen lo siguiente:

- Raíz /V1 CACertificate
- Subordinado _ 0/V1 CACertificate PathLen
- Subordinado _ 1/V1 CACertificate PathLen
- Subordinado _ 2/V1 CACertificate PathLen
- Subordinado _ 3/V1 CACertificate PathLen
- EndEntityCertificate/V1

La API `IssueCertificate` devolverá un error si intenta crear una entidad de certificación con una longitud de ruta mayor o igual a la longitud de ruta de su certificado de entidad de certificación emisor.

Para obtener más información acerca de las plantillas de certificados, consulte Utilice plantillas de certificados AWS Private CA.

Automatice la configuración de la jerarquía de CA con AWS CloudFormation

Cuando se haya decidido por un diseño para su jerarquía de CA, puede probarlo y ponerlo en producción mediante una AWS CloudFormation plantilla. Para obtener un ejemplo de una plantilla de este tipo, consulte Declaración de una jerarquía de entidad de certificación privada en la Guía del usuario de CloudFormation .

Administre el ciclo de vida de la CA privada

Los certificados de entidad de certificación tienen una duración determinada o un período de validez. Cuando un certificado de CA caduca, todos los certificados emitidos directa o indirectamente por un subordinado CAs inferior a él en la jerarquía de CA dejan de ser válidos. Puede evitar la caducidad del certificado de entidad de certificación planificando con antelación.

Elija los períodos de validez

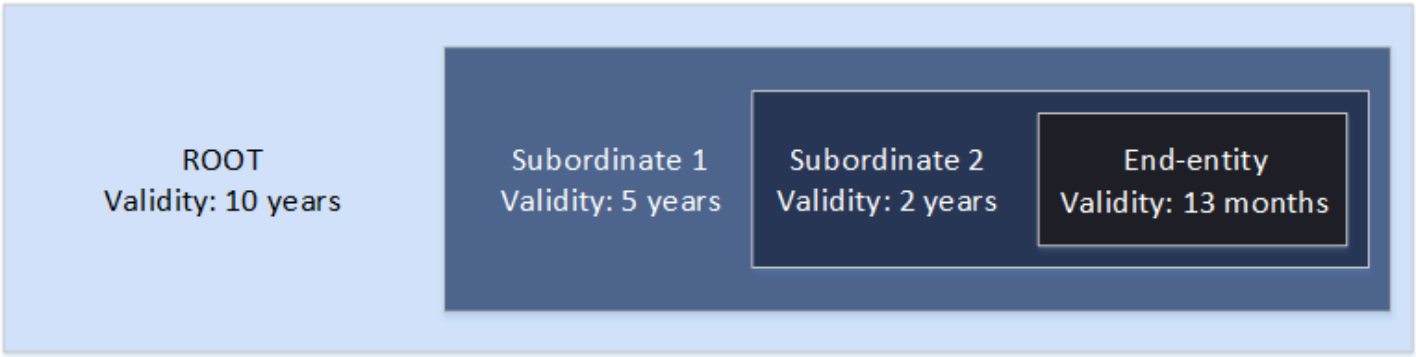
El período de validez de un certificado X.509 es un campo de certificado básico obligatorio. Determina el intervalo de tiempo durante el cual la entidad de certificación emisora certifica que se

puede confiar en el certificado, salvo revocación. (Un certificado raíz, al ser autofirmado, certifica su propio período de validez).

Autoridad de certificación privada de AWS y AWS Certificate Manager ayude a configurar los períodos de validez de los certificados con sujeción a las siguientes restricciones:

- Un certificado gestionado por Autoridad de certificación privada de AWS debe tener un período de validez inferior o igual al período de validez de la CA que lo emitió. En otras palabras, los certificados secundarios CAs y de la entidad final no pueden sobrevivir a los certificados principales. Se produce un error al intentar usar la API `IssueCertificate` para emitir un certificado de entidad de certificación con un período de validez mayor o igual que la entidad de certificación principal.
- Los certificados emitidos y gestionados por AWS Certificate Manager (aquellos para los que ACM genera la clave privada) tienen un período de validez de 13 meses (395 días). ACM gestiona el proceso de renovación de estos certificados. Si Autoridad de certificación privada de AWS solía emitir certificados directamente, puede elegir cualquier período de validez.

El diagrama siguiente muestra una configuración típica de períodos de validez anidados. El certificado raíz es el más duradero; los certificados de la entidad final son relativamente efímeros; y los subordinados CAs oscilan entre estos extremos.



Cuando planifique la jerarquía de la entidad de certificación, determine la duración óptima de los certificados de entidad de certificación. Trabaje retrospectivamente a partir de la duración deseada de los certificados de entidad final que desea emitir.

Certificados de entidad final

Los certificados de entidad final deben tener un período de validez adecuado para el caso de uso. Una duración breve minimiza la exposición de un certificado en caso de pérdida o robo de su clave

privada. Sin embargo, los períodos de vida breves suponen renovaciones frecuentes. Si no se renueva un certificado que caduca, puede producirse un tiempo de inactividad.

El uso distribuido de los certificados de la entidad final también puede plantear problemas logísticos si se produce una infracción de seguridad. Su planificación debe tener en cuenta los certificados de renovación y distribución, la revocación de certificados en riesgo y la rapidez con la que las revocaciones se propagan a los clientes que dependen de los certificados.

El período de validez predeterminado para un certificado de entidad final emitido a través de ACM es de 13 meses (395 días). En Autoridad de certificación privada de AWS, puede utilizar la `IssueCertificate` API para aplicar cualquier período de validez siempre que sea inferior al de la entidad emisora de certificados.

Certificados de entidad de certificación subordinados

Los certificados de entidades de certificación subordinados deben tener períodos de validez mucho más largos que los certificados que emiten. Un buen rango para la validez de un certificado de entidad de certificación es de dos a cinco veces el período de cualquier certificado de entidad de certificación secundario o certificado de entidad final que emite. Por ejemplo, suponga que tiene una jerarquía de entidades de certificación de dos niveles (entidad de certificación raíz y una entidad de certificación subordinada). Si desea emitir certificados de entidad final con una duración de un año, puede configurar la duración de la entidad de certificación emisora subordinada para que sea de tres años. Este es el período de validez predeterminado para un certificado de CA subordinado en Autoridad de certificación privada de AWS. Los certificados de entidad de certificación subordinados se pueden cambiar sin reemplazar el certificado de entidad de certificación raíz.

Certificados raíz

Los cambios realizados en un certificado de entidad de certificación raíz afectan a toda la PKI (infraestructura de clave pública) y requieren que actualice todos los almacenes de confianza del navegador y del sistema operativo cliente dependientes. Para minimizar el impacto operativo, debe elegir un período de validez largo para el certificado raíz. El Autoridad de certificación privada de AWS valor predeterminado para los certificados raíz es de diez años.

Administre la sucesión de CA

Tiene dos formas de administrar la sucesión de entidades de certificación: reemplazar la entidad de certificación antigua o volver a emitir la entidad de certificación con un nuevo período de validez.

Sustituya una CA antigua

Para reemplazar una entidad de certificación antigua, debe crear una entidad de certificación nueva y encadenarla a la misma entidad de certificación principal. Después, emitirá certificados de la nueva entidad de certificación.

Los certificados emitidos por la nueva entidad de certificación tienen una nueva cadena de entidad de certificación. Una vez establecida la nueva entidad de certificación, puede desactivar la antigua entidad de certificación para evitar que emita nuevos certificados. Mientras esté deshabilitada, la antigua CA admite la revocación de los certificados antiguos emitidos por la CA y, si está configurada para ello, sigue validando los certificados mediante listas de revocación de and/or certificados OCSP (). CRLs Cuando caduque el último certificado emitido desde la entidad de certificación antigua, puede eliminar la entidad de certificación antigua. Puede generar un informe de auditoría para todos los certificados emitidos por la entidad de certificación para confirmar que todos los certificados emitidos han caducado. Si la CA anterior tiene una CA subordinada CAs, también debe sustituirla, ya que la CA subordinada CAs caduca al mismo tiempo o antes que la CA principal. Comience reemplazando la entidad de certificación más alta de la jerarquía que debe reemplazarse. A continuación, cree una nueva subordinada de reemplazo CAs en cada nivel inferior subsiguiente.

AWS recomienda incluir un identificador de generación de CA en los nombres, CAs según sea necesario. Por ejemplo, suponga que nombra a la entidad de certificación de primera generación “entidad de certificación raíz corporativa”. Cuando cree la entidad emisora de certificados de segunda generación, asígnele el nombre “entidad de certificación raíz corporativa G2”. Esta sencilla convención de nomenclatura puede ayudar a evitar confusiones cuando ambas no CAs hayan caducado.

Se prefiere este método de sucesión de entidades de certificación porque gira la clave privada de la entidad de certificación. La rotación de la clave privada es una práctica recomendada para las claves de entidades de certificación. La frecuencia de rotación debe ser proporcional a la frecuencia de uso de las claves: a medida CAs que se expidan más certificados, se deben rotar con mayor frecuencia.

> **Note**
>
> Los certificados privados emitidos a través de ACM no se pueden renovar si reemplaza la entidad de certificación. Si utiliza ACM para la emisión y renovación, debe volver a emitir el certificado de CA para extender la duración de la CA.

Vuelva a emitir una CA antigua

Cuando una CA se acerca a su fecha de caducidad, un método alternativo para prolongar su vida útil consiste en volver a emitir el certificado de CA con una nueva fecha de caducidad. La reemisión deja todos los metadatos de la CA en su lugar y conserva las claves públicas y privadas existentes. En este escenario, la cadena de certificados existente y los certificados de entidad final sin caducar emitidos por la CA siguen siendo válidos hasta que caduquen. La emisión de nuevos certificados también puede continuar sin interrupciones. Para actualizar una CA con un certificado reemitido, siga los procedimientos de instalación habituales que se describen en Instalación del certificado de CA.

> **Note**
>
> Recomendamos sustituir las CA que vayan a caducar en lugar de volver a emitir su certificado dadas las ventajas de seguridad que se obtienen al cambiar a un nuevo par de claves.

Revocar una CA

Para revocar una CA, se revoca su certificado subyacente. Esto también revoca de manera efectiva todos los certificados emitidos por la CA. La información de revocación se distribuye a los clientes mediante el OCSP o una CRL. Debe revocar un certificado de CA solo si desea revocar efectivamente todos los certificados de entidad final y CA subordinada.

Planifique el método de revocación AWS Private CA de su certificado

Al planificar su PKI privada Autoridad de certificación privada de AWS, debe tener en cuenta cómo gestionar las situaciones en las que desee que los puntos finales dejen de confiar en un certificado emitido, por ejemplo, cuando se expone la clave privada de un punto final. Los enfoques habituales para solucionar este problema son utilizar certificados de corta duración o configurar la revocación de certificados. Los certificados de corta duración caducan en un período de tiempo tan breve, en horas o días, que la revocación no tiene sentido, ya que el certificado deja de ser válido aproximadamente al mismo tiempo que tarda en notificarse la revocación a un punto de conexión. En esta sección se describen las opciones de revocación para los clientes de Autoridad de certificación privada de AWS , incluida la configuración y las prácticas recomendadas.

Los clientes que buscan un método de revocación pueden elegir el Protocolo de estado de certificados en línea (OCSP), las listas de revocación de certificados (CRLs) o ambos.

> **Note**
>
> Si crea su CA sin configurar la revocación, siempre podrá configurarla más adelante. Para obtener más información, consulte Actualice una CA privada en AWS Private Certificate Authority.

- Protocolo de estado de certificados en línea (OCSP)

  Autoridad de certificación privada de AWS proporciona una solución OCSP totalmente gestionada para notificar a los puntos finales que los certificados se han revocado sin necesidad de que los clientes operen la infraestructura ellos mismos. Los clientes pueden habilitar OCSP en sistemas nuevos o existentes CAs con una sola operación mediante la Autoridad de certificación privada de AWS consola, la API, la CLI o mediante CloudFormation. Mientras que se CRLs almacenan y procesan en el terminal y pueden quedar obsoletos, los requisitos de almacenamiento y procesamiento del OCSP se gestionan de forma sincrónica en el backend del respondedor.

  Al habilitar el OCSP para una CA, Autoridad de certificación privada de AWS incluye la URL del respondedor del OCSP en la extensión Authority Information Access (AIA) de cada nuevo certificado emitido. Los clientes, como los navegadores web, consultan las CRL para determinar si se puede confiar en un certificado de entidad final o CA subordinada. El sistema de respuesta devuelve un mensaje de estado firmado criptográficamente para garantizar su autenticidad.

  El respondedor Autoridad de certificación privada de AWS OCSP cumple con la RFC 5019.

  Consideraciones sobre OCSP

  - Los mensajes de estado del OCSP se firman mediante el mismo algoritmo de firma para el que se configuró la CA emisora. CAs creados en la Autoridad de certificación privada de AWS consola, utilizan el algoritmo de firma SHA256 WITHRSA de forma predeterminada. Puedes encontrar otros algoritmos compatibles en la documentación de la CertificateAuthorityConfigurationAPI.
  - APIPassthrough y las plantillas de CSRPassthrough certificados no funcionarán con la extensión AIA si el respondedor OCSP está activado.

- Se puede acceder al punto de conexión del servicio OCSP administrado en la Internet pública. Los clientes que deseen utilizar el OCSP, pero prefieran no tener un punto de conexión público deberán utilizar su propia infraestructura de OCSP.

- Listas de revocación de certificados () CRLs

  Una lista de revocación de certificados (CRL) es un archivo que contiene una lista de certificados revocados antes de su fecha de caducidad programada. La CRL contiene una lista de certificados en los que ya no se debe confiar, el motivo de la revocación y otra información relevante.

  Al configurar la autoridad de certificación (CA), puede elegir si Autoridad de certificación privada de AWS crea una CRL completa o particionada. Su elección determina el número máximo de certificados que la autoridad de certificación puede emitir y revocar. Para obtener más información, consulte Cuotas de Autoridad de certificación privada de AWS.

  Consideraciones sobre la CRL

  - Consideraciones sobre la memoria y el ancho de banda: CRLs requieren más memoria que el OCSP debido a los requisitos locales de descarga y procesamiento. Sin embargo, CRLs podría reducir el ancho de banda de la red en comparación con el OCSP al almacenar en caché las listas de revocación en lugar de comprobar el estado de cada conexión. En el caso de los dispositivos con limitaciones de memoria, como algunos dispositivos de IoT, considere la posibilidad de utilizar dispositivos particionados. CRLs
  - Cambiar el tipo de CRL: al cambiar de una CRL completa a una particionada, Autoridad de certificación privada de AWS crea nuevas particiones según sea necesario y añade la extensión IDP a todas, incluida la original. CRLs Al cambiar de particionada a completa, solo se actualiza una única CRL y se evita la revocación futura de los certificados asociados a las particiones anteriores.

> **Note**
>
> Tanto el OCSP como el cambio de CRLs estado presentan cierto retraso entre la revocación y la disponibilidad del cambio de estado.
>
> - Cuando se revoca un certificado, las respuestas del OCSP pueden tardar hasta 60 minutos en reflejar el nuevo estado. En general, el OCSP suele permitir una distribución más rápida de la información de revocación porque, a diferencia de CRLs lo que los clientes pueden almacenar en caché durante días, los clientes no suelen almacenar en caché las respuestas del OCSP.

- Las CRL se actualizan aproximadamente 30 minutos después de que un certificado se revoque. Si por alguna razón se produce un error en la actualización de la CRL, Autoridad de certificación privada de AWS lo intenta de nuevo cada 15 minutos.

Requisitos generales para las configuraciones de revocación

Los siguientes requisitos se aplican a todas las configuraciones de revocación.

- Una configuración que inhabilite CRLs o un OCSP solo debe contener el `Enabled=False` parámetro y fallará si se incluyen otros parámetros, como `CustomCname` o `ExpirationInDays` están incluidos.
- En la configuración de una CRL, el parámetro `S3BucketName` debe cumplir las reglas de nomenclatura de los bucket de Amazon Simple Storage Service.
- Una configuración que contenga un parámetro de nombre canónico (CNAME) personalizado para CRLs un OCSP debe cumplir con RFC7230las restricciones sobre el uso de caracteres especiales en un CNAME.
- En la configuración de una CRL o un OCSP, el valor de un parámetro de CNAME no debe incluir un prefijo de protocolo como “http://” o “https://”.

Temas

- Configura una CRL para AWS Private CA
- Personaliza la URL de OCSP para AWS Private CA

Configura una CRL para AWS Private CA

Antes de poder configurar una lista de revocación de certificados (CRL) como parte del proceso de creación de una entidad de certificación, es posible que sea necesaria alguna configuración previa. En esta sección se explican los requisitos previos y las opciones que debe conocer antes de crear una CA con una CRL asociada.

Para obtener información sobre el uso del Protocolo de estado de certificados en línea (OCSP) como alternativa o complemento de una CRL, consulte Certificate revocation options y Personaliza la URL de OCSP para AWS Private CA.

Temas

- Tipos de CRL
- Estructura de CRL
- Políticas de acceso para CRLs Amazon S3
- Habilite S3 Block Public Access (BPA) con CloudFront
- Determinación del URI del punto de distribución (CDP) de la CRL

Tipos de CRL

- Completo: configuración predeterminada. Autoridad de certificación privada de AWS mantiene un único archivo CRL sin particiones para todos los certificados no vencidos emitidos por una entidad emisora de certificados que hayan sido revocados. Cada certificado que se Autoridad de certificación privada de AWS emite está vinculado a una CRL específica a través de su extensión de punto de distribución de la CRL (CDP), tal como se define en el RFC 5280. Puede tener hasta 1 millón de certificados privados para cada CA con la CRL completa habilitada. Para obtener más información, consulte las AWS Private CA cuotas.
- Particionado: si lo comparamos con uno completo CRLs, el número de certificados particionados aumenta CRLs considerablemente el número de certificados que su entidad emisora de certificados privada puede emitir y evita tener que rotarlos con frecuencia. CAs

> **Important**
>
> Si utiliza particionado CRLs, debe comprobar que el URI del punto de distribución emisor (IDP) asociado a la CRL coincide con el URI del CDP del certificado para asegurarse de que se ha obtenido la CRL correcta. Autoridad de certificación privada de AWS marca la extensión IDP como crítica y su cliente debe poder procesarla.

Estructura de CRL

Cada CRL es un archivo codificado con DER. Para descargar el archivo y usar OpenSSL para verlo, use un comando como el siguiente:

```
openssl crl -inform DER -in path-to-crl-file -text -noout
```

CRLs tienen el siguiente formato:

```
Certificate Revocation List (CRL):
          Version 2 (0x1)
      Signature Algorithm: sha256WithRSAEncryption
          Issuer: /C=US/ST=WA/L=Seattle/O=Example Company CA/OU=Corporate/
CN=www.example.com
          Last Update: Feb 26 19:28:25 2018 GMT
          Next Update: Feb 26 20:28:25 2019 GMT
          CRL extensions:
              X509v3 Authority Key Identifier:
                  keyid:AA:6E:C1:8A:EC:2F:8F:21:BC:BE:80:3D:C5:65:93:79:99:E7:71:65

              X509v3 CRL Number:
                  1519676905984
  Revoked Certificates:
      Serial Number: E8CBD2BEDB122329F97706BCFEC990F8
          Revocation Date: Feb 26 20:00:36 2018 GMT
          CRL entry extensions:
              X509v3 CRL Reason Code:
                  Key Compromise
      Serial Number: F7D7A3FD88B82C6776483467BBF0B38C
          Revocation Date: Jan 30 21:21:31 2018 GMT
          CRL entry extensions:
              X509v3 CRL Reason Code:
                  Key Compromise
      Signature Algorithm: sha256WithRSAEncryption
           82:9a:40:76:86:a5:f5:4e:1e:43:e2:ea:83:ac:89:07:49:bf:
           c2:fd:45:7d:15:d0:76:fe:64:ce:7b:3d:bb:4c:a0:6c:4b:4f:
           9e:1d:27:f8:69:5e:d1:93:5b:95:da:78:50:6d:a8:59:bb:6f:
           49:9b:04:fa:38:f2:fc:4c:0d:97:ac:02:51:26:7d:3e:fe:a6:
           c6:83:34:b4:84:0b:5d:b1:c4:25:2f:66:0a:2e:30:f6:52:88:
           e8:d2:05:78:84:09:01:e8:9d:c2:9e:b5:83:bd:8a:3a:e4:94:
           62:ed:92:e0:be:ea:d2:59:5b:c7:c3:61:35:dc:a9:98:9d:80:
           1c:2a:f7:23:9b:fe:ad:6f:16:7e:22:09:9a:79:8f:44:69:89:
           2a:78:ae:92:a4:32:46:8d:76:ee:68:25:63:5c:bd:41:a5:5a:
           57:18:d7:71:35:85:5c:cd:20:28:c6:d5:59:88:47:c9:36:44:
           53:55:28:4d:6b:f8:6a:00:eb:b4:62:de:15:56:c8:9c:45:d7:
           83:83:07:21:84:b4:eb:0b:23:f2:61:dd:95:03:02:df:0d:0f:
           97:32:e0:9d:38:de:7c:15:e4:36:66:7a:18:da:ce:a3:34:94:
           58:a6:5d:5c:04:90:35:f1:8b:55:a9:3c:dd:72:a2:d7:5f:73:
           5a:2c:88:85
```

> **Note**
>
> La CRL solo se depositará en Amazon S3 una vez que se haya emitido un certificado que haga referencia a ella. Antes de eso, solo habrá un archivo `acm-pca-permission-test-key` visible en el bucket de Amazon S3.

Políticas de acceso para CRLs Amazon S3

Si planea crear una CRL, debe preparar un bucket de Amazon S3 para almacenarla. Autoridad de certificación privada de AWS deposita automáticamente la CRL en el bucket de Amazon S3 que designe y la actualiza periódicamente. Para obtener más información, consulte Creación de un bucket.

Su bucket de S3 debe estar protegido por una política de permisos de IAM asociada. Los usuarios autorizados y los directores de servicio necesitan `Put` permiso Autoridad de certificación privada de AWS para poder colocar objetos en el depósito y `Get` para recuperarlos.

> **Note**
>
> La configuración de la política de IAM depende de la Regiones de AWS persona implicada. Las regiones se dividen en dos categorías:
>
> - Regiones habilitadas de forma predeterminada: regiones que están habilitadas de forma predeterminada para todos. Cuentas de AWS
> - Regiones deshabilitadas de forma predeterminada: regiones que están deshabilitadas de forma predeterminada, pero que el cliente puede habilitarlas manualmente.
>
> Para obtener más información y una lista de las regiones deshabilitadas de forma predeterminada, consulte Administración de Regiones de AWS. Para obtener más información sobre los principios de servicio en el contexto de la IAM, consulte los principios de servicio de AWS en las regiones en las que se ha optado por participar.
> Cuando se configura CRLs como método de revocación de certificados, Autoridad de certificación privada de AWS crea una CRL y la publica en un bucket de S3. El bucket de S3 requiere una política de IAM que permita al director del Autoridad de certificación privada de AWS servicio escribir en el bucket. El nombre de la entidad principal del servicio varía según las regiones utilizadas y no se admiten todas las posibilidades.

PCA	S3	Entidad principal de servicio
Ambos en la misma región		`acm-pca.amazonaws.com`
Habilitado	Habilitado	`acm-pca.amazonaws.com`
Deshabilitado	Habilitado	`acm-pca.Region.amazonaws.com`
Habilitado	Deshabilitado	No compatible

La política predeterminada no aplica ninguna restricción de `SourceArn` a la CA. Le recomendamos que aplique una política menos permisiva, como la siguiente, que restrinja el acceso tanto a una AWS cuenta específica como a una CA privada específica. Como alternativa, puede usar la clave de condición aws: SourceOrg ID para restringir el acceso a una organización específica. AWS Organizations Para obtener más información sobre las políticas de depósitos, consulte Políticas de depósitos de Amazon Simple Storage Service.

Si decide permitir la política predeterminada, siempre podrá modificarla más adelante.

Habilite S3 Block Public Access (BPA) con CloudFront

Los nuevos buckets de Amazon S3 se configuran de forma predeterminada con la característica Bloquear acceso público (BPA) activada. Incluido en las mejores prácticas de seguridad de Amazon S3, el BPA es un conjunto de controles de acceso que los clientes pueden usar para ajustar el acceso a los objetos de sus buckets de S3 y a los buckets en su conjunto. Cuando el BPA está activo y correctamente configurado, solo AWS los usuarios autorizados y autenticados tienen acceso a un depósito y a su contenido.

AWS recomienda el uso de BPA en todos los depósitos de S3 para evitar que la información confidencial quede expuesta a posibles adversarios. Sin embargo, si sus clientes de PKI obtienen información a CRLs través de la Internet pública (es decir, sin haber iniciado sesión en una cuenta), es necesario planificar más a fondo la información. AWS En esta sección se describe cómo configurar una solución de PKI privada mediante Amazon CloudFront, una red de entrega de contenido (CDN), para que sirva CRLs sin requerir el acceso de un cliente autenticado a un bucket de S3.

> **Note**
>
> Su uso CloudFront conlleva costes adicionales en su cuenta. AWS Para obtener más información, consulta los CloudFront precios de Amazon.
> Si decide almacenar su CRL en un bucket de S3 con el BPA activado y no lo utiliza CloudFront, debe crear otra solución de CDN para garantizar que su cliente de PKI tenga acceso a su CRL.

Configúrelo para el BPA CloudFront

Cree una CloudFront distribución que tenga acceso a su bucket privado de S3 y que pueda servir CRLs a clientes no autenticados.

Para configurar una CloudFront distribución para la CRL

1. Cree una CloudFront distribución nueva mediante el procedimiento descrito en Creación de una distribución en la Guía para CloudFront desarrolladores de Amazon.

   Mientras completa el procedimiento, aplique la siguiente configuración:

   - En Nombre de dominio de origen, elija su bucket de S3.
   - En Restringir acceso al bucket, elija Sí.
   - Elija Create a New Identity (Crear una nueva identidad) para Origin Access Identity (Identidad de acceso de origen).
   - Selecciona Yes, Update Bucket Policy (Sí, actualizar la política del bucket) en Grant Read Permissions on Bucket (Otorgar permisos de lectura en el bucket).

   > **Note**
   >
   > En este procedimiento, CloudFront modifica la política de su bucket para permitirle acceder a los objetos del bucket. Considere la posibilidad de editar esta política para permitir el acceso únicamente a los objetos de la carpeta `crl`.

2. Una vez inicializada la distribución, busque su nombre de dominio en la CloudFront consola y guárdelo para el siguiente procedimiento.

> **Note**
>
> Si tu bucket de S3 se creó recientemente en una región distinta de us-east-1, es posible que recibas un error de redireccionamiento temporal HTTP 307 al acceder a la aplicación publicada a través de ella. CloudFront Es posible que la dirección del bucket tarde varias horas en propagarse.

Configuración de CA para el BPA

Al configurar su nueva CA, incluya el alias en su CloudFront distribución.

Para configurar su CA con un CNAME para CloudFront

- Cree su CA mediante Cree una CA privada en AWS Private CA.

  Al realizar el procedimiento, el archivo de revocación `revoke_config.txt` debe incluir las siguientes líneas para especificar un objeto de CRL no público y proporcionar una URL al punto de conexión de distribución en el que: CloudFront

  ```
  "S3ObjectAcl":"BUCKET_OWNER_FULL_CONTROL",
   "CustomCname":"abcdef012345.cloudfront.net"
  ```

  Posteriormente, cuando emita certificados con esta CA, estos contendrán un bloque como el siguiente:

  ```
  X509v3 CRL Distribution Points:
   Full Name:
   URI:http://abcdef012345.cloudfront.net/crl/01234567-89ab-
  cdef-0123-456789abcdef.crl
  ```

> **Note**
>
> Si tiene certificados antiguos emitidos por esta CA, no podrán acceder a la CRL.

Determinación del URI del punto de distribución (CDP) de la CRL

Si necesita utilizar el URI del punto de distribución de CRL (CDP) en su flujo de trabajo, puede emitir un certificado utilizando el URI de CRL de ese certificado o utilizar el siguiente método. Esto solo funciona si está completo. CRLs CRLs Las particiones tienen un GUID aleatorio adjunto.

Si usa el bucket S3 como punto de distribución de CRL (CDP) para su CA, el URI del CDP puede tener uno de los siguientes formatos.

- `http://amzn-s3-demo-bucket.s3.region-code.amazonaws.com/crl/CA-ID.crl`
- `http://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/crl/CA-ID.crl`

Si ha configurado la CA con un CNAME personalizado, el URI del CDP incluirá el CNAME, por ejemplo, `http://alternative.example.com/crl/CA-ID.crl`

De forma predeterminada, Autoridad de certificación privada de AWS escribe las extensiones de CDP utilizando puntos finales solo regionales. IPv4 `amazonaws.com` Para usar CRLs over IPv6, lleve a cabo uno de los siguientes pasos para que CDPs se escriba con URLs ese punto en los puntos finales de doble pila de S3:

- Defina el nombre personalizado de su CRL para el dominio del punto final de doble pila de S3. Por ejemplo, `bucketname.s3.dualstack.region-code.amazonaws.com`
- Configura tu propio registro DNS CNAME que apunte al punto final de doble pila de S3 correspondiente y úsalo como nombre personalizado de tu CRL

Personaliza la URL de OCSP para AWS Private CA

> **Note**
>
> Este tema está dirigido a los clientes que desean personalizar la URL pública del punto final del respondedor del Protocolo de estado de certificados en línea (OCSP) con fines de marca u otros fines. Si planea usar la configuración predeterminada del OCSP Autoridad de certificación privada de AWS administrado, puede omitir este tema y seguir las instrucciones de configuración que aparecen en Configurar la revocación.

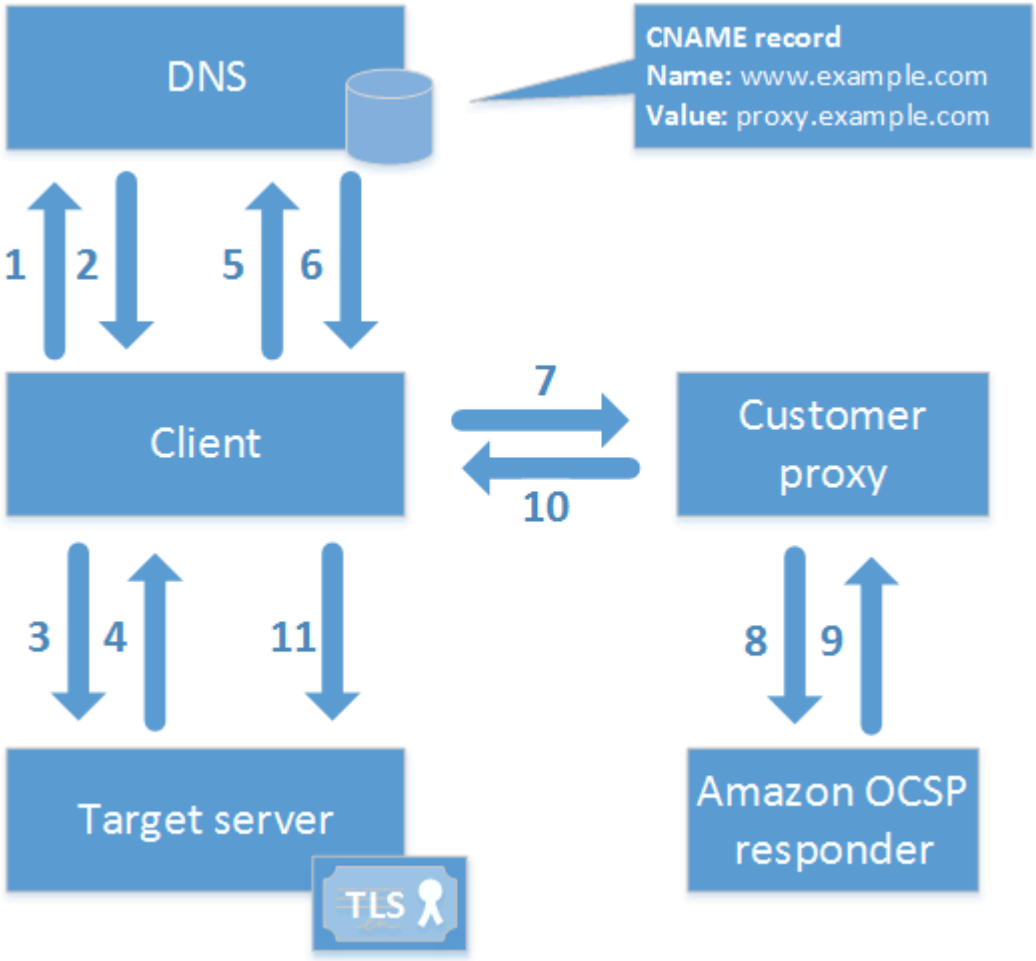
De forma predeterminada, al habilitar el OCSP para Autoridad de certificación privada de AWS, cada certificado que emita contiene la URL del respondedor de OCSP. AWS Esto permite a los clientes que soliciten una conexión criptográficamente segura enviar consultas de validación de OCSP directamente a AWS. Sin embargo, en algunos casos, puede ser preferible indicar una URL diferente en los certificados y, en última instancia, enviar las consultas de OCSP a AWS.

> **Note**
>
> Para obtener información sobre el uso de una lista de revocación de certificados (CRL) como alternativa o complemento del OCSP, consulte Configurar la revocación y Planear una lista de revocación de certificados (CRL).

La configuración de una URL personalizada para OCSP implica tres elementos.

- Configuración de CA: especifique una URL de OCSP personalizada en `RevocationConfiguration` para su CA, tal y como se describe en Ejemplo 2: crear una CA con OCSP activado y un CNAME personalizado y habilitado y Cree una CA privada en AWS Private CA.
- DNS: añada un registro CNAME a la configuración de su dominio para asignar la URL que aparece en los certificados a la URL de un servidor proxy. Para obtener más información, consulta Ejemplo 2: crear una CA con OCSP activado y un CNAME personalizado y habilitado en Cree una CA privada en AWS Private CA.
- Servidor proxy de reenvío: configure un servidor proxy que pueda reenviar de forma transparente el tráfico OCSP que reciba al sistema de respuesta de OCSP de AWS .

El siguiente diagrama ilustra cómo estos componentes funcionan juntos.



Como se muestra en el diagrama, el proceso de validación de OCSP personalizado consta de los siguientes pasos:

1. El cliente consulta el DNS del dominio de destino.
2. El cliente recibe la IP de destino.
3. El cliente abre una conexión TCP con el destino.
4. El cliente recibe el certificado TLS de destino.
5. El cliente consulta el DNS del dominio OCSP que aparece en el certificado.
6. El cliente recibe la IP del proxy.
7. El cliente envía la consulta OCSP al proxy.
8. El proxy reenvía la consulta al sistema de respuesta de OCSP.
9. El sistema de respuesta devuelve el estado del certificado al proxy.
10. El proxy reenvía el estado del certificado al cliente.

11. Si el certificado es válido, el cliente inicia el protocolo de enlace TLS.

> **Tip**
>
> Este ejemplo se puede implementar con Amazon CloudFront y Amazon Route 53 después de haber configurado una CA como se describe anteriormente.
>
> 1. En CloudFront, cree una distribución y configúrela de la siguiente manera:
>   - Cree un nombre alternativo que coincida con su CNAME personalizado.
>   - Enlace su certificado a él.
>   - `ocsp.acm-pca.<region>.amazonaws.com`Establézcala como origen.
>   - Para usar IPv6 las conexiones, use el punto final de dualstack `acm-pca-ocsp.<region>.api.aws`
>   - Implemente la política de `Managed-CachingDisabled`.
>   - Cambie Política del protocolo del visor a HTTP y HTTPS.
>   - Configure Métodos HTTP permitidos: GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE.
> 2. En Route 53, cree un registro DNS que asigne su CNAME personalizado a la URL de la distribución. CloudFront

Uso de OCSP en lugar de IPv6

La URL predeterminada del respondedor Autoridad de certificación privada de AWS OCSP es IPv4 -only. Para usar OCSP over IPv6, configure una URL de OCSP personalizada para su CA. La URL puede ser:

- El FQDN del respondedor OCSP PCA de doble pila, que adopta la forma `acm-pca-ocsp.region-name.api.aws`
- Un registro CNAME que ha configurado para apuntar al respondedor OCSP de doble pila, como se ha explicado anteriormente.

AWS Private CA Comprenda los modos de CA

Autoridad de certificación privada de AWS admite la creación de una autoridad de certificación (CA) en cualquiera de los dos modos. Los modos, certificado de uso general y certificado de corta duración, afectan al período de validez permitido de los certificados emitidos por la CA.

> **Note**
>
> Autoridad de certificación privada de AWS no comprueba la validez de los certificados de CA raíz.

De uso general (predeterminado)

Este modo permite a la CA emitir certificados con cualquier período de validez. La mayoría de las aplicaciones utilizan certificados de este tipo. Por lo general, la CA también especifica un mecanismo de revocación.

Certificado de corta duración

Este modo define una CA que emite exclusivamente certificados con un período de validez máximo de siete días. Estos certificados de corta duración caducan tan rápido que se pueden implementar sin que exista un mecanismo de revocación. Para algunas aplicaciones, tiene más sentido implementar certificados de corta duración con frecuencia que sobrecargar la red y el procesamiento debido a la revocación.

Los certificados de corta duración deben ser la última CA de la jerarquía de certificados. Los gastos generales son considerables, ya que la CA privada debe renovarse cada siete días.

CAs con un modo de certificado de corta duración, cuestan menos que los de uso general CAs. Para más información, consulte Precios de AWS Private Certificate Authority.

Para crear una CA que emita certificados de corta duración, defina el `UsageMode` parámetro en un certificado de corta duración mediante el procedimiento de creación de una CA para crear una CA.

> **Note**
>
> AWS Certificate Manager no puede emitir certificados firmados por una entidad emisora de certificados privada en el modo de corta duración.

Los siguientes servicios de AWS admiten el uso de certificados de corta duración:

- Amazon AppStream
- Amazon WorkSpaces

Planifique la resiliencia en AWS Private CA

La infraestructura AWS global se basa en AWS regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte Infraestructura global.AWS

Redundancia y recuperación de desastres

Tenga en cuenta la redundancia y la recuperación ante desastres al planificar su jerarquía de CA. Autoridad de certificación privada de AWS está disponible en varias regiones, lo que le permite crear redundantes CAs en varias regiones. El Autoridad de certificación privada de AWS servicio funciona con un acuerdo de nivel de servicio (SLA) con una disponibilidad del 99,9%. Existen al menos dos enfoques que puede tener en cuenta para la redundancia y la recuperación de desastres. Puede configurar la redundancia en la entidad de certificación raíz o en la entidad de certificación subordinada más alta. Cada enfoque tiene pros y contras.

1. Puede crear dos raíces CAs en dos AWS regiones diferentes para la redundancia y la recuperación ante desastres. Con esta configuración, cada CA raíz funciona de forma independiente en una AWS región, lo que le protege en caso de que se produzca un desastre en una sola región. Sin embargo, la creación de una entidad raíz CAs redundante aumenta la complejidad operativa: tendrá que distribuir ambos certificados de CA raíz a los almacenes de confianza de los navegadores y sistemas operativos de su entorno.
2. También puede crear sistemas subordinados redundantes CAs para implementarlos en cada una de sus AWS regiones y encadenarlos a la misma CA raíz única en una sola región. AWS El beneficio de este enfoque es que sólo necesita distribuir un único certificado de entidad de

certificación raíz a los almacenes de confianza de su entorno. La limitación es que no tiene una CA raíz redundante en caso de que se produzca un desastre que afecte a la AWS región en la que se encuentra su CA raíz.

Autoridades de certificación en AWS Private CA

Con AWS Private Certificate Authorityél, puede crear una jerarquía completamente AWS alojada de autoridades de certificación (CAs) raíz y subordinadas para uso interno de su organización. Para gestionar la revocación de certificados, puede habilitar el Protocolo de estado de certificados en línea (OCSP), las listas de revocación de certificados (CRLs) o ambos. Autoridad de certificación privada de AWS almacena y administra sus certificados de CA y las respuestas de OCSP CRLs, y las claves privadas de sus autoridades raíz se almacenan de forma segura en. AWS

> **Note**
>
> La implementación de OCSP en no Autoridad de certificación privada de AWS admite las extensiones de solicitud de OCSP. Si envía una consulta por lotes de OCSP que contiene varios certificados, el respondedor de AWS OCSP procesa solo el primer certificado de la cola y descarta los demás. Una revocación puede tardar hasta una hora en aparecer en las respuestas del OCSP.

Puede acceder Autoridad de certificación privada de AWS mediante la Consola de administración de AWS, la y la AWS CLI API. Autoridad de certificación privada de AWS En los siguientes temas, se muestra cómo utilizar la consola y la CLI. Para obtener más información sobre la API, consulte la Referencia de la API de AWS Private Certificate Authority. Para ver ejemplos de Java en los que se muestra cómo utilizar la API, consulte AWS Private CA Úselo con el AWS SDK para Java.

Tras crear una CA privada activa y configurar el acceso a ella, puede emitir y recuperar certificados, tal y como se describe enEmita y administre certificados en AWS Private CA.

Temas

- Configurado para usar AWS Private CA
- Cree una CA privada en AWS Private CA
- Instalación del certificado de CA
- Controle el acceso a la CA privada
- Lista privada CAs
- Ver una CA privada
- Agregue etiquetas para su CA privada
- Comprenda el estado de la AWS Private CA CA

- Actualice una CA privada en AWS Private Certificate Authority
- Eliminación de una CA privada
- Restaure una CA privada
- Utilice certificados de CA privados firmados externamente

Configurado para usar AWS Private CA

Si aún no es cliente de Amazon Web Services (AWS), regístrese para poder utilizar Autoridad de certificación privada de AWS. Su cuenta tiene acceso automáticamente a todos los servicios disponibles, pero solo se le cobrarán los servicios que utilice.

Temas

- Inscríbase en una Cuenta de AWS
- Creación de un usuario con acceso administrativo
- Instale el AWS Command Line Interface

Inscríbase en una Cuenta de AWS

Si no tiene uno Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirte a una Cuenta de AWS

1. Abrir https://portal.aws.amazon.com/billing/registro.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica o mensaje de texto e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en un Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea un. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar Tareas que requieren acceso de usuario raíz.

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. En cualquier momento, puede ver la actividad de su cuenta actual y administrarla accediendo a https://aws.amazon.com/y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de crear un usuario administrativo Cuenta de AWS, asegúrelo Usuario raíz de la cuenta de AWS AWS IAM Identity Center, habilite y cree un usuario administrativo para no usar el usuario root en las tareas diarias.

Proteja su Usuario raíz de la cuenta de AWS

1. Inicie sesión Consola de administración de AWScomo propietario de la cuenta seleccionando el usuario root e introduciendo su dirección de Cuenta de AWS correo electrónico. En la siguiente página, escriba su contraseña.

   Para obtener ayuda para iniciar sesión con el usuario raíz, consulte Iniciar sesión como usuario raíz en la Guía del usuario de AWS Sign-In .
2. Active la autenticación multifactor (MFA) para el usuario raíz.

   Para obtener instrucciones, consulte Habilitar un dispositivo MFA virtual para el usuario Cuenta de AWS raíz (consola) en la Guía del usuario de IAM.

Creación de un usuario con acceso administrativo

1. Activar IAM Identity Center.

   Consulte las instrucciones en Activar AWS IAM Identity Center en la Guía del usuario de AWS IAM Identity Center .
2. En IAM Identity Center, conceda acceso administrativo a un usuario.

   Para ver un tutorial sobre su uso Directorio de IAM Identity Center como fuente de identidad, consulte Configurar el acceso de los usuarios con la configuración predeterminada Directorio de IAM Identity Center en la Guía del AWS IAM Identity Center usuario.

Inicio de sesión como usuario con acceso de administrador

- Para iniciar sesión con el usuario de IAM Identity Center, use la URL de inicio de sesión que se envió a la dirección de correo electrónico cuando creó el usuario de IAM Identity Center.

  Para obtener ayuda para iniciar sesión con un usuario del Centro de identidades de IAM, consulte Iniciar sesión en el portal de AWS acceso en la Guía del AWS Sign-In usuario.

Concesión de acceso a usuarios adicionales

1. En IAM Identity Center, cree un conjunto de permisos que siga la práctica recomendada de aplicar permisos de privilegios mínimos.

   Para conocer las instrucciones, consulte Create a permission set en la Guía del usuario de AWS IAM Identity Center .

2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

   Para conocer las instrucciones, consulte Add groups en la Guía del usuario de AWS IAM Identity Center .

Instale el AWS Command Line Interface

Si no lo ha instalado AWS CLI pero quiere usarlo, siga las instrucciones que aparecen en AWS Command Line Interface. En esta guía, asumimos que ha configurado el punto de conexión, la región y los detalles de autenticación, y omitimos estos parámetros en los comandos de ejemplo.

Cree una CA privada en AWS Private CA

Puede usar los procedimientos de esta sección para crear una raíz CAs o una subordinadaCAs, lo que dará como resultado una jerarquía auditable de relaciones de confianza que se ajuste a las necesidades de su organización. Puede crear una CA utilizando la parte de PCA de Consola de administración de AWS, o. AWS CLI AWS CloudFormation

Para obtener información sobre la actualización de la configuración de una CA que ya ha creado, consulte Actualice una CA privada en AWS Private Certificate Authority.

Para obtener información sobre el uso de una entidad de certificación para firmar certificados de entidad final para los usuarios, dispositivos y aplicaciones, consulte Emita certificados privados de entidad final.

> **Note**
>
> A su cuenta se le cobra un precio mensual por cada entidad de certificación privada a partir del momento en que la crea.

Para obtener la información Autoridad de certificación privada de AWS de precios más reciente, consulte AWS Private Certificate Authority Precios. También puede utilizar la calculadora de precios de AWS para estimar los costos.

Temas

- Ejemplos de CLI para crear una CA privada

Console

Para crear una entidad de certificación privada con la consola de

1. Utilice los siguientes pasos para crear una CA privada mediante la Consola de administración de AWS.

   Para comenzar a utilizar la consola

   Inicia sesión en tu AWS cuenta y abre la Autoridad de certificación privada de AWS consola en**https://console.aws.amazon.com/acm-pca/home**.

   - Si vas a abrir la consola en una región en la que no tienes acceso a una cuenta privada CAs, aparecerá la página de introducción. Elija Crear una CA privada.
   - Si va a abrir la consola en una región en la que ya ha creado una CA, se abrirá la página de autoridades de certificación privadas con una lista de las suyas CAs. Seleccione Crear CA.

2. En Opciones de modo, elija el modo de caducidad de los certificados que emita su CA.

   - Uso general: emite certificados que se pueden configurar con cualquier fecha de caducidad. Es el predeterminado.
   - Certificado de corta duración: emite certificados con un período de validez máximo de siete días. Un período de validez breve puede sustituir en algunos casos a un mecanismo de revocación.

3. En la sección Tipo de opciones de la consola, seleccione el tipo de entidad de certificación privada que desea crear.

- Al elegir Raíz, se establece una nueva jerarquía de entidades de certificación. Esta entidad de certificación está respaldada por un certificado autofirmado. Actúa como la autoridad de firma definitiva para otros certificados CAs y para los de la entidad final de la jerarquía.
- Al elegir una Subordinada se crea una entidad de certificación que debe estar firmada por una entidad de certificación (CA) principal encima de ella en la jerarquía. Los certificados subordinados se CAs utilizan normalmente para crear otros certificados subordinados CAs o para emitir certificados de entidad final a usuarios, ordenadores y aplicaciones.

> **Note**
>
> Autoridad de certificación privada de AWS proporciona un proceso de firma automatizado cuando la CA principal de la CA subordinada también está alojada en. Autoridad de certificación privada de AWS Todo lo que tiene que hacer es elegir la CA principal que va a utilizar.
> Es posible que su CA subordinada deba estar firmada por un proveedor de servicios de confianza externo. Si es así, le Autoridad de certificación privada de AWS proporciona una solicitud de firma de certificados (CSR) que debe descargar y utilizar para obtener un certificado de CA firmado. Para obtener más información, consulte Instale un certificado de CA subordinado firmado por una CA principal externa.

4. En las opciones de nombre distintivo del asunto, configure el nombre del asunto de su CA privada. Debe escribir al menos uno de los siguientes valores:

   - Organización (O): por ejemplo, el nombre de una empresa
   - Unidad organizativa (OU): por ejemplo, una división dentro de una empresa
   - Nombre del país (C): código de país de dos letras
   - Nombre de estado o provincia: nombre completo de un estado o una provincia
   - Nombre de la localidad: el nombre de una ciudad
   - Nombre común (CN): cadena legible por humanos para identificar la CA.

> **Note**
>
> Puede personalizar aún más el nombre del asunto de un certificado aplicando una APIPassthrough plantilla en el momento de su emisión. Para obtener más información y un ejemplo detallado, consulte Emita un certificado con un nombre de asunto personalizado mediante una plantilla APIPassthrough.

Dado que el certificado de respaldo se firma automáticamente, la información de asunto que proporciona para una entidad de certificación (CA) privada es probablemente más dispersa que la que podría contener una entidad de certificación pública. Para obtener más información acerca de cada uno de los valores que componen un nombre distintivo de asunto, consulte RFC 5280.

5. En Opciones de algoritmos clave, elija el algoritmo clave y la potencia del algoritmo. El valor predeterminado es RSA 2048. Puede elegir entre los siguientes algoritmos:

   - ML-DSA-44
   - ML-DSA-65
   - ML-DSA-87
   - RSA 2048
   - RSA 3072
   - RSA 4096
   - ECDSA P256
   - ECDSA P384
   - ECDSA P521

6. En Opciones de revocación de certificados, puede seleccionar entre dos métodos para compartir el estado de revocación con los clientes que utilizan sus certificados:

   - Activar la distribución de CRL
   - Encender OCSP

Puede configurar una de estas opciones de revocación, ninguna o ambas para su CA. Aunque es opcional, se recomienda la revocación gestionada como práctica recomendada. Antes de completar este paso, consulte Planifique el método de revocación AWS Private CA de su certificado para obtener información sobre las ventajas de cada método, la configuración preliminar que podría ser necesaria y las funciones de revocación adicionales.

> **Note**
>
> Si crea su CA sin configurar la revocación, siempre podrá configurarla más adelante. Para obtener más información, consulte Actualice una CA privada en AWS Private Certificate Authority.

Para configurar las opciones de revocación de certificados, lleve a cabo los siguientes pasos.

a. En Opciones de revocación de certificados, elija Activar la distribución de CRL.

b. En el URI del bucket de S3, elija un bucket existente de la lista.

   Al especificar un depósito existente, debe asegurarse de que el BPA esté desactivado para la cuenta y para el depósito. De lo contrario, se produce un error en la operación para crear la CA. Si la CA se ha creado correctamente, debe adjuntarle una política de forma manual antes de empezar a CRLs generarla. Utilice uno de los patrones de políticas descritos en Políticas de acceso para CRLs Amazon S3. Para obtener más información, consulte Agregar una política de bucket mediante la consola de Amazon S3.

c. Expanda Configuración de CRL para obtener opciones de configuración adicionales.

   - Seleccione Habilitar la partición para habilitar la partición de. CRLs Si no habilita la partición, su CA estará sujeta al número máximo de certificados revocados. Para obtener más información, consulte Cuotas de AWS Private Certificate Authority. Para obtener más información sobre las particiones CRLs, consulta los tipos de CRL.
   - Agregue un Nombre de CRL personalizado para crear un alias para el bucket de Amazon S3. Este nombre se incluye en los certificados emitidos por la entidad de certificación (CA) en la extensión “Puntos de distribución de CRL” definida por RFC 5280. Para usar CRLs over IPv6, configúrelo en el punto final S3 de doble pila de su bucket, tal y como se describe en Using over. CRLs IPv6

- Añada una ruta personalizada para crear un alias de DNS para la ruta del archivo en su bucket de Amazon S3.
- Introduzca la validez en días en los que su CRL seguirá siendo válida. El valor predeterminado es 7 días. En el caso de Internet CRLs, es habitual que el período de validez sea de 2 a 7 días. Autoridad de certificación privada de AWS intenta regenerar la CRL en el punto medio del período especificado.

7. Para ver las opciones de revocación de certificados, seleccione Activar OCSP.

- En el campo Punto de conexión de OCSP personalizado (opcional), puede proporcionar un nombre de dominio completo (FQDN) para un punto de conexión de OCSP que no sea de Amazon. Para usar OCSP over IPv6, defina este campo en un punto final de doble pila, tal y como se describe en Using OCSP over. IPv6

  Al proporcionar un FQDN en este campo, Autoridad de certificación privada de AWS inserta el FQDN en la extensión Authority Information Access de cada certificado emitido, en lugar de la URL predeterminada del respondedor OCSP. AWS Cuando un punto de conexión recibe un certificado que contiene el FQDN personalizado, consulte esa dirección para obtener una respuesta del OCSP. Para que este mecanismo funcione, debe realizar dos acciones adicionales:

  - Utilice un servidor proxy para reenviar el tráfico que llegue a su FQDN personalizado al respondedor de OCSP. AWS
  - Agregue el registro CNAME correspondiente a su base de datos DNS.

  > **Tip**
  >
  > Para obtener más información sobre la implementación de una solución OCSP completa mediante un CNAME personalizado, consulte Personaliza la URL de OCSP para AWS Private CA.

  Por ejemplo, este es un registro CNAME para un OCSP personalizado tal como aparecería en Amazon Route 53.

Nombre del registro	Tipo	Política de direccionamiento	Diferenciador	Valor/ruta de destino del tráfico
alternative.example.com	CNAME	Sencillez	-	proxy.example.com

> **Note**
>
> El valor de CNAME no debe incluir un prefijo de protocolo como “http://” o “https://”.

8. En Agregar etiquetas, puede etiquetar la entidad de certificación (CA) de forma opcional. Las etiquetas son pares de valor de clave que sirven como metadatos para identificar y organizar los recursos de AWS . Para obtener una lista de los parámetros de las Autoridad de certificación privada de AWS etiquetas y obtener instrucciones sobre cómo añadir etiquetas CAs después de crearlas, consulteAgregue etiquetas para su CA privada.

> **Note**
>
> Para adjuntar etiquetas a una entidad de certificación (CA) privada durante el procedimiento de creación, el administrador de una CA debe asociar primero a la acción `CreateCertificateAuthority` una política de IAM integrada y permitir el etiquetado de forma explícita. Para obtener más información, consulte Tag-on-create: Adjuntar etiquetas a una CA en el momento de su creación.

9. En las opciones de permisos de CA, puede delegar opcionalmente los permisos de renovación automática al director del AWS Certificate Manager servicio. ACM solo puede renovar automáticamente los certificados de entidades finales privadas generados por esta CA si se concede este permiso. Puede asignar permisos de renovación en cualquier momento con la Autoridad de certificación privada de AWS CreatePermissionAPI o el comando CLI create-permission.

El valor predeterminado consiste en habilitar estos permisos.

> **Note**
>
> AWS Certificate Manager no admite la renovación automática de certificados de corta duración.

10. En Precios, confirme que entiende los precios de una entidad de certificación privada.

> **Note**
>
> Para obtener la información Autoridad de certificación privada de AWS de precios más reciente, consulte AWS Private Certificate Authority Precios. También puede utilizar la calculadora de precios de AWS para estimar los costos.

11. Elija Crear CA después de comprobar que toda la información introducida es correcta. Se abre la página de detalles de la CA y muestra su estado como Certificado pendiente.

> **Note**
>
> En la página de detalles, puede terminar de configurar su CA seleccionando Acciones, Instalar el certificado de CA o puede volver más tarde a la lista de entidades de certificación privadas y completar el procedimiento de instalación que corresponda en su caso:
>
> - Instale un certificado de CA raíz
> - Instale un certificado de CA subordinado hospedado por Autoridad de certificación privada de AWS
> - Instale un certificado de CA subordinado firmado por una CA principal externa

CLI

Utilice el comando create-certificate-authority para crear una entidad de certificación privada. Debe especificar la configuración de la CA (que contiene información sobre el algoritmo y el nombre del asunto), la configuración de revocación (si planea usar OCSP and/or como CRL) y

el tipo de CA (raíz o subordinada). Los detalles de la configuración y la revocación se incluyen en dos archivos que se proporcionan como argumentos al comando. Si lo desea, también puede configurar el modo de uso de la CA (para emitir certificados estándar o de corta duración), adjuntar etiquetas y proporcionar un token de idempotencia.

Si está configurando una CRL, debe disponer de un bucket de Amazon S3 seguro antes de ejecutar el comando de create-certificate-authority. Para obtener más información, consulte Políticas de acceso para CRLs Amazon S3.

El archivo de configuración de la CA determina la siguiente información:

- El nombre del algoritmo
- El tamaño de la clave que se va a utilizar para crear la clave privada de CA
- El tipo de algoritmo de firma que la CA utiliza para firmar su propia solicitud de firma de certificados y las respuestas del OCSP CRLs
- La información del sujeto de X.500

La configuración de revocación de OCSP define un objeto de `OcspConfiguration` con la siguiente información:

- Establezca la marca `Enabled` en “true”.
- (Opcional) Un CNAME personalizado declarado como valor para `OcspCustomCname`.

La configuración de revocación de una CRL define un objeto de `CrlConfiguration` con la siguiente información:

- Establezca la marca `Enabled` en “true”.
- El periodo de caducidad de la CRL en días (periodo de validez de la CRL).
- El bucket de Amazon S3 que contendrá la CRL.
- (Opcional) Un ObjectAcl valor de S3 que determina si la CRL es de acceso público. En el ejemplo que se presenta aquí, el acceso público está bloqueado. Para obtener más información, consulte Habilite S3 Block Public Access (BPA) con CloudFront.
- (Opcional) Un alias CNAME para el bucket de S3 que se va a incluir en los certificados de la CA. Si la CRL no es de acceso público, apuntará a un mecanismo de distribución como Amazon CloudFront.

- (Opcional) Un `CrlDistributionPointExtensionConfiguration` objeto con la siguiente información:
  - El `OmitExtension` indicador está establecido en «verdadero» o «falso». Esto controla si el valor predeterminado de la extensión CDP se escribirá en un certificado emitido por la CA. Para obtener más información acerca de la extensión CDP, consulte. Determinación del URI del punto de distribución (CDP) de la CRL CustomCname No se puede establecer A si OmitExtension es «verdadero».
- (Opcional) Una ruta personalizada para la CRL del bucket de S3.
- (Opcional) Un CrlTypevalor que determina si la CRL estará completa o particionada. Si no se proporciona, la CRL se completará de forma predeterminada.

> **Note**
>
> Puede habilitar ambos mecanismos de revocación en la misma CA definiendo tanto un objeto `OcspConfiguration` como un objeto `CrlConfiguration`. Si no proporciona ningún parámetro --revocation-configuration, ambos mecanismos están deshabilitados de forma predeterminada. Si necesita soporte para la validación de la revocación más adelante, consulte Actualizar una CA (CLI).

Consulte la siguiente sección para ver ejemplos de CLI.

Ejemplos de CLI para crear una CA privada

En los ejemplos siguientes se supone que ha configurado el directorio de configuración `.aws` con una región, un punto de conexión y unas credenciales predeterminados válidos. Para obtener información sobre la configuración del AWS CLI entorno, consulte Ajustes de configuración y archivos de credenciales. Para facilitar la lectura, en los comandos de ejemplo proporcionamos la entrada de configuración y revocación de la CA como archivos JSON. Modifique los archivos de ejemplo según sea necesario para su uso.

Todos los ejemplos utilizan el siguiente archivo de configuración `ca_config.txt` a menos que se indique lo contrario.

Archivo: ca_config.txt

```
{
```

```
   "KeyAlgorithm":"RSA_2048",
   "SigningAlgorithm":"SHA256WITHRSA",
   "Subject":{
      "Country":"US",
      "Organization":"Example Corp",
      "OrganizationalUnit":"Sales",
      "State":"WA",
      "Locality":"Seattle",
      "CommonName":"www.example.com"
   }
}
```

Ejemplo 1: crear una CA con OCSP activado

En este ejemplo, el archivo de revocación habilita la compatibilidad predeterminada con OCSP, que utiliza el Autoridad de certificación privada de AWS respondedor para comprobar el estado del certificado.

Archivo: revoke_config.txt para OCSP

```
{
   "OcspConfiguration":{
      "Enabled":true
   }
}
```

Comando

```
$ aws acm-pca create-certificate-authority \
     --certificate-authority-configuration file://ca_config.txt \
     --revocation-configuration file://revoke_config.txt \
     --certificate-authority-type "ROOT" \
     --idempotency-token 01234567 \
     --tags Key=Name,Value=MyPCA
```

Si se ejecuta correctamente, este comando devuelve el nombre de recurso de Amazon (ARN) de la nueva CA.

```
{
 "CertificateAuthorityArn":"arn:aws:acm-pca:region:account:
       certificate-authority/CA_ID"
```

```
}
```

Comando

```
$ aws acm-pca create-certificate-authority \
 --certificate-authority-configuration file://ca_config.txt \
 --revocation-configuration file://revoke_config.txt \
 --certificate-authority-type "ROOT" \
 --idempotency-token 01234567 \
 --tags Key=Name,Value=MyPCA-2
```

Si se ejecuta correctamente, este comando devuelve el nombre de recurso de Amazon (ARN) de la CA.

```
{
    "CertificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
}
```

Utilice el siguiente comando para inspeccionar la configuración de su CA.

```
$ aws acm-pca describe-certificate-authority \
      --certificate-authority-arn "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566" \
      --output json
```

Esta descripción debe contener la siguiente sección.

```
"RevocationConfiguration": {
   ...
   "OcspConfiguration": {
      "Enabled": true
   }
   ...
}
```

Ejemplo 2: crear una CA con OCSP activado y un CNAME personalizado y habilitado

En este ejemplo, el archivo de revocación permite la compatibilidad personalizada con OCSP. El parámetro `OcspCustomCname` toma un nombre de dominio completo (FQDN) como valor.

Al proporcionar un FQDN en este campo, Autoridad de certificación privada de AWS inserta el FQDN en la extensión Authority Information Access de cada certificado emitido, en lugar de la URL predeterminada del respondedor OCSP. AWS Cuando un punto de conexión recibe un certificado que contiene el FQDN personalizado, consulte esa dirección para obtener una respuesta del OCSP. Para que este mecanismo funcione, debe realizar dos acciones adicionales:

- Utilice un servidor proxy para reenviar el tráfico que llegue a su FQDN personalizado al respondedor de OCSP. AWS
- Agregue el registro CNAME correspondiente a su base de datos DNS.

> **Tip**
>
> Para obtener más información sobre la implementación de una solución OCSP completa mediante un CNAME personalizado, consulte Personaliza la URL de OCSP para AWS Private CA.

Por ejemplo, este es un registro CNAME para un OCSP personalizado tal como aparecería en Amazon Route 53.

Nombre del registro	Tipo	Política de direccionamiento	Diferenciador	Valor/ruta de destino del tráfico
alternative.example.com	CNAME	Sencillez	-	proxy.example.com

> **Note**
>
> El valor de CNAME no debe incluir un prefijo de protocolo como “http://” o “https://”.

Archivo: revoke_config.txt para OCSP

```
{
   "OcspConfiguration":{
```

```
      "Enabled":true,
      "OcspCustomCname":"alternative.example.com"
   }
}
```

Comando

```
$ aws acm-pca create-certificate-authority \
 --certificate-authority-configuration file://ca_config.txt \
 --revocation-configuration file://revoke_config.txt \
 --certificate-authority-type "ROOT" \
 --idempotency-token 01234567 \
 --tags Key=Name,Value=MyPCA-3
```

Si se ejecuta correctamente, este comando devuelve el nombre de recurso de Amazon (ARN) de la CA.

```
{
    "CertificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
}
```

Utilice el siguiente comando para inspeccionar la configuración de su CA.

```
$ aws acm-pca describe-certificate-authority \
      --certificate-authority-arn "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566" \
      --output json
```

Esta descripción debe contener la siguiente sección.

```
"RevocationConfiguration": {
   ...
   "OcspConfiguration": {
      "Enabled": true,
      "OcspCustomCname": "alternative.example.com"
   }
   ...
}
```

Ejemplo 3: crear una CA con una CRL asociada

En este ejemplo, la configuración de revocación define los parámetros de la CRL.

Archivo: revoke_config.txt

```
{
   "CrlConfiguration":{
      "Enabled":true,
      "ExpirationInDays":7,
      "S3BucketName":"amzn-s3-demo-bucket"
   }
}
```

Comando

```
$ aws acm-pca create-certificate-authority \
      --certificate-authority-configuration file://ca_config.txt \
      --revocation-configuration file://revoke_config.txt \
      --certificate-authority-type "ROOT" \
      --idempotency-token 01234567 \
      --tags Key=Name,Value=MyPCA-1
```

Si se ejecuta correctamente, este comando devuelve el nombre de recurso de Amazon (ARN) de la CA.

```
{
    "CertificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
}
```

Utilice el siguiente comando para inspeccionar la configuración de su CA.

```
$ aws acm-pca describe-certificate-authority \
      --certificate-authority-arn "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566" \
      --output json
```

Esta descripción debe contener la siguiente sección.

```
"RevocationConfiguration": {
```

```
   ...
   "CrlConfiguration": {
      "Enabled": true,
      "ExpirationInDays": 7,
      "S3BucketName": "amzn-s3-demo-bucket"
   },
   ...
}
```

Ejemplo 4: crear una CA con una CRL asociada y un CNAME personalizado y habilitado

En este ejemplo, la configuración de revocación define los parámetros de la CRL que incluye un CNAME personalizado.

Archivo: revoke_config.txt

```
{
   "CrlConfiguration":{
      "Enabled":true,
      "ExpirationInDays":7,
      "CustomCname": "alternative.example.com",
      "S3BucketName":"amzn-s3-demo-bucket"
   }
}
```

Comando

```
$ aws acm-pca create-certificate-authority \
      --certificate-authority-configuration file://ca_config.txt \
      --revocation-configuration file://revoke_config.txt \
      --certificate-authority-type "ROOT" \
      --idempotency-token 01234567 \
      --tags Key=Name,Value=MyPCA-1
```

Si se ejecuta correctamente, este comando devuelve el nombre de recurso de Amazon (ARN) de la CA.

```
{
    "CertificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
```

```
}
```

Utilice el siguiente comando para inspeccionar la configuración de su CA.

```
$ aws acm-pca describe-certificate-authority \
      --certificate-authority-arn "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566" \
      --output json
```

Esta descripción debe contener la siguiente sección.

```
"RevocationConfiguration": {
   ...
   "CrlConfiguration": {
      "Enabled": true,
      "ExpirationInDays": 7,
      "CustomCname": "alternative.example.com",
      "S3BucketName": "amzn-s3-demo-bucket",
   ...
   }
}
```

Ejemplo 5: crear una CA y especificar el modo de uso

En este ejemplo, el modo de uso de la CA se especifica al crear una CA. Si no se especifica, el parámetro del modo de uso se establece de forma predeterminada en GENERAL_PURPOSE. En este ejemplo, el parámetro se establece en SHORT_LIVED_CERTIFICATE, lo que significa que la CA emitirá certificados con un período de validez máximo de siete días. En situaciones en las que no es conveniente configurar la revocación, un certificado de corta duración que se haya visto comprometido caduca rápidamente como parte de las operaciones normales. En consecuencia, este ejemplo de CA carece de un mecanismo de revocación.

> **Note**
>
> Autoridad de certificación privada de AWS no comprueba la validez de los certificados de CA raíz.

```
$ aws acm-pca create-certificate-authority \
```

```
      --certificate-authority-configuration file://ca_config.txt \
      --certificate-authority-type "ROOT" \
      --usage-mode SHORT_LIVED_CERTIFICATE \
      --tags Key=usageMode,Value=SHORT_LIVED_CERTIFICATE
```

Utilice el describe-certificate-authoritycomando de AWS CLI para mostrar detalles sobre la CA resultante, como se muestra en el siguiente comando:

```
$ aws acm-pca describe-certificate-authority \
      --certificate-authority-arn arn:aws:acm:region:account:certificate-
authority/CA_ID
```

```
{
    "CertificateAuthority":{
       "Arn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID",
       "CreatedAt":"2022-09-30T09:53:42.769000-07:00",
       "LastStateChangeAt":"2022-09-30T09:53:43.784000-07:00",
       "Type":"ROOT",
       "UsageMode":"SHORT_LIVED_CERTIFICATE",
       "Serial":"serial_number",
       "Status":"PENDING_CERTIFICATE",
       "CertificateAuthorityConfiguration":{
          "KeyAlgorithm":"RSA_2048",
          "SigningAlgorithm":"SHA256WITHRSA",
          "Subject":{
             "Country":"US",
             "Organization":"Example Corp",
             "OrganizationalUnit":"Sales",
             "State":"WA",
             "Locality":"Seattle",
             "CommonName":"www.example.com"
          }
       },
       "RevocationConfiguration":{
          "CrlConfiguration":{
             "Enabled":false
          },
          "OcspConfiguration":{
             "Enabled":false
          }
       },
 ...
```

Ejemplo 6: crear una CA para iniciar sesión en Active Directory

Puede crear una CA privada adecuada para su uso en el NTAuth almacén empresarial de Microsoft Active Directory (AD), donde podrá emitir certificados de inicio de sesión con tarjeta o de controlador de dominio. Para obtener información sobre la importación de un certificado de CA a AD, consulte Cómo importar certificados de entidades emisoras de certificados (CA) de terceros al almacén empresarial. NTAuth

La herramienta certutil de Microsoft se puede utilizar para publicar certificados de CA en AD invocando la opción -dspublish. Un certificado publicado en AD con certutil es de confianza en todo el bosque. Con la política de grupo, también puede limitar la confianza a un subconjunto de todo el bosque, por ejemplo, a un único dominio o a un grupo de computadoras de un dominio. Para que el inicio de sesión funcione, la CA emisora también debe estar publicada en la NTAuth tienda. Para obtener más información, consulte Distribuir certificados a computadoras de clientes mediante la política de grupo.

Este ejemplo usa el siguiente archivo de configuración `ca_config_AD.txt`.

Archivo: ca_config_AD.txt

```
{
   "KeyAlgorithm":"RSA_2048",
   "SigningAlgorithm":"SHA256WITHRSA",
   "Subject":{
      "CustomAttributes":[
         {
            "ObjectIdentifier":"2.5.4.3",
            "Value":"root CA"
         },
         {
            "ObjectIdentifier":"0.9.2342.19200300.100.1.25",
            "Value":"example"
         },
         {
            "ObjectIdentifier":"0.9.2342.19200300.100.1.25",
            "Value":"com"
         }
      ]
   }
}
```

Comando

```
$ aws acm-pca create-certificate-authority \
      --certificate-authority-configuration file://ca_config_AD.txt \
      --certificate-authority-type "ROOT" \
      --tags Key=application,Value=ActiveDirectory
```

Si se ejecuta correctamente, este comando devuelve el nombre de recurso de Amazon (ARN) de la CA.

```
{
 "CertificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
 }
```

Utilice el siguiente comando para inspeccionar la configuración de su CA.

```
$ aws acm-pca describe-certificate-authority \
      --certificate-authority-arn "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566" \
      --output json
```

Esta descripción debe contener la siguiente sección.

```
...

"Subject":{
   "CustomAttributes":[
      {
         "ObjectIdentifier":"2.5.4.3",
         "Value":"root CA"
      },
      {
         "ObjectIdentifier":"0.9.2342.19200300.100.1.25",
         "Value":"example"
      },
      {
         "ObjectIdentifier":"0.9.2342.19200300.100.1.25",
         "Value":"com"
      }
   ]
}
...
```

Ejemplo 7: Crear una CA Matter con una CRL adjunta y omitir la extensión CDP en los certificados emitidos

Puede crear una CA privada adecuada para emitir certificados para el estándar de hogar inteligente Matter. En este ejemplo, la configuración de CA `ca_config_PAA.txt` define una autoridad de certificación de productos (PAA) de Matter con el identificador de proveedor (VID) establecido en. FFF1

Archivo: ca_config_PAA.txt

```
{
   "KeyAlgorithm":"EC_prime256v1",
   "SigningAlgorithm":"SHA256WITHECDSA",
   "Subject":{
      "Country":"US",
      "Organization":"Example Corp",
      "OrganizationalUnit":"SmartHome",
      "State":"WA",
      "Locality":"Seattle",
      "CommonName":"Example Corp Matter PAA",
   "CustomAttributes":[
      {
        "ObjectIdentifier":"1.3.6.1.4.1.37244.2.1",
        "Value":"FFF1"
      }
    ]
  }
}
```

La configuración de revocación habilita CRLs y configura la CA para que omita la URL de CDP predeterminada de todos los certificados emitidos.

Archivo: revoke_config.txt

```
{
   "CrlConfiguration":{
      "Enabled":true,
      "ExpirationInDays":7,
      "S3BucketName":"amzn-s3-demo-bucket",
   "CrlDistributionPointExtensionConfiguration":{
  "OmitExtension":true
   }
```

```
   }
}
```

Comando

```
$ aws acm-pca create-certificate-authority \
      --certificate-authority-configuration file://ca_config_PAA.txt \
      --revocation-configuration file://revoke_config.txt \
      --certificate-authority-type "ROOT" \
      --idempotency-token 01234567 \
      --tags Key=Name,Value=MyPCA-1
```

Si se ejecuta correctamente, este comando devuelve el nombre de recurso de Amazon (ARN) de la CA.

```
{
    "CertificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
}
```

Utilice el siguiente comando para inspeccionar la configuración de su CA.

```
$ aws acm-pca describe-certificate-authority \
      --certificate-authority-arn "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566" \
      --output json
```

Esta descripción debe contener la siguiente sección.

```
"RevocationConfiguration": {
   ...
   "CrlConfiguration": {
      "Enabled": true,
      "ExpirationInDays": 7,
      "S3BucketName": "amzn-s3-demo-bucket",
   "CrlDistributionPointExtensionConfiguration":{
  "OmitExtension":true
   }
   },
   ...
}
```

```
...
```

Instalación del certificado de CA

Complete los siguientes procedimientos para crear e instalar el certificado de entidad de certificación privado. A continuación, su entidad de certificación estará lista para su uso.

Autoridad de certificación privada de AWS admite tres escenarios para instalar un certificado de CA:

- Instalación de un certificado para una CA raíz alojada por Autoridad de certificación privada de AWS
- Instalación de un certificado de entidad de certificación subordinada cuya autoridad principal está alojada por Autoridad de certificación privada de AWS
- Instalación de un certificado de entidad de certificación subordinada cuya autoridad principal está alojada externamente

En las secciones siguientes se describen los procedimientos para cada supuesto. Los procedimientos de la consola comienzan en la página Privada de la consola CAs.

Algoritmos de firma compatibles

La compatibilidad con los algoritmos de firma de los certificados de CA depende del algoritmo de firma de la CA principal y del Región de AWS. Las siguientes restricciones se aplican tanto a la consola como a las AWS CLI operaciones.

- Una entidad emisora de certificados principal con el algoritmo de claves RSA puede emitir certificados con los siguientes algoritmos de firma:
  - SHA256 RSA
  - SHA384 RSA
  - SHA512 RSA
- En una versión antigua Región de AWS, una entidad emisora de certificados principal con el algoritmo de claves EDCSA podía emitir certificados con los siguientes algoritmos de firma:
  - SHA256 ECDSA
  - SHA384 ECDSA
  - SHA512 ECDSA

El legado incluye Regiones de AWS :

Nombre de la región	Ubicación geográfica
eu-north-1	Europa (Estocolmo)
me-south-1	Middle East (Bahrain)
ap-south-1	Asia-Pacífico (Mumbai)
eu-west-3	Europa (París)
us-east-2	Este de EE. UU. (Ohio)
af-south-1	África (Ciudad del Cabo)
eu-west-1	Europa (Irlanda)
eu-central-1	Europa (Fráncfort)
sa-east-1	América del Sur (São Paulo)
ap-east-1	Asia-Pacífico (Hong Kong)
us-east-1	Este de EE. UU. (Norte de Virginia)
ap-northeast-2	Asia-Pacífico (Seúl)
eu-west-2	Europa (Londres)

Nombre de la región	Ubicación geográfica
ap-northeast-1	Asia-Pacífico (Tokio)
us-gov-east-1	AWS GovCloud (Este de EE. UU.)
us-gov-west-1	AWS GovCloud (Estados Unidos-Oeste)
us-west-2	Oeste de EE. UU. (Oregón)
us-west-1	Oeste de EE. UU. (Norte de California)
ap-southeast-1	Asia-Pacífico (Singapur)
ap-southeast-2	Asia-Pacífico (Sídney)

- En el caso de una entidad no heredada Región de AWS, se aplican las siguientes reglas a la EDCSA:
  - Una entidad de certificación principal con el algoritmo de firma EC_prime256v1 puede emitir certificados con el ECDSA P256.
  - Una entidad de certificación principal con el algoritmo de firma EC_secp384r1 puede emitir certificados con el ECDSA P384.
- En cada caso Región de AWS, se aplican las siguientes reglas a la EDCSA:
  - Una entidad emisora de certificados principal con el algoritmo de firma EC_SECP521R1 puede emitir certificados con el ECDSA P521.

Instale un certificado de CA raíz

Puede instalar un certificado de CA raíz desde Consola de administración de AWS o desde AWS CLI.

Para crear e instalar un certificado para su entidad de certificación raíz privada (consola)

1. (Opcional) Si aún no está en la página de detalles de la CA, abra la Autoridad de certificación privada de AWS consola de https://console.aws.amazon.com/acm-pca/su casa. En la página de autoridades de certificación privadas, elija una CA raíz con el estado Certificado pendiente o Activo.
2. Seleccione Acciones, Instalar el certificado de CA para abrir la página Instalación del certificado de CA raíz.
3. En la sección Especifique los parámetros del certificado de CA raíz, especifique los siguientes parámetros de certificado:

   - Validez: especifica la fecha y la hora de caducidad del certificado de CA. El período de validez Autoridad de certificación privada de AWS predeterminado de un certificado de CA raíz es de 10 años.
   - Algoritmo de firma: especifica el algoritmo de firma que se utilizará cuando la CA raíz emita nuevos certificados. Las opciones disponibles varían en función del Región de AWS lugar en el que se cree la CA. Para obtener más informaciónAlgoritmos de firma compatibles, consulteAlgoritmos criptográficos compatibles en AWS Private Certificate Authority, y SigningAlgorithmen CertificateAuthorityConfiguration.
     - SHA256 RSA
     - SHA384 RSA
     - SHA512 RSA

   Revise la configuración para comprobar que es correcta y, a continuación, seleccione Confirmar e instalar. Autoridad de certificación privada de AWS exporta una CSR para su entidad emisora de certificados, genera un certificado mediante una plantilla de certificado de entidad emisora raíz y firma automáticamente el certificado. Autoridad de certificación privada de AWS a continuación, importa el certificado de CA raíz autofirmado.
4. La página de detalles de la CA muestra el estado de la instalación (éxito o error) en la parte superior. Si la instalación se realizó correctamente, la entidad de certificación raíz recién completada muestra el estado Activo en el panel General.

Para crear e instalar un certificado para su entidad de certificación raíz privada (AWS CLI)

1. Genere una solicitud de firma de certificado (CSR).

```
$ aws acm-pca get-certificate-authority-csr \
     --certificate-authority-arn arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566 \
     --output text \
     --region region > ca.csr
```

El archivo resultante `ca.csr`, un archivo PEM codificado en formato base64, tiene el siguiente aspecto.

```
-----BEGIN CERTIFICATE REQUEST-----
MIIC1DCCAbwCAQAwbTELMAkGA1UEBhMCVVMxFTATBgNVBAoMDEV4YW1wbGUgQ29y
cDEOMAwGA1UECwwFU2FsZXMxCzAJBgNVBAgMAldBMRgwFgYDVQQDDA93d3cuZXhh
bXBsZS5jb20xEDAOBgNVBAcMB1NlYXR0bGUwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQDD+7eQChWUO2m6pHslI7AVSFkWvbQofKIHvbvy7wm8VO9/BuI7
LE/jrnd1jGoyI7jaMHKXPtEP3uNlCzv+oEza07OjgjqPZVehtA6a3/3vdQ1qCoD2
rXpv6VIzcq2onx2X7m+Zixwn2oY1l1ELXP7I5g0GmUStymq+pY5VARPy3vTRMjgC
JEiz8w7VvC15uIsHFAWa2/NvKyndQMPaCNft238wesV5s2cXOUS173jghIShg99o
ymf0TRUgvAGQMCXvsW07MrP5VDmBU7k/AZ9ExsUfMe2OB++fhfQWr2N7/lpC4+DP
qJTfXTEexLfRTLeLuGEaJL+c6fMyG+Yk53tZAgMBAAGgIjAgBgkqhkiG9w0BCQ4x
EzARMA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEBAA7xxLVI5s1B
qmXMMT44y1DZtQx3RDPanMNGLGO1TmLtyqqnUH49Tla+2p7nrl0tojUf/3PaZ52F
QN09SrFk8qtYSKnMGd5PZL0A+NFsNW+w4BAQNKlg9m617YEsnkztbfKRloaJNYoA
HZaRvbA0lMQ/tU2PKZR2vnao444Ugm0O/t3jx5rj817b31hQcHHQ0lQuXV2kyTrM
ohWeLf2fL+K0xJ9ZgXD4KYnY0zarpreA5RBeO5xs3Ms+oGWc13qQfMBx33vrrz2m
dw5iKjg71uuUUmtDV6ewwGa/VO5hNinYAfogdu5aGuVbnTFT3n45B8WHz2+9r0dn
bA7xUel1SuQ=
-----END CERTIFICATE REQUEST-----
```

Puede usar OpenSSL para ver y verificar el contenido de la CSR.

```
openssl req -text -noout -verify -in ca.csr
```

Su resultado es similar al siguiente.

```
verify OK
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=US, O=Example Corp, OU=Sales, ST=WA, CN=www.example.com,
 L=Seattle
```

```
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c3:fb:b7:90:0a:15:94:3b:69:ba:a4:7b:25:23:
                    b0:15:48:59:16:bd:b4:28:7c:a2:07:bd:bb:f2:ef:
                    09:bc:54:ef:7f:06:e2:3b:2c:4f:e3:ae:77:75:8c:
                    6a:32:23:b8:da:30:72:97:3e:d1:0f:de:e3:65:0b:
                    3b:fe:a0:4c:da:d3:b3:a3:82:3a:8f:65:57:a1:b4:
                    0e:9a:df:fd:ef:75:0d:6a:0a:80:f6:ad:7a:6f:e9:
                    52:33:72:ad:a8:9f:1d:97:ee:6f:99:8b:1c:27:da:
                    86:35:97:51:0b:5c:fe:c8:e6:0d:06:99:44:ad:ca:
                    6a:be:a5:8e:55:01:13:f2:de:f4:d1:32:38:02:24:
                    48:b3:f3:0e:d5:bc:2d:79:b8:8b:07:14:05:9a:db:
                    f3:6f:2b:29:dd:40:c3:da:08:d7:ed:db:7f:30:7a:
                    c5:79:b3:67:17:39:44:b5:ef:78:e0:84:84:a1:83:
                    df:68:ca:67:f4:4d:15:20:bc:01:90:30:25:ef:b1:
                    6d:3b:32:b3:f9:54:39:81:53:b9:3f:01:9f:44:c6:
                    c5:1f:31:ed:8e:07:ef:9f:85:f4:16:af:63:7b:fe:
                    5a:42:e3:e0:cf:a8:94:df:5d:31:1e:c4:b7:d1:4c:
                    b7:8b:b8:61:1a:24:bf:9c:e9:f3:32:1b:e6:24:e7:
                    7b:59
                Exponent: 65537 (0x10001)
        Attributes:
        Requested Extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         0e:f1:c4:b5:48:e6:cd:41:aa:65:cc:31:3e:38:cb:50:d9:b5:
         0c:77:44:33:da:9c:c3:46:2c:63:b5:4e:62:ed:ca:aa:a7:50:
         7e:3d:4e:56:be:da:9e:e7:ae:5d:2d:a2:35:1f:ff:73:da:67:
         9d:85:40:dd:3d:4a:b1:64:f2:ab:58:48:a9:cc:19:de:4f:64:
         bd:00:f8:d1:6c:35:6f:b0:e0:10:10:34:a9:60:f6:6e:b5:ed:
         81:2c:9e:4c:ed:6d:f2:91:96:86:89:35:8a:00:1d:96:91:bd:
         b0:34:94:c4:3f:b5:4d:8f:29:94:76:be:76:a8:e3:8e:14:82:
         6d:0e:fe:dd:e3:c7:9a:e3:f3:5e:db:df:58:50:70:71:d0:d2:
         54:2e:5d:5d:a4:c9:3a:cc:a2:15:9e:2d:fd:9f:2f:e2:b4:c4:
         9f:59:81:70:f8:29:89:d8:d3:36:ab:a6:b7:80:e5:10:5e:3b:
         9c:6c:dc:cb:3e:a0:65:9c:d7:7a:90:7c:c0:71:df:7b:eb:af:
         3d:a6:77:0e:62:2a:38:3b:d6:eb:94:52:6b:43:57:a7:b0:c0:
         66:bf:54:ee:61:36:29:d8:01:fa:20:76:ee:5a:1a:e5:5b:9d:
         31:53:de:7e:39:07:c5:87:cf:6f:bd:af:47:67:6c:0e:f1:51:
         e9:75:4a:e4
```

2. Utilice la CSR del paso anterior como argumento para el parámetro `--csr` y emita el certificado raíz.

> **Note**
>
> Si utiliza la AWS CLI versión 1.6.3 o posterior, utilice el prefijo `fileb://` al especificar el archivo de entrada necesario. Esto garantiza que analice correctamente los Autoridad de certificación privada de AWS datos codificados en Base64.

```
$ aws acm-pca issue-certificate \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --csr file://ca.csr \
     --signing-algorithm SHA256WITHRSA \
     --template-arn arn:aws:acm-pca:::template/RootCACertificate/V1 \
     --validity Value=365,Type=DAYS
```

3. Recupere el certificado de raíz.

```
$ aws acm-pca get-certificate \
 --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
 --certificate-arn arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID \
 --output text > cert.pem
```

El archivo resultante `cert.pem`, un archivo PEM codificado en formato base64, tiene el siguiente aspecto.

```
-----BEGIN CERTIFICATE-----
MIIDpzCCAo+gAwIBAgIRAIIuOarlQETlUQEOZJGZYdIwDQYJKoZIhvcNAQELBQAw
bTELMAkGA1UEBhMCVVMxFTATBgNVBAoMDEV4YW1wbGUgQ29ycDEOMAwGA1UECwwF
U2FsZXMxCzAJBgNVBAgMAldBMRgwFgYDVQQDDA93d3cuZXhhbXBsZS5jb20xEDAO
BgNVBAcMB1NlYXR0bGUwHhcNMjEwMzA4MTU0NjI3WhcNMjIwMzA4MTY0NjI3WjBt
MQswCQYDVQQGEwJVUzEVMBMGA1UECgwMRXhhbXBsZSBDb3JwMQ4wDAYDVQQLDAVT
YWxlczELMAkGA1UECAwCV0ExGDAWBgNVBAMMD3d3dy5leGFtcGxlLmNvbTEQMA4G
A1UEBwwHU2VhdHRsZTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMP7
t5AKFZQ7abqkeyUjsBVIWRa9tCh8oge9u/LvCbxU738G4jssT+Oud3WMajIjuNow
cpc+0Q/e42ULO/6gTNrTs6OCOo9lV6G0Dprf/e91DWoKgPatem/pUjNyraifHZfu
b5mLHCfahjWXUQtc/sjmDQaZRK3Kar6ljlUBE/Le9NEyOAIkSLPzDtW8LXm4iwcU
```

```
BZrb828rKd1Aw9oI1+3bfzB6xXmzZxc5RLXveOCEhKGD32jKZ/RNFSC8AZAwJe+x
bTsys/lUOYFTuT8Bn0TGxR8x7Y4H75+F9BavY3v+WkLj4M+olN9dMR7Et9FMt4u4
YRokv5zp8zIb5iTne1kCAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4E
FgQUaW3+r328uTLokog2TklmoBK+yt4wDgYDVR0PAQH/BAQDAgGGMA0GCSqGSIb3
DQEBCwUAA4IBAQAXjd/7UZ8RDE+PLWSDNGQdLemOBTcawF+tK+PzA4Evlmn9VuNc
g+x3oZvVZSDQBANUz0b9oPeo54aE38dW1zQm2qfTab8822aqeWMLyJ1dMsAgqYX2
t9+u6w3NzRCw8Pvz18V69+dFE5AeXmNP0Z5/gdz8H/NSpctjlzopbScRZKCSlPid
Rf3ZOPm9QP92YpWyYDkfAU04xdDo1vR0MYjKPkl4LjRqSU/tcCJnPMbJiwq+bWpX
2WJoEBXB/p15Kn6JxjI0ze2SnSI48JZ8it4fvxrhOo0VoLNIuCuNXJOwU17Rdl1W
YJidaq7je6k18AdgPA0Kh8y1XtfUH3fTaVw4
-----END CERTIFICATE-----
```

Puede usar OpenSSL para ver y verificar el contenido del certificado.

```
openssl x509 -in cert.pem -text -noout
```

Su resultado es similar al siguiente.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            82:2e:39:aa:e5:40:44:e5:51:01:0e:64:91:99:61:d2
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=Example Corp, OU=Sales, ST=WA, CN=www.example.com,
 L=Seattle
        Validity
            Not Before: Mar  8 15:46:27 2021 GMT
            Not After : Mar  8 16:46:27 2022 GMT
        Subject: C=US, O=Example Corp, OU=Sales, ST=WA, CN=www.example.com,
 L=Seattle
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c3:fb:b7:90:0a:15:94:3b:69:ba:a4:7b:25:23:
                    b0:15:48:59:16:bd:b4:28:7c:a2:07:bd:bb:f2:ef:
                    09:bc:54:ef:7f:06:e2:3b:2c:4f:e3:ae:77:75:8c:
                    6a:32:23:b8:da:30:72:97:3e:d1:0f:de:e3:65:0b:
                    3b:fe:a0:4c:da:d3:b3:a3:82:3a:8f:65:57:a1:b4:
                    0e:9a:df:fd:ef:75:0d:6a:0a:80:f6:ad:7a:6f:e9:
                    52:33:72:ad:a8:9f:1d:97:ee:6f:99:8b:1c:27:da:
```

```
                86:35:97:51:0b:5c:fe:c8:e6:0d:06:99:44:ad:ca:
                6a:be:a5:8e:55:01:13:f2:de:f4:d1:32:38:02:24:
                48:b3:f3:0e:d5:bc:2d:79:b8:8b:07:14:05:9a:db:
                f3:6f:2b:29:dd:40:c3:da:08:d7:ed:db:7f:30:7a:
                c5:79:b3:67:17:39:44:b5:ef:78:e0:84:84:a1:83:
                df:68:ca:67:f4:4d:15:20:bc:01:90:30:25:ef:b1:
                6d:3b:32:b3:f9:54:39:81:53:b9:3f:01:9f:44:c6:
                c5:1f:31:ed:8e:07:ef:9f:85:f4:16:af:63:7b:fe:
                5a:42:e3:e0:cf:a8:94:df:5d:31:1e:c4:b7:d1:4c:
                b7:8b:b8:61:1a:24:bf:9c:e9:f3:32:1b:e6:24:e7:
                7b:59
            Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
                69:6D:FE:AF:7D:BC:B9:32:E8:92:88:36:4E:49:66:A0:12:BE:CA:DE
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
    Signature Algorithm: sha256WithRSAEncryption
         17:8d:df:fb:51:9f:11:0c:4f:8f:2d:64:83:34:64:1d:2d:e9:
         8e:05:37:1a:c0:5f:ad:2b:e3:f3:03:81:2f:96:69:fd:56:e3:
         5c:83:ec:77:a1:9b:d5:65:20:d0:04:03:54:cf:46:fd:a0:f7:
         a8:e7:86:84:df:c7:56:d7:34:26:da:a7:d3:69:bf:3c:db:66:
         aa:79:63:0b:c8:9d:5d:32:c0:20:a9:85:f6:b7:df:ae:eb:0d:
         cd:cd:10:b0:f0:fb:f3:d7:c5:7a:f7:e7:45:13:90:1e:5e:63:
         4f:d1:9e:7f:81:dc:fc:1f:f3:52:a5:cb:63:97:3a:29:6d:27:
         11:64:a0:92:94:f8:9d:45:fd:d9:38:f9:bd:40:ff:76:62:95:
         b2:60:39:1f:01:4d:38:c5:d0:e8:d6:f4:74:31:88:ca:3e:49:
         78:2e:34:6a:49:4f:ed:70:22:67:3c:c6:c9:8b:0a:be:6d:6a:
         57:d9:62:68:10:15:c1:fe:9d:79:2a:7e:89:c6:32:34:cd:ed:
         92:9d:22:38:f0:96:7c:8a:de:1f:bf:1a:e1:3a:8d:15:a0:b3:
         48:b8:2b:8d:5c:93:b0:53:5e:d1:76:5d:56:60:98:9d:6a:ae:
         e3:7b:a9:35:f0:07:60:3c:0d:0a:87:cc:b5:5e:d7:d4:1f:77:
         d3:69:5c:38
```

4. Importe el certificado de CA raíz para instalarlo en la CA.

> **Note**
>
> Si utiliza la AWS CLI versión 1.6.3 o posterior, utilice el prefijo `fileb://` al especificar el archivo de entrada necesario. Esto garantiza que analice correctamente los Autoridad de certificación privada de AWS datos codificados en Base64.

```
$ aws acm-pca import-certificate-authority-certificate \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --certificate file://cert.pem
```

Inspeccione el nuevo estado de la CA.

```
$ aws acm-pca describe-certificate-authority \
 --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
 --output json
```

El estado ahora aparece como ACTIVO.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-05T14:24:12.867000-08:00",
        "LastStateChangeAt": "2021-03-08T12:37:14.235000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T07:46:27-08:00",
        "NotAfter": "2022-03-08T08:46:27-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
```

```
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "CustomCname": "alternative.example.com",
                "S3BucketName": "amzn-s3-demo-bucket"
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

Instale un certificado de CA subordinado hospedado por Autoridad de certificación privada de AWS

Puede usarlo Consola de administración de AWS para crear e instalar un certificado para su CA Autoridad de certificación privada de AWS subordinada alojada.

Para crear e instalar un certificado para la CA Autoridad de certificación privada de AWS subordinada alojada

1. (Opcional) Si aún no está en la página de detalles de la CA, abra la Autoridad de certificación privada de AWS consola de https://console.aws.amazon.com/acm-pca/su casa. En la página Autoridades de certificación privadas, elija una CA raíz con el estado Certificado pendiente o Activo.
2. Seleccione Acciones, Instalar el certificado de CA para abrir la página Instalación del certificado de CA subordinado.
3. En la página Instalar un certificado de CA subordinado, en Seleccione el tipo de CA, elija AWS Private CAinstalar un certificado gestionado por Autoridad de certificación privada de AWS.
4. En Seleccionar la CA principal, elija una CA de la lista de CA privada principal. La lista se filtra para mostrar los CAs que cumplen los siguientes criterios:

    - Debe tener permisos para usar la CA.

- La CA no firmaría sola.
- La CA está en el estado `ACTIVE`.
- El modo CA es `GENERAL_PURPOSE`.

5. En la sección Especifique los parámetros del certificado de entidad de certificación subordinada, especifique los siguientes parámetros de certificado:

   - Validez: especifica la fecha y la hora de caducidad del certificado de CA.
   - Algoritmo de firma: especifica el algoritmo de firma que se utilizará cuando la CA raíz emita nuevos certificados. Las opciones son:
     - SHA256 RSA
     - SHA384 RSA
     - SHA512 RSA
   - Longitud de la ruta: número de capas de confianza que la CA subordinada puede añadir al firmar nuevos certificados. Una longitud de ruta de cero (el valor predeterminado) significa que solo se pueden crear certificados y no certificados de CA. Una longitud de ruta de uno o más significa que la CA subordinada puede emitir certificados para crear más CAs subordinados a ella.
   - ARN de plantilla: muestra el ARN de la plantilla de configuración de este certificado de CA. La plantilla cambia si cambia la longitud de ruta especificada. Si crea un certificado mediante el comando issue-certificate de la CLI o la IssueCertificateacción de la API, debe especificar el ARN manualmente. Para obtener información sobre las plantillas de certificados de CA disponibles, consulte Utilice plantillas de certificados AWS Private CA.

6. Revise la configuración para comprobar que es correcta y, a continuación, seleccione Confirmar e instalar. Autoridad de certificación privada de AWS exporta una CSR, genera un certificado mediante una plantilla de certificado de CA subordinada y lo firma con la CA principal seleccionada. Autoridad de certificación privada de AWS a continuación, importa el certificado de CA subordinada firmado.

7. La página de detalles de la CA muestra el estado de la instalación (éxito o error) en la parte superior. Si la instalación se realizó correctamente, la CA subordinada recién completada muestra el estado Activo en el panel General.

Instale un certificado de CA subordinado firmado por una CA principal externa

Después de crear una CA privada subordinada como se describe enCree una CA privada en AWS Private CA, tiene la opción de activarla instalando un certificado de CA firmado por una autoridad de firma externa. Para firmar su certificado de CA subordinada con una CA externa, primero debe configurar un proveedor de servicios de confianza externo como autoridad de firma o disponer el uso de un proveedor externo.

> **Note**
>
> Los procedimientos para crear u obtener una entidad de certificación externa están fuera del alcance de esta guía.

Una vez que haya creado una CA subordinada y tenga acceso a una autoridad de firma externa, complete las siguientes tareas:

1. Obtenga una solicitud de firma de certificado (CSR) de. Autoridad de certificación privada de AWS
2. Envíe la CSR a su autoridad de firma externa y obtenga un certificado de CA firmado junto con cualquier certificado en cadena.
3. Importe el certificado y la cadena de CA Autoridad de certificación privada de AWS para activar su CA subordinada.

Para obtener procedimientos detallados, consulte Utilice certificados de CA privados firmados externamente.

Controle el acceso a la CA privada

Cualquier usuario con los permisos necesarios en una CA privada Autoridad de certificación privada de AWS puede usar esa CA para firmar otros certificados. El propietario de la CA puede emitir certificados o delegar los permisos necesarios para emitirlos a un usuario AWS Identity and Access Management (de IAM) que resida en la misma Cuenta de AWS. Un usuario que reside en una AWS cuenta diferente también puede emitir certificados si el propietario de la CA lo autoriza mediante una política basada en los recursos.

Los usuarios autorizados, ya sean de una sola cuenta o de varias cuentas, pueden usar nuestros AWS Certificate Manager recursos al Autoridad de certificación privada de AWS emitir certificados. Los certificados que se emiten desde la Autoridad de certificación privada de AWS IssueCertificateAPI o el comando de CLI issue-certificate no se administran. Estos certificados requieren una instalación manual en los dispositivos de destino y una renovación manual cuando caducan. Se administran los certificados emitidos desde la consola ACM, la RequestCertificateAPI ACM o el comando CLI request-certificate. Estos certificados se pueden instalar fácilmente en los servicios que se integran con ACM. Si el administrador de CA lo permite y la cuenta del emisor tiene una función vinculada al servicio para ACM, los certificados administrados se renuevan automáticamente cuando caducan.

Temas

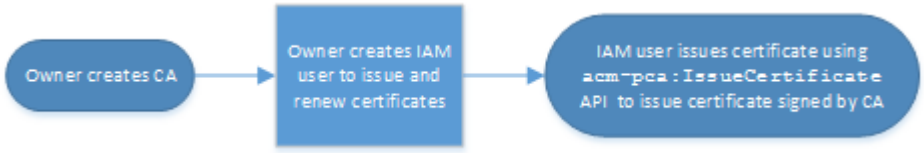
- Cree permisos de cuenta única para un usuario de IAM
- Asociar una política para el acceso entre cuentas

Cree permisos de cuenta única para un usuario de IAM

Cuando el administrador de la CA (es decir, el propietario de la CA) y el emisor del certificado residen en una sola AWS cuenta, se recomienda separar las funciones de emisor y administrador mediante la creación de un usuario AWS Identity and Access Management (de IAM) con permisos limitados. Para obtener información sobre el uso de IAM con permisos Autoridad de certificación privada de AWS, junto con ejemplos de permisos, consulte. Identity and Access Management (IAM) para AWS Private Certificate Authority

Caso 1 de cuenta única: emisión de un certificado no administrado

En este caso, el propietario de la cuenta crea una CA privada y, a continuación, crea un usuario de IAM con permiso para emitir certificados firmados por la CA privada. El usuario de IAM emite un certificado llamando a la Autoridad de certificación privada de AWS `IssueCertificate` API.

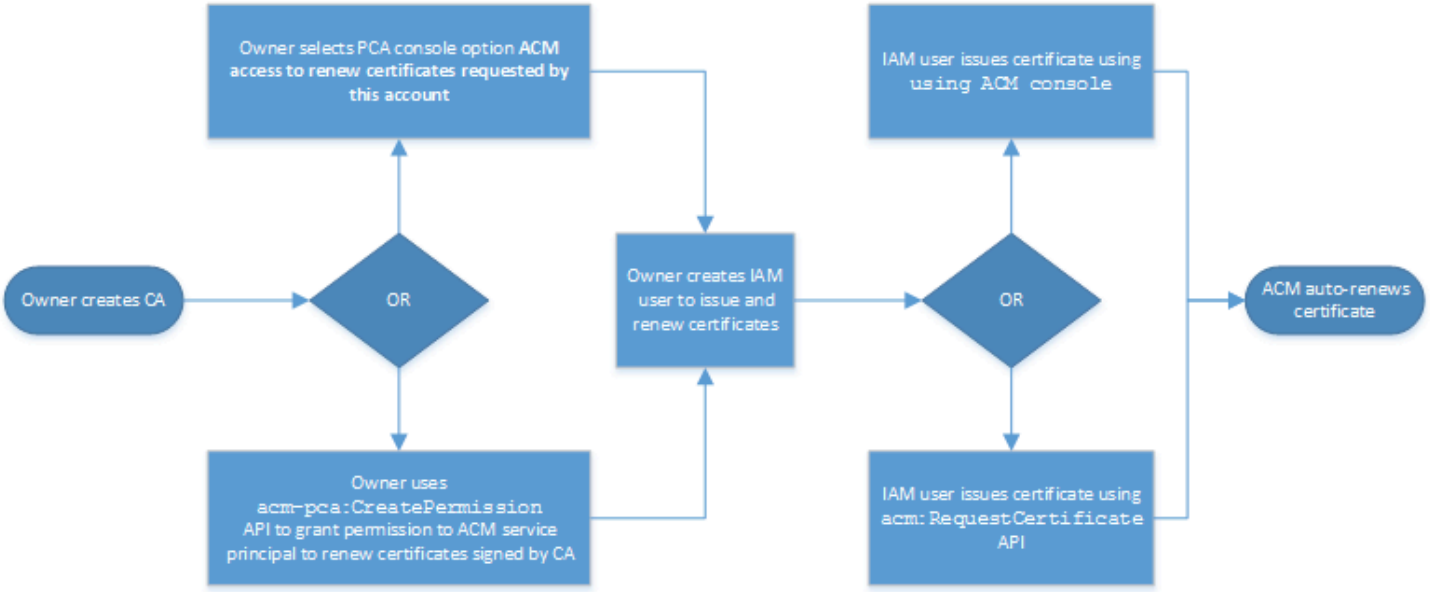


Los certificados emitidos de esta manera no se administran, lo que significa que un administrador debe exportarlos e instalarlos en los dispositivos en los que vayan a usarse. También deben renovarse manualmente cuando caduquen. La emisión de un certificado mediante esta API requiere una solicitud de firma de certificado (CSR) y un key pair generado fuera de Autoridad de certificación

privada de AWS OpenSSL o un programa similar. Para obtener más información, consulte la `IssueCertificate`documentación de https://docs.aws.amazon.com/privateca/latest/APIReference/API_IssueCertificate.html.

Caso 2 de cuenta única: emisión de un certificado gestionado a través de ACM

Este segundo caso se refiere a las operaciones de API tanto de ACM como de PCA. El propietario de la cuenta crea un usuario privado de CA e IAM como antes. A continuación, el propietario de la cuenta concede permiso a la entidad principal del servicio de ACM para renovar automáticamente todos los certificados firmados por esta CA. El usuario de IAM vuelve a emitir el certificado, pero esta vez mediante una llamada a la API `RequestCertificate` de ACM, que se encarga de la CSR y de la generación de claves. Cuando el certificado caduca, ACM automatiza el flujo de trabajo de renovación.



El propietario de la cuenta tiene la opción de conceder el permiso de renovación a través de la consola de administración durante o después de la creación de la CA o mediante la API `CreatePermission` de la PCA. Los certificados gestionados creados a partir de este flujo de trabajo están disponibles para su uso con los AWS servicios que están integrados con ACM.

La siguiente sección contiene los procedimientos para conceder permisos de renovación.

Asignación de permisos de renovación de certificados a ACM

Con la renovación administrada en AWS Certificate Manager (ACM), puede automatizar el proceso de renovación de certificados tanto en entidades de certificación públicas como privadas. Para que ACM renueve automáticamente los certificados generados por una CA privada, la propia CA

debe conceder todos los permisos posibles al a la entidad principal del servicio de ACM. Si no se conceden estos permisos de renovación para ACM, el propietario de la CA (o un representante autorizado) deben renovar manualmente cada certificado privado cuando caduca.

> **Important**
>
> Estos procedimientos para asignar permisos de renovación solo se aplican cuando el propietario de la CA y el emisor del certificado residen en la misma cuenta. AWS En el caso de un escenario entre cuentas, consulte Asociar una política para el acceso entre cuentas.

Los permisos de renovación se pueden delegar durante la creación de entidad de certificación privada o modificar en cualquier momento después, siempre que la entidad de certificación esté en el estado `ACTIVE`.

Puede administrar permisos de entidad de certificación desde la consola de Autoridad de certificación privada de AWS, la AWS Command Line Interface (AWS CLI) o la API de Autoridad de certificación privada de AWS:

Para asignar permisos de CA privada a ACM (consola)

1. Inicie sesión en su AWS cuenta y abra la Autoridad de certificación privada de AWS consola de su casahttps://console.aws.amazon.com/acm-pca/.
2. En la página Autoridad de certificación privada, elija una CA privada de la lista.
3. Elija Acciones y configure los permisos de la CA.
4. Seleccione Autorizar el acceso de ACM para renovar los certificados solicitados por esta cuenta.
5. Seleccione Save.

Para administrar los permisos de ACM en Autoridad de certificación privada de AWS ()AWS CLI

Utilice el comando create-permission para asignar permisos a ACM. Debe asignar todos los permisos posibles (`IssueCertificate`, `GetCertificate` y `ListPermissions`) para que pueda renovar automáticamente los certificados.

```
$ aws acm-pca create-permission \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --actions IssueCertificate GetCertificate ListPermissions \
```

```
    --principal acm.amazonaws.com
```

Utilice el comando list-permissions para enumerar los permisos delegados por una entidad de certificación.

```
$ aws acm-pca list-permissions \
    --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID
```

Utilice el comando delete-permission para revocar los permisos asignados por una CA a un director de servicio. AWS

```
$ aws acm-pca delete-permission \
    --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
    --principal acm.amazonaws.com
```

Asociar una política para el acceso entre cuentas

Cuando el administrador de la CA y el emisor del certificado residen en cuentas de AWS diferentes, el administrador de la CA debe compartir el acceso a la CA. Esto se logra al asociar una política basada en recursos a la CA. La política concede permisos de emisión a un director específico, que puede ser el propietario de la AWS cuenta, un usuario de IAM, un identificador o el AWS Organizations identificador de una unidad organizativa.

Un administrador de CA puede asociar y administrar políticas de las siguientes maneras:

- En la consola de administración, mediante AWS Resource Access Manager (RAM), que es un método estándar para compartir AWS recursos entre cuentas. Al compartir un recurso de CA AWS RAM con una entidad principal de otra cuenta, la política basada en los recursos requerida se adjunta automáticamente a la entidad emisora de certificados. Para obtener más información sobre la RAM, consulte la Guía del usuario de AWS RAM.

  > **Note**
  >
  > Para abrir fácilmente la consola de RAM, seleccione una CA y, a continuación, seleccione Acciones y Administrar recursos compartidos.

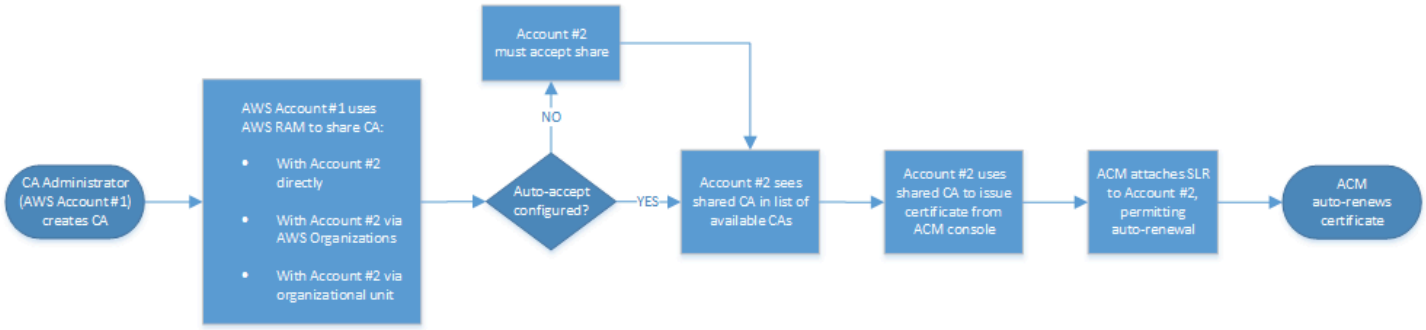
- Mediante programación, mediante la APIs PutPolicyPCA, y. GetPolicyDeletePolicy

- Use los comandos de la PCA put-policy, get-policy y delete-policy del AWS CLI de forma manual.

Solo el método de consola requiere acceso a la RAM.

Caso multicuenta 1: emisión de un certificado gestionado desde la consola

En este caso, el administrador de CA usa AWS Resource Access Manager (AWS RAM) para compartir el acceso a la CA con otra AWS cuenta, lo que permite a esa cuenta emitir certificados ACM administrados. El diagrama muestra que AWS RAM puede compartir la CA directamente con la cuenta, o indirectamente a través de un AWS Organizations ID del que la cuenta es miembro.

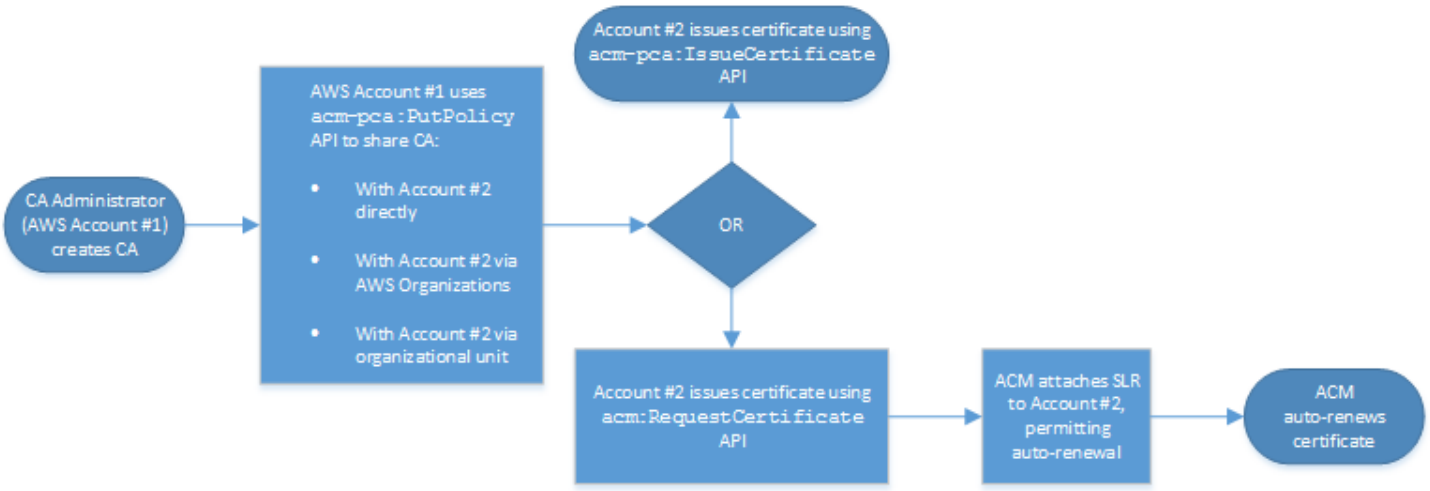


Una vez que RAM comparte un recurso AWS Organizations, el destinatario principal debe aceptar el recurso para que surta efecto. El destinatario puede configurarse AWS Organizations para aceptar automáticamente las acciones ofrecidas.

> **Note**
>
> La cuenta del destinatario es responsable de configurar la renovación automática en ACM. Por lo general, la primera vez que se utiliza una CA compartida, ACM instala una función vinculada a un servicio que le permite realizar llamadas de certificado desatendidas en Autoridad de certificación privada de AWS. Si esto no funciona (normalmente debido a la falta de un permiso), los certificados de la CA no se renuevan automáticamente. Solo el usuario de ACM puede resolver el problema, no el administrador de la CA. Para obtener más información, consulte Utilización de un rol asociado a servicios (SLR) con ACM.

Caso multicuenta 2: emisión de certificados administrados y no administrados mediante la API o la CLI

Este segundo caso demuestra las opciones de compartición y emisión que son posibles mediante la Autoridad de certificación privada de AWS API AWS Certificate Manager and. Todas estas operaciones también se pueden realizar mediante los AWS CLI comandos correspondientes.



Como las operaciones de la API se utilizan directamente en este ejemplo, el emisor del certificado puede elegir entre dos operaciones de API para emitir un certificado. La acción de la API de la PCA `IssueCertificate` da como resultado un certificado no administrado que no se renovará automáticamente y que deberá exportarse e instalarse manualmente. La acción de la API de ACM RequestCertificateda como resultado un certificado gestionado que se puede instalar fácilmente en los servicios integrados de ACM y se renueva automáticamente.

> **Note**
>
> La cuenta del destinatario es responsable de configurar la renovación automática en ACM. Por lo general, la primera vez que se utiliza una CA compartida, ACM instala una función vinculada a un servicio que le permite realizar llamadas de certificado desatendidas en Autoridad de certificación privada de AWS. Si esto no funciona (normalmente debido a la falta de un permiso), los certificados de la CA no se renovarán automáticamente y solo el usuario de ACM podrá resolver el problema, no el administrador de la CA. Para obtener más información, consulte Utilización de un rol asociado a servicios (SLR) con ACM.

Lista privada CAs

Puedes usar la Autoridad de certificación privada de AWS consola o AWS CLI publicar los anuncios privados CAs que sean de tu propiedad o a los que tengas acceso.

Para ver una lista de los disponibles CAs mediante la consola

1. Inicia sesión en tu AWS cuenta y abre la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/casa.
2. Revise la información de la lista de Entidades de certificación privadas. Puedes navegar por varias páginas CAs utilizando los números de página que aparecen en la esquina superior derecha. Cada CA ocupa una fila en la que se muestran algunas o todas las siguientes columnas:

- Asunto: información sobre el nombre distintivo de la entidad de certificación (CA).
- Id (Identificador): identificador único hexadecimal de 32 bytes del certificado
- Estado: estado de la CA. Los valores posibles son Crear, Certificado pendiente, Activo, Eliminado, Inhabilitado, Expirado y Fallado.
- Tipo: el tipo de CA. Los valores posibles son Raíz y Subordinado.
- Modo: el modo de la CA. Los valores posibles son de uso general (emite certificados que se pueden configurar con cualquier fecha de caducidad) y Certificado de corta duración (emite certificados con un período de validez máximo de siete días). Un período de validez breve puede sustituir en algunos casos a un mecanismo de revocación. El valor predeterminado es de uso general.
- Propietario: la AWS cuenta propietaria de la CA. Puede ser su cuenta o una cuenta en la que se le hayan delegado los permisos de administración de CA.
- Algoritmo de clave: el algoritmo de clave pública compatible con la CA. Los valores posibles son ML_DSA_44, ML_DSA_65, ML_DSA_87, RSA_2048, RSA_3072, RSA_4096, EC_Prime256v1, EC_SECP384R1 y EC_SECP521R1.
- Algoritmo de firma: algoritmo que utiliza la CA para firmar solicitudes de certificado. (Este parámetro no debe confundirse con el parámetro `SigningAlgorithm` utilizado para firmar certificados cuando se emiten). Los valores posibles son ML_DSA_44, ML_DSA_65, ML_DSA_87, SHA256WITHRSA, WITHRSA, SHA384SHA512WITHECDSA, SHA256WITHECDSA y SHA384WITHECDSA. SHA512

> **Note**
>
> Puede personalizar las columnas que desee mostrar, así como otras opciones de configuración, seleccionando el icono de configuración situado en la esquina superior derecha de la consola.

Para ver una lista de las disponibles CAs , utilice la AWS CLI

Utilice el list-certificate-authoritiescomando para enumerar las disponibles CAs , como se muestra en el siguiente ejemplo:

```
$ aws acm-pca list-certificate-authorities --max-items 10
```

El comando devuelve información similar a la siguiente:

```
{
   "CertificateAuthorities":[
      {
         "Arn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID",
         "CreatedAt":"2022-05-02T11:59:02.022000-07:00",
         "LastStateChangeAt":"2022-05-02T11:59:18.498000-07:00",
         "Type":"ROOT",
         "Serial":"serial_number",
         "Status":"ACTIVE",
         "NotBefore":"2022-05-02T10:59:17-07:00",
         "NotAfter":"2032-05-02T11:59:17-07:00",
         "CertificateAuthorityConfiguration":{
            "KeyAlgorithm":"RSA_2048",
            "SigningAlgorithm":"SHA256WITHRSA",
            "Subject":{
               "Organization":"testing_com"
            }
         },
         "RevocationConfiguration":{
            "CrlConfiguration":{
               "Enabled":false
            }
         }
      }
 ...
```

```
    ]
}
```

Ver una CA privada

Puede usar la consola ACM o la AWS CLI para ver los metadatos detallados sobre una CA privada y cambiar varios de los valores según sea necesario. Para obtener información detallada sobre la actualizaciónCAs, consulteActualice una CA privada en AWS Private Certificate Authority.

Para ver los detalles de la CA en la consola

1. Inicia sesión en tu AWS cuenta y abre la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/casa.
2. Revise la lista de Entidades de certificación privadas. Puedes navegar por varias páginas CAs utilizando los números de página que aparecen en la esquina superior derecha.
3. Para mostrar metadatos detallados de una CA de la lista, elija el botón de radio de la CA que desee inspeccionar. Se abrirá un panel de detalles con las siguientes vistas en pestañas:

   - Pestaña Asunto: información sobre el nombre distintivo de la entidad de certificación (CA). Para obtener más información, consulte Subject distinguished name. Estos campos incluyen:
     - Asunto: resumen de los campos de información sobre el nombre proporcionados
     - Organización (O): por ejemplo, el nombre de una empresa
     - Unidad organizativa (OU): por ejemplo, una división dentro de una empresa
     - Nombre del país (C): código de país de dos letras
     - Nombre de estado o provincia: nombre completo de un estado o una provincia
     - Nombre de la localidad: el nombre de una ciudad
     - Nombre común (CN): cadena legible por humanos para identificar la CA.
   - Pestaña Certificado de CA: información sobre la validez del certificado de CA
     - Válido hasta: fecha y hora hasta que el certificado de CA sea válido
     - Vence en: número de días hasta la caducidad del certificado
   - Pestaña de Configuración de revocación: sus selecciones actuales de opciones de revocación de certificados. Seleccione Editar para actualizar.
     - Distribución de la lista de revocación de certificados (CRL): estado activado o desactivado
     - Protocolo de estado de certificados en línea (OCSP): estado de habilitado o deshabilitado

- Pestaña Permisos: su selección actual de permisos de renovación de certificados para esta CA a través de AWS Certificate Manager (ACM). Seleccione Editar para actualizar.
- Autorización ACM para renovaciones: estado de autorización o no autorización
- Pestaña Etiquetas: su asignación actual de etiquetas personalizables para esta CA. Elija Administrar etiquetas para actualizar.
- Pestaña de recursos compartidos: su asignación actual de recursos compartidos para esta CA a través de AWS Resource Access Manager (RAM). Seleccione Administrar recursos compartidos para actualizar.
  - Nombre: el nombre del recurso compartido
  - Estado: el estado actual del recurso compartido

4. Elija el campo de ID de la CA que desee inspeccionar para abrir el panel General. El identificador único hexadecimal de 32 bytes de la CA aparece en la parte superior. El panel proporciona la siguiente información adicional:

- Estado: estado de la CA. Los valores posibles son Crear, Certificado pendiente, Activo, Eliminado, Inhabilitado, Expirado y Fallado.
- ARN: el Nombre de recurso de Amazon para la CA.
- Propietario: la AWS cuenta propietaria de la CA. Puede ser su cuenta (propia) o una cuenta en la que se le hayan delegado los permisos de administración de CA.
- Tipo de CA: el tipo de CA. Los valores posibles son Raíz y Subordinado.
- Creado en: fecha y hora en que se ha creado la CA.
- Fecha de caducidad: la fecha y la hora en la que el certificado de CA caduca.
- Modo: el modo de la CA. Los valores posibles son de uso general (certificados que se pueden configurar con cualquier fecha de caducidad) y Certificado de corta duración (emite certificados con un período de validez máximo de siete días). Un período de validez breve puede sustituir en algunos casos a un mecanismo de revocación. El valor predeterminado es de uso general.
- Algoritmo de clave: el algoritmo de clave pública compatible con la CA. Los valores posibles son ML-DSA-44, ML-DSA-65, ML-DSA-87, RSA 2048, RSA 3072, RSA 4096, ECDSA P256, ECDSA P384 y ECDSA P521.
- Algoritmo de firma: el algoritmo que la CA utiliza para firmar su propia solicitud de firma de certificado y las respuestas del OCSP (no debe confundirse con el parámetro utilizado en la API). CRLs `SigningAlgorithm` IssueCertificate Los valores posibles son ML-DSA-44, ML-

DSA-65, ML-DSA-87, RSA, RSA y RSA, ECDSA, ECDSA SHA256 SHA384 SHA512 SHA256 SHA384 SHA512

- Estándar de seguridad del almacenamiento de claves: nivel de conformidad con las normas federales de procesamiento de información (FIPS). Puede elegir entre estos valores: FIPS 140-2 de nivel 2 o superior, FIPS 140-2 de nivel 3 o superior y CCPC de nivel 1 o superior. Este parámetro varía según la región. AWS

> **Note**
>
> A partir del 26 de enero de 2023, AWS Private CA protege todas las claves privadas de CA en regiones no chinas mediante módulos de seguridad de hardware (HSMs) que cumplen con el nivel 3 de la norma FIPS PUB 140-2.

Para ver y modificar los detalles de la CA mediante el AWS CLI

Utilice el comando de describe-certificate-authority en AWS CLI para mostrar detalles sobre la CA resultante, como se muestra en el siguiente comando:

```
$ aws acm-pca describe-certificate-authority --certificate-authority-arn
 arn:aws:acm:region:account:certificate-authority/CA_ID
```

El comando devuelve información similar a la siguiente:

```
{
   "CertificateAuthority":{
      "Arn":"arn:aws:acm:region:account:certificate-authority/CA_ID",
      "CreatedAt":"2022-05-02T11:59:02.022000-07:00",
      "LastStateChangeAt":"2022-05-02T11:59:18.498000-07:00",
      "Type":"ROOT",
      "Serial":"serial_number",
      "Status":"ACTIVE",
      "NotBefore":"2022-05-02T10:59:17-07:00",
      "NotAfter":"2031-05-02T11:59:17-07:00",
      "CertificateAuthorityConfiguration":{
         "KeyAlgorithm":"RSA_2048",
         "SigningAlgorithm":"SHA256WITHRSA",
         "Subject":{
            "Organization":"testing_com"
         }
```

```
      },
      "RevocationConfiguration":{
         "CrlConfiguration":{
            "Enabled":false
         }
      }
   }
}
```

Para obtener información sobre la actualización de una CA privada desde la línea de comando, consulte Actualizar una CA (CLI).

Agregue etiquetas para su CA privada

Las etiquetas son palabras o frases que funcionan como metadatos para identificar y organizar sus recursos de AWS . Cada etiqueta consta de una clave y un valor. Puede usar la Autoridad de certificación privada de AWS consola, AWS Command Line Interface (AWS CLI) o la API de la PCA para agregar, ver o eliminar etiquetas de forma privada CAs.

Puede agregar o quitar etiquetas personalizadas para su CA privada en cualquier momento. Por ejemplo, puedes etiquetar private CAs con pares clave-valor, como `Environment=Prod` o `Environment=Beta` para identificar a qué entorno está destinada la CA. Para obtener más información, consulte Crear una entidad de certificación privada.

> **Note**
>
> Para adjuntar etiquetas a una entidad de certificación (CA) privada durante el procedimiento de creación, el administrador de una CA debe asociar primero a la acción `CreateCertificateAuthority` una política de IAM integrada y permitir el etiquetado de forma explícita. Para obtener más información, consulte Tag-on-create: Adjuntar etiquetas a una CA en el momento de su creación.

Otros AWS recursos también admiten el etiquetado. Por lo tanto, puede asignar la misma etiqueta a distintos recursos para indicar si están relacionados. Por ejemplo, puede asignar una etiqueta como `Website=example.com` a la CA, al equilibrador de carga del Elastic Load Balancing y a otros recursos relacionados. Para obtener más información sobre el etiquetado de AWS recursos, consulte Etiquetado de los recursos de Amazon EC2 en la Guía del usuario de Amazon EC2.

Las siguientes restricciones básicas se aplican a las etiquetas: Autoridad de certificación privada de AWS

- El número máximo de etiquetas por cada CA privada es 50.
- La longitud máxima de una clave de etiqueta es de 128 caracteres.
- La longitud máxima de un valor de etiqueta es de 256 caracteres.
- La clave y el valor de la etiqueta pueden contener los siguientes caracteres: A-Z, a-z y .:+= @_%- (guion).
- Las claves y los valores de las etiquetas distinguen entre mayúsculas y minúsculas.
- Los prefijos `aws:` y `rds:` están reservados para uso de AWS ; no puede agregar, editar o eliminar etiquetas cuyas claves comiencen por `aws:` o `rds:`. Etiquetas predeterminadas que comienzan por tu tags-per-resource cuota `aws:` y `rds:` no se tienen en cuenta.
- Si pretende utilizar su esquema de etiquetado en múltiples servicios y recursos, recuerde que otros servicios pueden tener restricciones diferentes de caracteres permitidos. Consulte la documentación correspondiente a dicho servicio.
- Autoridad de certificación privada de AWS las etiquetas no están disponibles para su uso en Resource Groups ni en el editor de etiquetas del Consola de administración de AWS.

Puede etiquetar una entidad de certificación privada desde la Consola de Autoridad de certificación privada de AWS, la AWS Command Line Interface (AWS CLI) o la API de Autoridad de certificación privada de AWS.

Para agregar etiquetas en una entidad de certificación privada (consola)

1. Inicia sesión en tu AWS cuenta y abre la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/casa.
2. En la página Autoridad de certificación privada, elija una CA privada de la lista.
3. En el área de detalles situada debajo de la lista, seleccione la pestaña Etiquetas. Aparecerá una lista de etiquetas existentes.
4. Elija Manage tags (Administrar etiquetas).
5. Elija Add new tag (Agregar nueva etiqueta).
6. Escriba una clave y un par de valores.
7. Seleccione Save.

Para etiquetar una entidad de certificación privada (AWS CLI)

Usa el tag-certificate-authoritycomando para añadir etiquetas a tu CA privada.

```
$ aws acm-pca tag-certificate-authority \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --tags Key=Admin,Value=Alice
```

Utilice el comando list-tags para mostrar las etiquetas de una CA privada.

```
$ aws acm-pca list-tags \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --max-results 10
```

Utilice el untag-certificate-authoritycomando para eliminar etiquetas de una CA privada.

```
$ aws acm-pca untag-certificate-authority \
     --certificate-authority-arn arn:aws:acm-pca:aregion:account:certificate-
authority/CA_ID \
     --tags Key=Purpose,Value=Website
```

Comprenda el estado de la AWS Private CA CA

El estado de una CA gestionada por medio de Autoridad de certificación privada de AWS una acción del usuario o, en algunos casos, de una acción de servicio. Por ejemplo, el estado de una CA cambia cuando caduca. Las opciones de estado disponibles para los administradores de entidades de certificación varían en función del estado actual de la entidad de certificación.

Autoridad de certificación privada de AWS puede informar de los siguientes valores de estado. La tabla muestra las capacidades de CA disponibles en cada estado.

> **Note**
>
> Para todos los valores de estado excepto `DELETED` y `FAILED`, se le facturará la CA.

Status	Emitir certificados	Valide los certificados con OCSP	Genera CRLs	Genere auditorías	Puede actualizar el certificado de CA	Los certificados se pueden revocar	Se le factura por la CA
`CREATING`: se está creando la CA.	No	No	No	No	No	No	Sí
`PENDING_CERTIFICATE` : la CA se ha creado y necesita un certificado para estar operativa.*	No	No	No	No	No	No	Sí
`ACTIVE`	Sí	Sí	Sí	Sí	Sí	Sí	Sí
`DISABLED`: ha desactivado manualmente la CA.	No	Sí	Sí	Sí	No	Sí	Sí
`EXPIRED`: el certificado de CA ha caducado.**	No	No	No	No	Sí	No	Sí
`FAILED`	La acción `CreateCertificateAuthority` falló. Esto puede ocurrir debido a una interrupción de la red, un AWS fallo del servidor u otros errores. No se puede recuperar una entidad de certificación con error. Elimine la entidad de certificación y cree una nueva.						No
`DELETED`	Su CA se encuentra dentro del período de restauración, que puede durar de 7 a 30 días. Después de este período, se elimina permanentemente. • Si llama a la API `RestoreCertificateAuthority` en una entidad de certificación con el estado `DELETED` y un						No

Status	Emitir certificados	Valide los certificados con OCSP	Genera CRLs	Genere auditorí s	Puede actualizar el certificado de CA	Los certificados se pueden revocar	Se le factura por la CA
	certificado caducado, la entidad de certificación se establecerá en `EXPIRED`. • Para obtener más información sobre la eliminación de una entidad de certificación, consulte Eliminación de una CA privada.						

Para completar la activación, debe generar una CSR, obtener un certificado de CA firmado por una CA e importar el certificado a ella. Autoridad de certificación privada de AWS La CSR se puede enviar a su nueva CA (para que la firme automáticamente) o a una CA raíz o subordinada en las instalaciones. Para obtener más información, consulte Instalación del certificado de CA.

No puede cambiar directamente el estado de una entidad de certificación caducada. Si importa un certificado nuevo para la CA, Autoridad de certificación privada de AWS restablece el estado a `ACTIVE` menos que se haya establecido `DISABLED` antes de que caducara el certificado.

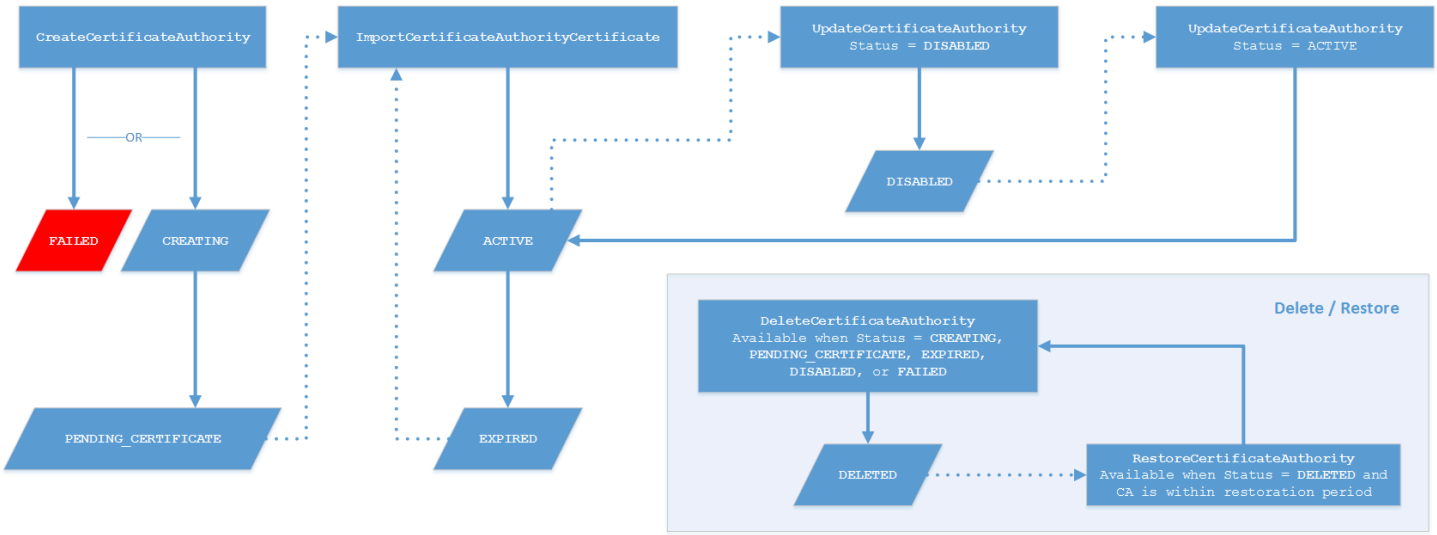
Consideraciones adicionales sobre los certificados de CA caducados:

- Los certificados de CA no se renuevan automáticamente de forma predeterminada. Para obtener información sobre cómo automatizar la renovación AWS Certificate Manager, consulte. Asignación de permisos de renovación de certificados a ACM
- Si intenta emitir un nuevo certificado con una entidad de certificación caducada, la API `IssueCertificate` devuelve `InvalidStateException`. Una entidad de certificación raíz caducada debe firmar automáticamente un nuevo certificado de entidad de certificación raíz antes de poder emitir los nuevos certificados subordinados.
- The `ListCertificateAuthorities`y `DescribeCertificateAuthority` APIs devuelva el estado de `EXPIRED` si el certificado de CA ha caducado, independientemente de si el estado de CA está establecido en `ACTIVE` o`DISABLED`. Sin embargo, si la entidad de certificación caducada se ha establecido en `DELETED`, el estado devuelto es `DELETED`.

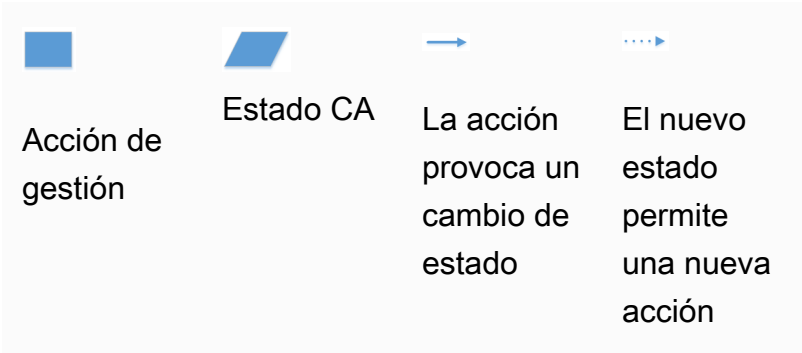
- La API `UpdateCertificateAuthority` no puede actualizar el estado de una entidad de certificación caducada.
- La API `RevokeCertificate` se puede utilizar para revocar cualquier certificado caducado, incluido un certificado de entidad de certificación.

Relación entre el estado de la CA y el ciclo de vida de la CA

El siguiente diagrama ilustra el ciclo de vida de la entidad de certificación como una interacción de las acciones de administración con el estado de la entidad de certificación.



Clave del diagrama



En la parte superior del diagrama, las acciones de administración se aplican a través de la consola, la CLI o la API de Autoridad de certificación privada de AWS . Las acciones llevan a cabo la entidad de certificación a través de la creación, activación, caducidad y renovación. El estado de la entidad de certificación cambia en respuesta (como se muestra en las líneas sólidas) a acciones manuales o

actualizaciones automatizadas. En la mayoría de los casos, un nuevo estado da lugar a una nueva acción posible (indicada por una línea de puntos) que el administrador de la entidad de certificación puede aplicar. El recuadro inferior derecho muestra los posibles valores de estado que permiten eliminar y restaurar acciones.

Actualice una CA privada en AWS Private Certificate Authority

Después de crear una entidad de certificación privada, puede actualizar su estado o la configuración de revocación. En este tema se proporcionan detalles sobre el estado y el ciclo de vida de la CA, junto con ejemplos de actualizaciones de la consola y la CLI CAs.

Actualizar una CA (consola)

Los siguientes procedimientos muestran cómo actualizar configuraciones de CA existentes utilizando Consola de administración de AWS.

Actualizar el estado de CA (consola)

En este ejemplo, el estado de una CA habilitada cambia a inhabilitado.

Para actualizar el estado de una CA

1. Inicia sesión en tu AWS cuenta y abre la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/casa
2. En la página Autoridad de certificación privada, elija una CA privada que esté actualmente activa de la lista.
3. En el menú Acciones, seleccione Desactivar para deshabilitar la CA privada.

Actualizar la configuración de revocación de una CA (consola)

Puede actualizar la configuración de revocación de su CA privada, por ejemplo, añadiendo o quitando la compatibilidad con OCSP o CRL, o modificando su configuración.

> **Note**
>
> Los cambios en la configuración de revocación de una CA no afectan a los certificados que ya se emitieron. Para que la revocación gestionada funcione, los certificados antiguos deben volver a emitirse.

Para OCSP, puede cambiar las opciones siguientes:

- Habilitación y desactivación de OCSP.
- Habilite o deshabilite un nombre de dominio completo OCSP personalizado (FQDN).
- Cambie el FQDN.

Para una CRL, puede elegir una de las siguientes configuraciones:

- El tipo de CRL (completa o particionada)
- Si la entidad de certificación privada genera una lista de revocación de certificados (CRL)
- El número de días para que caduque una CRL. Tenga en cuenta que Autoridad de certificación privada de AWS comienza a intentar regenerar la CRL a la mitad del número de días que especifique.
- El nombre del bucket de Amazon S3 donde se guarda la CRL.
- Una alias para que el nombre del bucket de Amazon S3 no esté visible públicamente.

> **Important**
>
> Cambiar cualquiera de estos parámetros pueden tener consecuencias negativas. Algunos ejemplos incluyen deshabilitar la generación de CRL, cambiar el período de validez o cambiar el bucket de S3 después de poner su CA privada en producción. Estos cambios pueden infringir los certificados existentes que dependen de la CRL y de la configuración de la CRL actual. El alias se puede cambiar sin problema siempre que el alias anterior siga asociado al bucket correcto.

Para actualizar la configuración de revocación

1. Inicia sesión en tu AWS cuenta y abre la Autoridad de certificación privada de AWS consola en casahttps://console.aws.amazon.com/acm-pca/.
2. En la página Autoridad de certificación privada, elija una CA privada de la lista. Esto abre el panel de detalles de la CA.
3. Seleccione la pestaña Configuración de revocación y, a continuación, elija Editar.
4. En las Opciones de revocación de certificados, se muestran dos opciones:

- Activar la distribución de CRL
- Encender OCSP

Puede configurar uno de estos mecanismos de revocación, ninguno o ambos para su CA. Aunque es opcional, se recomienda la revocación gestionada como práctica recomendada. Antes de completar este paso, consulte Planifique el método de revocación AWS Private CA de su certificado para obtener información sobre las ventajas de cada método, la configuración preliminar que podría ser necesaria y las características de revocación adicionales.

Para configurar una CRL

1. Selección Activar la distribución de CRL.
2. Si desea crear un bucket de Amazon S3 para las entradas de CRL, seleccione Crear un nuevo bucket de S3. Proporcione un nombre de bucket único. (No es necesario incluir la ruta de acceso al bucket). De lo contrario, deje esta opción sin seleccionar y elija un bucket existente de la lista de nombres de buckets de S3.

   Si creas un depósito nuevo, Autoridad de certificación privada de AWS crea y adjunta la política de acceso requerida. Si decide utilizar un depósito existente, debe adjuntarle una política de acceso antes de empezar a CRLs generarlo. Utilice uno de los patrones de políticas descritos en Políticas de acceso para CRLs Amazon S3. Para obtener más información sobre cómo adjuntar un política, consulte Agregar una política de bucket mediante la consola de Amazon S3.

   > **Note**
   >
   > Al utilizar la Autoridad de certificación privada de AWS consola, se produce un error al intentar crear una CA si se cumplen las dos condiciones siguientes:
   >
   > - Está aplicando la configuración de bloqueo de acceso público en su cuenta o bucket de Amazon S3.
   > - Ha solicitado Autoridad de certificación privada de AWS crear un bucket de Amazon S3 automáticamente.
   >
   > En esta situación, la consola intenta, de forma predeterminada, crear un bucket de acceso público y Amazon S3 rechaza esta acción. Compruebe su configuración de

Amazon S3 si esto ocurre. Para obtener más información, consulte Bloqueo del acceso público al almacenamiento de Amazon S3.

3. Expanda Avanzado para obtener opciones de configuración adicionales.

   - Seleccione Habilitar la partición para habilitar la partición de. CRLs Si no habilita la partición, su CA estará sujeta al número máximo de certificados revocados que se indica en las cuotas.AWS Private Certificate Authority Para obtener más información sobre las particiones CRLs, consulta los tipos de CRL.
   - Agregue un Nombre de CRL personalizado para crear un alias para el bucket de Amazon S3. Este nombre se incluye en los certificados emitidos por la entidad de certificación (CA) en la extensión “Puntos de distribución de CRL” definida por RFC 5280. Para usar CRLs over IPv6, configúrelo en el punto final S3 de doble pila de su bucket, tal y como se describe en Using over. CRLs IPv6
   - Añada una ruta personalizada para crear un alias de DNS para la ruta del archivo en su bucket de Amazon S3.
   - Introduzca la validez en días en los que su CRL seguirá siendo válida. El valor predeterminado es 7 días. En el caso de Internet CRLs, es habitual que el período de validez sea de 2 a 7 días. Autoridad de certificación privada de AWS intenta regenerar la CRL en el punto medio del período especificado.

4. Cuando haya terminado, elija Guardar cambios.

Para configurar OCSP

1. En la página Revocación de certificados, elija Activar OCSP.
2. En el campo Punto de conexión de OCSP personalizado (opcional), puede proporcionar un nombre de dominio completo (FQDN) para un punto de conexión de OCSP. Para usar OCSP over IPv6, defina este campo en un punto final de doble pila, tal y como se describe en Using OCSP over. IPv6

   Al proporcionar un FQDN en este campo, Autoridad de certificación privada de AWS inserta el FQDN en la extensión Authority Information Access de cada certificado emitido, en lugar de la URL predeterminada del respondedor OCSP. AWS Cuando un punto de conexión recibe un certificado que contiene el FQDN personalizado, consulte esa dirección para obtener una respuesta del OCSP. Para que este mecanismo funcione, debe realizar dos acciones adicionales:

- Utilice un servidor proxy para reenviar el tráfico que llegue a su FQDN personalizado al respondedor de OCSP. AWS
- Agregue el registro CNAME correspondiente a su base de datos DNS.

> **Tip**
>
> Para obtener más información sobre la implementación de una solución OCSP completa mediante un CNAME personalizado, consulte Personaliza la URL de OCSP para AWS Private CA.

Por ejemplo, este es un registro CNAME para un OCSP personalizado tal como aparecería en Amazon Route 53.

Nombre del registro	Tipo	Política de direccionamiento	Diferenciador	Valor/ruta de destino del tráfico
alternative.example.com	CNAME	Sencillez	-	proxy.example.com

> **Note**
>
> El valor de CNAME no debe incluir un prefijo de protocolo como “http://” o “https://”.

3. Cuando haya terminado, elija Guardar cambios.

Actualizar una CA (CLI)

Los siguientes procedimientos muestran cómo actualizar el estado y la configuración de revocación de una CA existente utilizando AWS CLI.

> **Note**
>
> Los cambios en la configuración de revocación de una CA no afectan a los certificados que ya se emitieron. Para que la revocación gestionada funcione, los certificados antiguos deben volver a emitirse.

Para actualizar el estado de la CA (AWS CLI)

Utilice el comando update-certificate-authority.

Esto resulta útil cuando tiene una CA existente con un estado `DISABLED` que desea configurar `ACTIVE`. Para empezar, confirme el estado inicial de la CA con el siguiente comando.

```
$ aws acm-pca describe-certificate-authority \
      --certificate-authority-arn "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566" \
      --output json
```

Esto devuelve un resultado similar a lo siguiente.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-05T14:24:12.867000-08:00",
        "LastStateChangeAt": "2021-03-08T13:17:40.221000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "DISABLED",
        "NotBefore": "2021-03-08T07:46:27-08:00",
        "NotAfter": "2022-03-08T08:46:27-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
```

```
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "CustomCname": "alternative.example.com",
                "S3BucketName": "amzn-s3-demo-bucket"
   },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

El siguiente comando establece el estado de la CA privada en `ACTIVE`. Esto solo es posible si hay un certificado válido instalado en la CA.

```
$ aws acm-pca update-certificate-authority \
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
      --status "ACTIVE"
```

Inspeccione el nuevo estado de la CA.

```
$ aws acm-pca describe-certificate-authority \
      --certificate-authority-arn "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566" \
      --output json
```

El estado ahora aparece como `ACTIVE`.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-05T14:24:12.867000-08:00",
        "LastStateChangeAt": "2021-03-08T13:23:09.352000-08:00",
        "Type": "ROOT",
```

```
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T07:46:27-08:00",
        "NotAfter": "2022-03-08T08:46:27-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "CustomCname": "alternative.example.com",
                "S3BucketName": "amzn-s3-demo-bucket"
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

En algunos casos, es posible que tenga una CA activa sin un mecanismo de revocación configurado. Si quiere empezar a utilizar una lista de revocación de certificados (CRL), utilice el siguiente procedimiento.

Para agregar una CRL a una CA existente (AWS CLI)

1. Puede utilizar el siguiente comando para encontrar el estado actual de la CA.

```
$ aws acm-pca describe-certificate-authority
 --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
 --output json
```

El resultado confirma que la CA tiene un estado `ACTIVE`, pero no está configurada para usar una CRL.


```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-08T14:36:26.449000-08:00",
        "LastStateChangeAt": "2021-03-08T14:50:52.224000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T13:46:50-08:00",
        "NotAfter": "2022-03-08T14:46:50-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": false
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```


2. Cree y guarde un archivo con un nombre `revoke_config.txt` para definir los parámetros de configuración de la CRL.

```
{
   "CrlConfiguration":{
```

```
      "Enabled": true,
      "ExpirationInDays": 7,
      "S3BucketName": "amzn-s3-demo-bucket"
   }
}
```

> **Note**
>
> Al actualizar una CA de atestación de dispositivos Matter para habilitarla CRLs, debe configurarla para que omita la extensión CDP de los certificados emitidos para cumplir con el estándar Matter actual. Para ello, defina los parámetros de configuración de la CRL tal y como se muestra a continuación:
>
> ```
> {
>    "CrlConfiguration":{
>       "Enabled": true,
>       "ExpirationInDays": 7,
>       "S3BucketName": "amzn-s3-demo-bucket"
>       "CrlDistributionPointExtensionConfiguration":{
>          "OmitExtension": true
>       }
>    }
> }
> ```

3. Utilice el update-certificate-authoritycomando y el archivo de configuración de revocación para actualizar la CA.

```
$ aws acm-pca update-certificate-authority \
      --certificate-authority-arn arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566 \
      --revocation-configuration file://revoke_config.txt
```

4. Inspeccione nuevamente el estado de la CA.

```
$ aws acm-pca describe-certificate-authority
 --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
 --output json
```

El resultado confirma que la CA está ahora configurada para usar una CRL.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-08T14:36:26.449000-08:00",
        "LastStateChangeAt": "2021-03-08T14:50:52.224000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T13:46:50-08:00",
        "NotAfter": "2022-03-08T14:46:50-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "S3BucketName": "amzn-s3-demo-bucket",
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

En algunos casos, es posible que desee agregar soporte de revocación de OCSP en lugar de habilitar una CRL tal y como se hizo en el procedimiento anterior. En ese caso, siga los siguientes pasos.

Para añadir compatibilidad con OCSP a una CA existente (AWS CLI)

1. Cree y guarde un archivo con un nombre; por ejemplo, `revoke_config.txt`, para definir los parámetros de OCSP.

```
{
   "OcspConfiguration":{
      "Enabled":true
   }
}
```

2. Utilice el update-certificate-authoritycomando y el archivo de configuración de revocación para actualizar la CA.

```
$ aws acm-pca update-certificate-authority \
      --certificate-authority-arn arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566 \
      --revocation-configuration file://revoke_config.txt
```

3. Inspeccione nuevamente el estado de la CA.

```
$ aws acm-pca describe-certificate-authority
 --certificate-authority-arnarn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
 --output json
```

El resultado confirma que la CA está ahora configurada para usar un OCSP.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-08T14:36:26.449000-08:00",
        "LastStateChangeAt": "2021-03-08T14:50:52.224000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T13:46:50-08:00",
        "NotAfter": "2022-03-08T14:46:50-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
```

```
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": false
            },
            "OcspConfiguration": {
                "Enabled": true
            }
        }
    }
}
```

> **Note**
>
> También puede configurar el soporte de CRL y OCSP en una CA.

Eliminación de una CA privada

Puede eliminar una CA privada de la Consola de administración de AWS o de AWS CLI forma permanente. Es posible que desee hacer esto, por ejemplo, para sustituirla por una nueva entidad de certificación que tenga otra clave privada. Para eliminar una entidad de certificación de forma segura, siga estos pasos:

1. Cree la CA de sustitución.
2. Una vez que la nueva CA privada esté en producción, deshabilite la antigua, pero no la elimine inmediatamente.
3. Manténgala deshabilitada hasta que caduquen todos los certificados que haya emitido.
4. Elimine la CA antigua.

Autoridad de certificación privada de AWS no comprueba que todos los certificados emitidos hayan caducado antes de procesar una solicitud de eliminación. Puede generar un informe de auditoría para determinar qué certificados han caducado. Mientras la CA esté deshabilitada, podrá revocar los certificados, pero no emitir otros nuevos.

Si tiene que eliminar una CA privada antes de que todos los certificados que ha emitido hayan caducado, le recomendamos que revoque también el certificado de la CA. El certificado de la CA aparecerá en la CRL de la CA principal y los clientes dejarán de confiar en la CA privada.

> **Important**
>
> Una entidad de certificación privada se puede eliminar si tiene el estado `PENDING_CERTIFICATE`, `CREATING`, `EXPIRED`, `DISABLED` o `FAILED`. Para poder eliminar una entidad de certificación con el estado `ACTIVE`, primero debe desactivarla o la solicitud de eliminación generará una excepción. Si va a eliminar una CA privada que tiene el estado `PENDING_CERTIFICATE` o `DISABLED`, puede establecer la duración del periodo de restauración entre 7 y 30 días, donde el valor predeterminado es de 30 días. Durante este período, el estado se establece en `DELETED` y la entidad de certificación se puede restaurar. Una entidad de certificación (CA) privada que se elimina mientras se encuentra en el estado `CREATING` o `FAILED` no tiene un período de restauración asignado y no se puede restaurar. Para obtener más información, consulte Restaure una CA privada.
>
> Una vez que se elimina una CA privada, no se le aplicarán cargos por ella. Sin embargo, si la CA eliminada se restaura, se le cobrará por el tiempo que ha pasado entre su eliminación y su restauración. Para obtener más información, consulte Precios para AWS Private Certificate Authority.

Para eliminar una entidad de certificación privada (consola)

1. Inicia sesión en tu AWS cuenta y abre la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/casa.
2. En la página Autoridad de certificación privada, elija una CA privada de la lista.
3. Si la CA tiene el estado `ACTIVE`, debe deshabilitarla. En el menú Actions (Acciones), seleccione Disable (Deshabilitar). Cuando se le solicite, elija Comprendo el riesgo, continúe.
4. En el caso de una CA que no se encuentre en el estado `ACTIVE`, elija Acciones, Eliminar.
5. Si su CA se encuentra en el estado `DISABLED`, `EXPIRED` o `PENDING_CERTIFICATE`, la página Eliminar CA le permite especificar un período de restauración de 7 a 30 días. Si la CA privada no

se encuentra en ninguno de estos estados, no podrá restaurarse más adelante y la eliminación será permanente.

6. Elija Eliminar.
7. Si está seguro de que desea eliminar la CA privada, seleccione Permanently delete (Eliminar permanentemente) cuando se lo pregunten. El estado de la CA privada cambiará a `DELETED`. Sin embargo, podrá restaurarla antes de que finalice el periodo de restauración. Para comprobar el período de restauración de una entidad emisora de certificados privada en el `DELETED` estado, llama a la operación DescribeCerticateAuthorityo ListCertificateAuthoritiesAPI.

Para eliminar una entidad de certificación privada (AWS CLI)

Utilice el delete-certificate-authoritycomando para eliminar una CA privada.

```
$ aws acm-pca delete-certificate-authority \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --permanent-deletion-time-in-days 16
```

Restaure una CA privada

Puede restaurar una CA privada que haya eliminado siempre que esté dentro del periodo de restauración que especificó tras eliminarla. El período de restauración es de 7 a 30 días. Al final de ese período, la CA se elimina definitivamente. Para obtener más información, consulte Eliminación de una CA privada. No puede restaurar una CA privada que se haya eliminado definitivamente.

> **Note**
>
> Una vez que se elimina una CA privada, no se le aplicarán cargos por ella. Sin embargo, si la CA eliminada se restaura, se le cobrará por el tiempo que ha pasado entre su eliminación y su restauración. Para obtener más información, consulte Precios para AWS Private Certificate Authority.

Restauración de una CA privada (consola)

Puede utilizarla Consola de administración de AWS para restaurar una CA privada.

Para restaurar una CA privada (consola)

1. Inicia sesión en tu AWS cuenta y abre la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/casa.
2. En la página Entidad de certificación privada, elija una CA privada eliminada de la lista.
3. En el menú Actions (Acciones), seleccione Restore (Restaurar).
4. En la página Restaurar CA, vuelva a seleccionar Restaurar.
5. Si la operación se realiza correctamente, el estado de la CA privada se establecerá en el que tenía antes de su eliminación. Seleccione Acciones, Habilitar y Habilitar de nuevo para cambiar su estado a `ACTIVE`. Si la entidad de certificación privada tiene el estado `PENDING_CERTIFICATE` en el momento de la eliminación, debe importar un certificado de entidad de certificación en la entidad de certificación privada para poder activarla.

Restaure una CA privada (AWS CLI)

Utilice el restore-certificate-authoritycomando para restaurar una CA privada eliminada que se encuentre en el `DELETED` estado. En los pasos siguientes, se explica todo el proceso necesario para eliminar, restaurar y volver a activar una CA privada.

Para eliminar, restaurar y volver a activar una CA privada (AWS CLI)

1. Elimine la CA privada.

   Ejecute el delete-certificate-authoritycomando para eliminar la CA privada. Si el estado de la CA privada es `DISABLED` o `PENDING_CERTIFICATE`, puede establecer el parámetro `--permanent-deletion-time-in-days` para especificar el periodo de restauración de la CA privada entre 7 y 30 días. Si no especifica ningún periodo de restauración, el valor predeterminado es de 30 días. Si la operación se realiza correctamente, este comando establece el estado de la CA privada en `DELETED`.

   > **Note**
   >
   > Para que una CA privada puede restaurarse, su estado en el momento de realizar la eliminación debe ser `DISABLED` o `PENDING_CERTIFICATE`.

```
$ aws acm-pca delete-certificate-authority \
```

```
    --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
    --permanent-deletion-time-in-days 16
```

2. Restaure la CA privada.

   Ejecute el restore-certificate-authoritycomando para restaurar la CA privada. Debe ejecutar el comando antes de que expire el periodo de restauración establecido con el comando delete-certificate-authority. Si la operación se realiza correctamente, el comando establece el estado de la CA privada en el que tenía antes de su eliminación.

```
$ aws acm-pca restore-certificate-authority \
    --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID
```

3. Establezca la CA privada en el estado `ACTIVE`.

   Ejecute el update-certificate-authoritycomando para cambiar el estado de la CA privada a`ACTIVE`.

```
$ aws acm-pca update-certificate-authority \
    --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
    --status ACTIVE
```

Utilice certificados de CA privados firmados externamente

Si la raíz de confianza de su jerarquía de CA privada debe ser una CA ajena a ella Autoridad de certificación privada de AWS, puede crear y autofirmar su propia CA raíz. Alternativamente, puede obtener un certificado de entidad de certificación privada firmado por una entidad de certificación privada externa operada por su organización. Sea cual sea su origen, puede utilizar esta CA obtenida externamente para firmar un certificado de CA subordinada privada que Autoridad de certificación privada de AWS gestione.

> **Note**
>
> Los procedimientos para crear u obtener una entidad de certificación externa están fuera del alcance de esta guía.

El uso de una CA principal externa Autoridad de certificación privada de AWS permite aplicar las restricciones de nombres de las CA, tal como se definen en la sección Restricciones de nombres de la RFC 5280. Las restricciones proporcionan una forma para que los administradores de entidades de certificación restrinjan los nombres de asunto en los certificados.

Si tiene previsto firmar un certificado de CA subordinada privada con una CA externa, debe realizar tres tareas antes de tener una CA en funcionamiento en Autoridad de certificación privada de AWS:

1. Genere una solicitud de firma de certificado (CSR).
2. Envíe la CSR a su autoridad de firma externa y obtenga un certificado firmado y un certificado en cadena.
3. Instale un certificado firmado en. Autoridad de certificación privada de AWS

Los siguientes procedimientos describen cómo completar estas tareas utilizando el Consola de administración de AWS o el AWS CLI.

Para obtener e instalar un certificado de CA firmado externamente (consola)

1. (Opcional) Si aún no está en la página de detalles de la CA, abra la Autoridad de certificación privada de AWS consola de https://console.aws.amazon.com/acm-pca/su casa. En la página de autoridades de certificación privadas, elija una CA raíz con el estado Certificado pendiente, Activo, Deshabilitado o Caducado.
2. Seleccione Acciones, Instalar el certificado de CA para abrir la página Instalación del certificado de CA subordinado.
3. En la página Instalar un certificado de CA subordinado, en Seleccionar el tipo de CA, elija CA privada externa.
4. En CSR para esta CA, la consola muestra el texto ASCII de la CSR codificado en Base64. Puede copiar el texto con el botón Copiar o puede seleccionar Exportar CSR a un archivo y guardarlo localmente.

   > **Note**
   >
   > Se debe conservar el formato exacto del texto de la CSR al copiar y pegar.

5. Si no puede realizar inmediatamente los pasos necesarios sin conexión para obtener un certificado firmado por la autoridad firmante externa, puede cerrar la página y volver a ella una vez que disponga de un certificado firmado y de una cadena de certificados.

De lo contrario, si está preparado, realice una de las siguientes acciones:

- Pegue el texto ASCII codificado en Base64 del cuerpo del certificado y de la cadena de certificados en sus respectivos cuadros de texto.
- Seleccione Cargar para cargar el cuerpo del certificado y la cadena de certificados de los archivos locales en sus cuadros de texto respectivos.

6. Seleccione Confirmar e instalar.

Para obtener e instalar un certificado de CA firmado externamente (CLI)

1. Utilice el get-certificate-authority-csrcomando para recuperar la solicitud de firma del certificado (CSR) de su CA privada. Si desea enviar la CSR a su pantalla, utilice la `--output text` opción para eliminar los CR/LF caracteres del final de cada línea. Para enviar la CSR a un archivo, utilice la opción de redirección (>) seguida del nombre de archivo.

```
$ aws acm-pca get-certificate-authority-csr \
--certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566 \
--output text
```

Tras guardar una CSR como archivo local, puede inspeccionarla mediante el siguiente comando de OpenSSL:

```
openssl req -in path_to_CSR_file -text -noout
```

Este comando genera un resultado similar al siguiente. Observe que la extensión de entidad de certificación es TRUE, lo que indica que la CSR es para un certificado de entidad de certificación.

```
Certificate Request:
Data:
Version: 0 (0x0)
Subject: O=ExampleCompany, OU=Corporate Office, CN=Example CA 1
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
        Modulus:
            00:d4:23:51:b3:dd:01:09:01:0b:4c:59:e4:ea:81:
```

```
            1d:7f:48:36:ef:2a:e9:45:82:ec:95:1d:c6:d7:c9:
            7f:19:06:73:c5:cd:63:43:14:eb:c8:03:82:f8:7b:
            c7:89:e6:8d:03:eb:b6:76:58:70:f2:cb:c3:4c:67:
            ea:50:fd:b9:17:84:b8:60:2c:64:9d:2e:d5:7d:da:
            46:56:38:34:a9:0d:57:77:85:f1:6f:b8:ce:73:eb:
            f7:62:a7:8e:e6:35:f5:df:0c:f7:3b:f5:7f:bd:f4:
            38:0b:95:50:2c:be:7d:bf:d9:ad:91:c3:81:29:23:
            b2:5e:a6:83:79:53:f3:06:12:20:7e:a8:fa:18:d6:
            a8:f3:a3:89:a5:a3:6a:76:da:d0:97:e5:13:bc:84:
            a6:5c:d6:54:1a:f0:80:16:dd:4e:79:7b:ff:6d:39:
            b5:67:56:cb:02:6b:14:c3:17:06:0e:7d:fb:d2:7e:
            1c:b8:7d:1d:83:13:59:b2:76:75:5e:d1:e3:23:6d:
            8a:5e:f5:85:ca:d7:e9:a3:f1:9b:42:9f:ed:8a:3c:
            14:4d:1f:fc:95:2b:51:6c:de:8f:ee:02:8c:0c:b6:
            3e:2d:68:e5:f8:86:3f:4f:52:ec:a6:f0:01:c4:7d:
            68:f3:09:ae:b9:97:d6:fc:e4:de:58:58:37:09:9a:
            f6:27
        Exponent: 65537 (0x10001)
Attributes:
Requested Extensions:
    X509v3 Basic Constraints:
        CA:TRUE
Signature Algorithm: sha256WithRSAEncryption
 c5:64:0e:6c:cf:11:03:0b:b7:b8:9e:48:e1:04:45:a0:7f:cc:
 a7:fd:e9:4d:c9:00:26:c5:6e:d0:7e:69:7a:fb:17:1f:f3:5d:
 ac:f3:65:0a:96:5a:47:3c:c1:ee:45:84:46:e3:e6:05:73:0c:
 ce:c9:a0:5e:af:55:bb:89:46:21:92:7b:10:96:92:1b:e6:75:
 de:02:13:2d:98:72:47:bd:b1:13:1a:3d:bb:71:ae:62:86:1a:
 ee:ae:4e:f4:29:2e:d6:fc:70:06:ac:ca:cf:bb:ee:63:68:14:
 8e:b2:8f:e3:8d:e8:8f:e0:33:74:d6:cf:e2:e9:41:ad:b6:47:
 f8:2e:7d:0a:82:af:c6:d8:53:c2:88:a0:32:05:09:e0:04:8f:
 79:1c:ac:0d:d4:77:8e:a6:b2:5f:07:f8:1b:e3:98:d4:12:3d:
 28:32:82:b5:50:92:a4:b2:4c:28:fc:d2:73:75:75:ff:10:33:
 2c:c0:67:4b:de:fd:e6:69:1c:a8:bb:e8:31:93:07:35:69:b7:
 d6:53:37:53:d5:07:dd:54:35:74:50:50:f9:99:7d:38:b7:b6:
 7f:bd:6c:b8:e4:2a:38:e5:04:00:a8:a3:d9:e5:06:38:e0:38:
 4c:ca:a9:3c:37:6d:ba:58:38:11:9c:30:08:93:a5:62:00:18:
 d1:83:66:40
```

2. Envíe la CSR a su autoridad de firma externa y obtenga los archivos que contienen el certificado firmado codificado en Base64 con formato PEM y la cadena del certificado.
3. Utilice el import-certificate-authority-certificatecomando para importar el archivo de certificado de CA privado y el archivo en Autoridad de certificación privada de AWS cadena.

```
$ aws acm-pca import-certificate-authority-certificate \
--certificate-authority-arn arn:aws:acm-pca:region:account:\
certificate-authority/12345678-1234-1234-1234-123456789012 \
--certificate file://C:\example_ca_cert.pem \
--certificate-chain file://C:\example_ca_cert_chain.pem
```

Emita y administre certificados en AWS Private CA

Una vez que haya creado y activado una entidad emisora de certificados (CA) privada y haya configurado el acceso a ella, usted o sus usuarios autorizados podrán emitir y gestionar los certificados. Si aún no ha configurado políticas AWS Identity and Access Management (IAM) para la CA, puede obtener más información sobre cómo configurarlas en la sección Identity and Access Management de esta guía. Para obtener información sobre la configuración del acceso de CA en escenarios entre cuentas y en una sola cuenta, consulte Controle el acceso a la CA privada.

Temas

- Emita certificados privados de entidad final
- Recupera un certificado privado
- Enumere los certificados privados
- Exporte un certificado privado y su clave secreta
- Revocar un certificado privado
- Automatice la exportación de un certificado renovado
- Utilice plantillas de certificados AWS Private CA

Emita certificados privados de entidad final

Con una CA privada, puede solicitar certificados de entidad final privada a AWS Certificate Manager (ACM) o. Autoridad de certificación privada de AWS Las capacidades de ambos servicios se comparan en la siguiente tabla.

Funcionalidad	ACM	Autoridad de certificación privada de AWS
Emitir un certificado de entidad final	✓ (utilizando RequestCertificate o la consola)	✓ (utilizando IssueCertificate)
Asociación con balanceadores de carga y servicios conectados a Internet AWS	✓	No compatible

Funcionalidad	ACM	Autoridad de certificación privada de AWS
Renovación administrada de certificados	✓	Con compatibilidad indirecta a través de ACM
Soporte de consola	✓	No compatible
Compatibilidad con API	✓	✓
Compatibilidad con CLI	✓	✓

Cuando Autoridad de certificación privada de AWS crea un certificado, sigue una plantilla que especifica el tipo de certificado y la longitud de la ruta. Si no se proporciona ningún ARN de plantilla a la instrucción API o CLI que crea el certificado, se aplica la plantilla EndEntityCertificate/V1 de forma predeterminada. Para obtener más información acerca de las plantillas de certificados disponibles, consulte Utilice plantillas de certificados AWS Private CA.

Si bien los certificados ACM se diseñan en función de la confianza pública, satisfacen Autoridad de certificación privada de AWS las necesidades de su PKI privada. En consecuencia, puede configurar los certificados mediante la Autoridad de certificación privada de AWS API y la CLI de formas no permitidas por ACM. Estos incluyen los siguientes:

- Cree un certificado con cualquier nombre de sujeto.
- Utilizar cualquier algoritmo de clave privada y cualquier longitud de clave que sean compatibles.
- Mediante cualquiera de los algoritmos de firma compatibles.
- Especificar cualquier periodo de validez para la CA privada y los certificados privados.

Tras crear un certificado TLS privado con él Autoridad de certificación privada de AWS, puede importarlo a ACM y utilizarlo con un servicio compatible. AWS

> **Note**
>
> Los certificados creados con el siguiente procedimiento, mediante el issue-certificate comando o con la acción de la IssueCertificateAPI, no se pueden exportar directamente para su uso externo. AWS Sin embargo, puede usar su CA privada para firmar los certificados emitidos a través de ACM y esos certificados se pueden exportar junto con sus claves

secretas. Para obtener más información, consulte Solicitar un certificado privado y Exportar un certificado privado en la Guía del usuario de ACM.

Emita un certificado estándar (AWS CLI)

Puede usar el comando issue-certificate de la Autoridad de certificación privada de AWS CLI o la acción de la API IssueCertificatepara solicitar un certificado de entidad final. Este comando necesita el nombre de recurso de Amazon (ARN) de la CA privada que desea utilizar para emitir el certificado. También debe generar una solicitud de firma de certificado (CSR) mediante un programa como OpenSSL.

Si usa la Autoridad de certificación privada de AWS API o AWS CLI emite un certificado privado, el certificado no se administra, lo que significa que no puede usar la consola ACM, la CLI de ACM o la API de ACM para verlo o exportarlo, y el certificado no se renueva automáticamente. Sin embargo, puede usar el comando PCA get-certificate para recuperar los detalles del certificado y, si es propietario de la CA, puede crear un informe de auditoría.

Consideraciones sobre la creación de certificados

- En conformidad con RFC 5280, la longitud del nombre de dominio (técnicamente, el nombre común) que proporcione no puede superar los 64 octetos (caracteres), incluidos los puntos. Para agregar un nombre de dominio más largo, especifíquelo en el campo Nombre alternativo del asunto, que admite nombres de hasta 253 octetos de longitud.
- Si utiliza la AWS CLI versión 1.6.3 o posterior, utilice el prefijo `fileb://` al especificar los archivos de entrada codificados en base64, como. CSRs Esto garantiza que se analizan los datos correctamente Autoridad de certificación privada de AWS .

El siguiente comando de OpenSSL genera una CSR y una clave privada para un certificado:

```
$ openssl req -out csr.pem -new -newkey rsa:2048 -nodes -keyout private-key.pem
```

Puede inspeccionar el contenido de la CSR de la siguiente manera:

```
$ openssl req -in csr.pem -text -noout
```

El resultado debe parecerse al siguiente ejemplo abreviado:

```
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=US, O=Big Org, CN=example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:ca:85:f4:3a:b7:5f:e2:66:be:fc:d8:97:65:3d:
                    a4:3d:30:c6:02:0a:9e:1c:ca:bb:15:63:ca:22:81:
                    00:e1:a9:c0:69:64:75:57:56:53:a1:99:ee:e1:cd:
                    ...
                    aa:38:73:ff:3d:b7:00:74:82:8e:4a:5d:da:5f:79:
                    5a:89:52:e7:de:68:95:e0:16:9b:47:2d:57:49:2d:
                    9b:41:53:e2:7f:e1:bd:95:bf:eb:b3:a3:72:d6:a4:
                    d3:63
                Exponent: 65537 (0x10001)
        Attributes:
            a0:00
    Signature Algorithm: sha256WithRSAEncryption
         74:18:26:72:33:be:ef:ae:1d:1e:ff:15:e5:28:db:c1:e0:80:
         42:2c:82:5a:34:aa:1a:70:df:fa:4f:19:e2:5a:0e:33:38:af:
         21:aa:14:b4:85:35:9c:dd:73:98:1c:b7:ce:f3:ff:43:aa:11:
         ....
         3c:b2:62:94:ad:94:11:55:c2:43:e0:5f:3b:39:d3:a6:4b:47:
         09:6b:9d:6b:9b:95:15:10:25:be:8b:5c:cc:f1:ff:7b:26:6b:
         fa:81:df:e4:92:e5:3c:e5:7f:0e:d8:d9:6f:c5:a6:67:fb:2b:
         0b:53:e5:22
```

El siguiente comando crea un certificado. Como no se especifica ninguna plantilla, se emite un certificado de entidad final base de forma predeterminada.

```
$ aws acm-pca issue-certificate \
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
      --csr fileb://csr.pem \
      --signing-algorithm "SHA256WITHRSA" \
      --validity Value=365,Type="DAYS"
```

Se devuelve el ARN del certificado emitido:

```
{
```

```
   "CertificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
}
```

> **Note**
>
> Autoridad de certificación privada de AWS devuelve inmediatamente un ARN con un número de serie cuando recibe el issue-certificate comando. Sin embargo, el procesamiento del certificado se realiza de forma asíncrona y, aun así, puede fallar. Si esto ocurre, un comando get-certificate que utilice el nuevo ARN también fallará.

Emita un certificado con un nombre de asunto personalizado mediante una plantilla APIPassthrough

En este ejemplo, se emite un certificado que contiene elementos de nombre de asunto personalizados. Además de proporcionar una CSR como la que aparece enEmita un certificado estándar (AWS CLI), se pasan dos argumentos adicionales al issue-certificate comando: el ARN de APIPassthrough una plantilla y un archivo de configuración JSON que especifica los atributos personalizados y sus identificadores OIDs de objeto (). No puedes usarlo `StandardAttributes` junto con`CustomAttributes`. Sin embargo, puedes pasar el estándar OIDs como parte de. `CustomAttributes` El nombre OIDs del asunto predeterminado aparece en la siguiente tabla (información de la RFC 4519 y de la base de datos de referencia global OID):

Nombre del asunto	Abreviatura	ID de objeto
countryName	c	2.5.4.6
CommonName	cn	2.5.4.3
dnQualifier [calificador de nombre distintivo]		2.5.4,46
generationQualifier		2.5.4,44
givenName		2.5.4,42
initials		2.5.4,43

Nombre del asunto	Abreviatura	ID de objeto
locality	l	2.5.4.7
organizationName	o	2.5.4,10
organizationalUnitName	ou	2.5.4,11
pseudonym		2.5.4,65
serialNumber		2.5.4.5
st [estado]		2.5.4.8
surname	sn	2.5.4.4
título		2.5.4.12
domainComponent	dc	0,9,234,19200300.100,1,25
userid		0,9,2342 19200300.1001.1

El archivo de configuración de ejemplo `api_passthrough_config.txt` contiene el código siguiente:

```
{
  "Subject": {
    "CustomAttributes": [
      {
        "ObjectIdentifier": "2.5.4.6",
        "Value": "US"
      },
      {
        "ObjectIdentifier": "1.3.6.1.4.1.37244.1.1",
        "Value": "BCDABCDA12341234"
      },
      {
        "ObjectIdentifier": "1.3.6.1.4.1.37244.1.5",
        "Value": "CDABCDAB12341234"
      }
    ]
```

```
  }
}
```

Escriba el siguiente comando para extraer el certificado:

```
$ aws acm-pca issue-certificate \
      --validity Type=DAYS,Value=10
      --signing-algorithm "SHA256WITHRSA" \
      --csr fileb://csr.pem \
      --api-passthrough file://api_passthrough_config.txt \
      --template-arn arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APIPassthrough/V1 \
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
```

Se devuelve el ARN del certificado emitido:

```
{
   "CertificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
}
```

Recupere el certificado localmente de la siguiente manera:

```
$ aws acm-pca get-certificate \
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
      --certificate-arn arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID | \
      jq -r .'Certificate' > cert.pem
```

Puede inspeccionar el contenido del certificado mediante OpenSSL:

```
$ openssl x509 -in cert.pem -text -noout
```

> **Note**
>
> También es posible crear una CA privada que transfiera atributos personalizados a cada certificado que emita.

Emita un certificado con extensiones personalizadas mediante una plantilla APIPassthrough

En este ejemplo, se emite un certificado que contiene extensiones personalizadas. Para ello, debe pasar tres argumentos al issue-certificate comando: el ARN de una APIPassthrough plantilla y un archivo de configuración JSON que especifica las extensiones personalizadas, y una CSR como la que se muestra en. Emita un certificado estándar (AWS CLI)

El archivo de configuración de ejemplo `api_passthrough_config.txt` contiene el código siguiente:

```
{
  "Extensions": {
    "CustomExtensions": [
      {
        "ObjectIdentifier": "2.5.29.30",
        "Value": "MBWgEzARgg8ucGVybWl0dGVkLnRlc3Q=",
        "Critical": true
      }
    ]
  }
}
```

El certificado personalizado se emite de la siguiente manera:

```
$ aws acm-pca issue-certificate \
      --validity Type=DAYS,Value=10
      --signing-algorithm "SHA256WITHRSA" \
      --csr fileb://csr.pem \
      --api-passthrough file://api_passthrough_config.txt \
      --template-arn arn:aws:acm-pca:::template/EndEntityCertificate_APIPassthrough/V1
 \
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
```

Se devuelve el ARN del certificado emitido:

```
{
   "CertificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
}
```

Recupere el certificado localmente de la siguiente manera:

```
$ aws acm-pca get-certificate \
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
      --certificate-arn arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID | \
      jq -r .'Certificate' > cert.pem
```

Puede inspeccionar el contenido del certificado mediante OpenSSL:

```
$ openssl x509 -in cert.pem -text -noout
```

Recupera un certificado privado

Puede utilizar la Autoridad de certificación privada de AWS API y AWS CLI emitir un certificado privado. Si lo hace, puede usar la Autoridad de certificación privada de AWS API AWS CLI o para recuperar ese certificado. Si utilizó ACM para crear una CA privada y para solicitar certificados, debe utilizar ACM para exportar el certificado y la clave privada cifrada. Para obtener más información, consulte Exportación de un certificado privado.

Para recuperar un certificado de entidad final

Use el AWS CLI comando get-certificate para recuperar un certificado de entidad final privada. También puede utilizar la operación de API. GetCertificate Recomendamos formatear el resultado con jq, un analizador similar a un set.

> **Note**
>
> Si desea revocar un certificado, puede utilizar el comando get-certificate para recuperar el número de serie en formato hexadecimal. También puede crear un informe de auditoría para recuperarlo. Para obtener más información, consulte Utilice los informes de auditoría con su CA privada.

```
$ aws acm-pca get-certificate \
      --certificate-arn arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID \
```

```
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 | \
      jq -r '.Certificate, .CertificateChain'
```

Este comando genera el certificado y la cadena de certificados con formato estándar.

```
-----BEGIN CERTIFICATE-----
...base64-encoded certificate...
-----END CERTIFICATE----
-----BEGIN CERTIFICATE-----
...base64-encoded certificate...
-----END CERTIFICATE----
-----BEGIN CERTIFICATE-----
...base64-encoded certificate...
-----END CERTIFICATE----
```

Para recuperar el certificado de una CA

Puede usar la Autoridad de certificación privada de AWS API AWS CLI para recuperar el certificado de la autoridad de certificación (CA) de su CA privada. Ejecute el comando get-certificate-authority-certificate. También puede llamar a la operación GetCertificateAuthorityCertificate. Recomendamos formatear el resultado con jq, un analizador similar a un set.

```
$ aws acm-pca get-certificate-authority-certificate \
     --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
     | jq -r '.Certificate'
```

Este comando genera el certificado de CA con formato estándar.

```
-----BEGIN CERTIFICATE-----
...base64-encoded certificate...
-----END CERTIFICATE----
```

Enumere los certificados privados

Para enumerar sus certificados privados, genere un informe de auditoría, extráigalo de su bucket de S3 y analice el contenido del informe según sea necesario. Para obtener más información sobre cómo crear los informes de auditoría de Autoridad de certificación privada de AWS , consulte Utilice los informes de auditoría con su CA privada. Para obtener más información sobre cómo recuperar un

objeto de un bucket de S3, consulte Descargar un objeto en la Guía del usuario de Amazon Simple Storage Service.

Los siguientes ejemplos ilustran los enfoques para crear informes de auditoría y analizarlos para obtener datos útiles. Los resultados se formatean en JSON y los datos se filtran con jq, un analizador similar a un sed.

1. Creación de un informe de auditoría.

El siguiente comando genera un informe de auditoría para una CA específica.

```
$ aws acm-pca create-certificate-authority-audit-report \
     --region region \
     --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
     --s3-bucket-name bucket_name \
     --audit-report-response-format JSON
```

Si se ejecuta correctamente, el comando devuelve el ID y la ubicación del nuevo informe de auditoría.

```
{
   "AuditReportId":"audit_report_ID",
   "S3Key":"audit-report/CA_ID/audit_report_ID.json"
}
```

2. Recupera y formatea un informe de auditoría.

Este comando recupera un informe de auditoría, muestra su contenido en una salida estándar y filtra los resultados para mostrar solo los certificados emitidos a partir del 1 de diciembre de 2020.

```
$ aws s3api get-object \
     --region region \
     --bucket bucket_name \
     --key audit-report/CA_ID/audit_report_ID.json \
     /dev/stdout | jq '.[] | select(.issuedAt >= "2020-12-01")'
```

Los elementos devueltos arrojaron los siguientes resultados:

```
{
   "awsAccountId":"account",
```

```
   "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial":"serial_number",
   "subject":"CN=pca.alpha.root2.leaf5",
   "notBefore":"2020-12-21T21:28:09+0000",
   "notAfter":"9999-12-31T23:59:59+0000",
   "issuedAt":"2020-12-21T22:28:09+0000",
   "templateArn":"arn:aws:acm-pca:::template/EndEntityCertificate/V1"
}
```

3. Guarde un informe de auditoría localmente.

Si desea realizar varias consultas, es conveniente guardar un informe de auditoría en un archivo local.

```
$ aws s3api get-object \
     --region region \
     --bucket bucket_name \
     --key audit-report/CA_ID/audit_report_ID.json > my_local_audit_report.json
```

El mismo filtro que antes produce el mismo resultado:

```
$ cat my_local_audit_report.json | jq '.[] | select(.issuedAt >= "2020-12-01")'
{
   "awsAccountId":"account",
   "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial":"serial_number",
   "subject":"CN=pca.alpha.root2.leaf5",
   "notBefore":"2020-12-21T21:28:09+0000",
   "notAfter":"9999-12-31T23:59:59+0000",
   "issuedAt":"2020-12-21T22:28:09+0000",
   "templateArn":"arn:aws:acm-pca:::template/EndEntityCertificate/V1"
}
```

4. Consulta dentro de un intervalo de fechas

Puede consultar los certificados emitidos dentro de un intervalo de fechas de la siguiente manera:

```
$ cat my_local_audit_report.json | jq '.[] | select(.issuedAt >= "2020-11-01"
 and .issuedAt <= "2020-11-10")'
```

El contenido filtrado se muestra en la salida estándar:

```
{
   "awsAccountId": "account",
   "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial": "serial_number",
   "subject": "CN=pca.alpha.root2.leaf1",
   "notBefore": "2020-11-06T19:18:21+0000",
   "notAfter": "9999-12-31T23:59:59+0000",
   "issuedAt": "2020-11-06T20:18:22+0000",
   "templateArn": "arn:aws:acm-pca:::template/EndEntityCertificate/V1"
}
{
   "awsAccountId": "account",
   "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial": "serial_number",
   "subject": "CN=pca.alpha.root2.rsa2048sha256",
   "notBefore": "2020-11-06T19:15:46+0000",
   "notAfter": "9999-12-31T23:59:59+0000",
   "issuedAt": "2020-11-06T20:15:46+0000",
   "templateArn": "arn:aws:acm-pca:::template/RootCACertificate/V1"
}
{
   "awsAccountId": "account",
   "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial": "serial_number",
   "subject": "CN=pca.alpha.root2.leaf2",
   "notBefore": "2020-11-06T20:04:39+0000",
   "notAfter": "9999-12-31T23:59:59+0000",
   "issuedAt": "2020-11-06T21:04:39+0000",
   "templateArn": "arn:aws:acm-pca:::template/EndEntityCertificate/V1"
}
```

5. Busque certificados siguiendo una plantilla especificada.

El siguiente comando filtra el contenido del informe mediante una plantilla ARN:

```
$ cat my_local_audit_report.json | jq '.[] | select(.templateArn == "arn:aws:acm-
pca:::template/RootCACertificate/V1")'
```

El resultado muestra los registros de certificados coincidentes:

```
{
   "awsAccountId": "account",
   "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial": "serial_number",
   "subject": "CN=pca.alpha.root2.rsa2048sha256",
   "notBefore": "2020-11-06T19:15:46+0000",
   "notAfter": "9999-12-31T23:59:59+0000",
   "issuedAt": "2020-11-06T20:15:46+0000",
   "templateArn": "arn:aws:acm-pca:::template/RootCACertificate/V1"
}
```

6. Filtrar los certificados revocados

Para encontrar todos los certificados revocados, ejecute el siguiente comando:

```
$ cat my_local_audit_report.json | jq '.[] | select(.revokedAt != null)'
```

Un certificado revocado se muestra de la siguiente manera:

```
{
   "awsAccountId": "account",
   "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial": "serial_number",
   "subject": "CN=pca.alpha.root2.leaf2",
   "notBefore": "2020-11-06T20:04:39+0000",
   "notAfter": "9999-12-31T23:59:59+0000",
   "issuedAt": "2020-11-06T21:04:39+0000",
   "revokedAt": "2021-05-27T18:57:32+0000",
   "revocationReason": "UNSPECIFIED",
   "templateArn": "arn:aws:acm-pca:::template/EndEntityCertificate/V1"
}
```

7. Filtración mediante una expresión regular.

El siguiente comando busca los nombres de los asuntos que contienen la cadena “hoja”:

```
$ cat my_local_audit_report.json | jq '.[] | select(.subject|test("leaf"))'
```

Los registros de certificados coincidentes se devuelven de la siguiente manera:

```
{
   "awsAccountId": "account",
   "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial": "serial_number",
   "subject": "CN=pca.alpha.roo2.leaf4",
   "notBefore": "2020-11-16T18:17:10+0000",
   "notAfter": "9999-12-31T23:59:59+0000",
   "issuedAt": "2020-11-16T19:17:12+0000",
   "templateArn": "arn:aws:acm-pca:::template/EndEntityCertificate/V1"
}
{
   "awsAccountId": "account",
   "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial": "serial_number",
   "subject": "CN=pca.alpha.root2.leaf5",
   "notBefore": "2020-12-21T21:28:09+0000",
   "notAfter": "9999-12-31T23:59:59+0000",
   "issuedAt": "2020-12-21T22:28:09+0000",
   "templateArn": "arn:aws:acm-pca:::template/EndEntityCertificate/V1"
}
{
   "awsAccountId": "account",
   "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial": "serial_number",
   "subject": "CN=pca.alpha.root2.leaf1",
   "notBefore": "2020-11-06T19:18:21+0000",
   "notAfter": "9999-12-31T23:59:59+0000",
   "issuedAt": "2020-11-06T20:18:22+0000",
   "templateArn": "arn:aws:acm-pca:::template/EndEntityCertificate/V1"
}
```

Exporte un certificado privado y su clave secreta

Autoridad de certificación privada de AWS no puede exportar directamente un certificado privado que haya firmado y emitido. Sin embargo, puede utilizarlo AWS Certificate Manager para exportar dicho certificado junto con su clave secreta cifrada. De este modo, el certificado es completamente

portátil para su implementación en cualquier lugar de su PKI privada. Para obtener más información, consulte Exportación de un certificado privado en la Guía del AWS Certificate Manager usuario.

Como ventaja adicional, AWS Certificate Manager ofrece la renovación gestionada de los certificados privados que se emitieron mediante la consola ACM, la `RequestCertificate` acción de la API de ACM o el request-certificate comando de la sección ACM del. AWS CLI Para obtener más información sobre las renovaciones, consulte Renovación de certificados en una PKI privada.

Revocar un certificado privado

Puede revocar un Autoridad de certificación privada de AWS certificado mediante el AWS CLI comando revoke-certificate o la acción de la API. RevokeCertificate Es posible que sea necesario revocar un certificado antes de su caducidad programada si, por ejemplo, su clave secreta está comprometida o su dominio asociado deja de ser válido. Para que la revocación sea efectiva, el cliente que utilice el certificado necesita una forma de comprobar el estado de la revocación siempre que intente crear una conexión de red segura.

Autoridad de certificación privada de AWS proporciona dos mecanismos totalmente gestionados que permiten comprobar el estado de las revocaciones: el Protocolo de estado de certificados en línea (OCSP) y las listas de revocación de certificados (). CRLs Con el OCSP, el cliente consulta una base de datos de revocaciones autorizada que devuelve un estado en tiempo real. Con una CRL, el cliente compara el certificado con una lista de certificados revocados que descarga y almacena periódicamente. Los clientes se niegan a aceptar certificados que han sido revocados.

Tanto el OCSP como el OCSP CRLs dependen de la información de validación incluida en los certificados. Por este motivo, la CA emisora debe configurarse para admitir uno de estos mecanismos o ambos antes de su emisión. Para obtener información sobre cómo seleccionar e implementar la revocación gestionada mediante Autoridad de certificación privada de AWS, consulte. Planifique el método de revocación AWS Private CA de su certificado

Los certificados revocados siempre se registran en los informes de Autoridad de certificación privada de AWS auditoría.

> **Note**
>
> Para las personas que llaman desde varias cuentas, es necesario compartirlos con el `AWSRAMRevokeCertificateCertificateAuthority` permiso correspondiente. Los permisos de revocación no están incluidos en. `AWSRAMDefaultPermissionCertificateAuthority` Para permitir la revocación

por parte de los emisores de varias cuentas, el administrador de la CA debe crear dos comparticiones de RAM, ambas dirigidas a la misma CA:

1. Un recurso compartido con el permiso `AWSRAMRevokeCertificateCertificateAuthority`.
2. Un recurso compartido con el permiso `AWSRAMDefaultPermissionCertificateAuthority`.

Para revocar un certificado

Utilice la acción de la RevokeCertificateAPI o el comando revoke-certificate para revocar un certificado de PKI privado. El número de serie debe estar en formato hexadecimal. Puede recuperar el número de serie llamando al comando get-certificate. El comando `revoke-certificate` no devuelve ninguna respuesta.

```
$ aws acm-pca revoke-certificate \
     --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
     --certificate-serial serial_number \
     --revocation-reason "KEY_COMPROMISE"
```

Certificados y OCSP revocados

Cuando se revoca un certificado, las respuestas del OCSP pueden tardar hasta 60 minutos en reflejar el nuevo estado. En general, el OCSP suele permitir una distribución más rápida de la información de revocación porque, a diferencia de CRLs lo que los clientes pueden almacenar en caché durante días, los clientes no suelen almacenar en caché las respuestas de OCSP.

Certificados revocados en una CRL

Las CRL se actualizan aproximadamente 30 minutos después de que un certificado se revoque. Si por alguna razón se produce un error en la actualización de la CRL, Autoridad de certificación privada de AWS realiza nuevos intentos cada 15 minutos.

Con Amazon CloudWatch, puedes crear alarmas para las métricas `CRLGenerated` y`MisconfiguredCRLBucket`. Para obtener más información, consulta CloudWatchMétricas compatibles. Para obtener más información sobre la creación y la configuración CRLs, consulteConfigura una CRL para AWS Private CA.

En el siguiente ejemplo, se muestra un certificado revocado de una lista de revocación de certificados (CRL).

```
Certificate Revocation List (CRL):
        Version 2 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: /C=US/ST=WA/L=Seattle/O=Examples LLC/OU=Corporate Office/
CN=www.example.com
        Last Update: Jan 10 19:28:47 2018 GMT
        Next Update: Jan  8 20:28:47 2028 GMT
        CRL extensions:
            X509v3 Authority key identifier:
                keyid:3B:F0:04:6B:51:54:1F:C9:AE:4A:C0:2F:11:E6:13:85:D8:84:74:67

            X509v3 CRL Number:
                1515616127629
Revoked Certificates:
    Serial Number: B17B6F9AE9309C51D5573BCA78764C23
        Revocation Date: Jan  9 17:19:17 2018 GMT
        CRL entry extensions:
            X509v3 CRL Reason Code:
                Key Compromise
    Signature Algorithm: sha256WithRSAEncryption
         21:2f:86:46:6e:0a:9c:0d:85:f6:b6:b6:db:50:ce:32:d4:76:
         99:3e:df:ec:6f:c7:3b:7e:a3:6b:66:a7:b2:83:e8:3b:53:42:
         f0:7a:bc:ba:0f:81:4d:9b:71:ee:14:c3:db:ad:a0:91:c4:9f:
         98:f1:4a:69:9a:3f:e3:61:36:cf:93:0a:1b:7d:f7:8d:53:1f:
         2e:f8:bd:3c:7d:72:91:4c:36:38:06:bf:f9:c7:d1:47:6e:8e:
         54:eb:87:02:33:14:10:7f:b2:81:65:a1:62:f5:fb:e1:79:d5:
         1d:4c:0e:95:0d:84:31:f8:5d:59:5d:f9:2b:6f:e4:e6:60:8b:
         58:7d:b2:a9:70:fd:72:4f:e7:5b:e4:06:fc:e7:23:e7:08:28:
         f7:06:09:2a:a1:73:31:ec:1c:32:f8:dc:03:ea:33:a8:8e:d9:
         d4:78:c1:90:4c:08:ca:ba:ec:55:c3:00:f4:2e:03:b2:dd:8a:
         43:13:fd:c8:31:c9:cd:8d:b3:5e:06:c6:cc:15:41:12:5d:51:
         a2:84:61:16:a0:cf:f5:38:10:da:a5:3b:69:7f:9c:b0:aa:29:
         5f:fc:42:68:b8:fb:88:19:af:d9:ef:76:19:db:24:1f:eb:87:
         65:b2:05:44:86:21:e0:b4:11:5c:db:f6:a2:f9:7c:a6:16:85:
         0e:81:b2:76
```

Certificados revocados en un informe de auditoría

Todos los certificados, incluidos los que se han revocado, se incluyen en el informe de auditoría de una CA privada. En el ejemplo siguiente, se muestra un informe de auditoría con un certificado

emitido y un certificado revocado. Para obtener más información, consulte Utilice los informes de auditoría con su CA privada.


```
[
   {
      "awsAccountId":"account",
      "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
      "serial":"serial_number",

 "Subject":"1.2.840.113549.1.9.1=#161173616c6573406578616d706c652e636f6d,CN=www.example1.com,OU
 Company,L=Seattle,ST=Washington,C=US",
      "notBefore":"2018-02-26T18:39:57+0000",
      "notAfter":"2019-02-26T19:39:57+0000",
      "issuedAt":"2018-02-26T19:39:58+0000",
      "revokedAt":"2018-02-26T20:00:36+0000",
      "revocationReason":"KEY_COMPROMISE"
   },
   {
      "awsAccountId":"account",
      "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
      "serial":"serial_number",

 "Subject":"1.2.840.113549.1.9.1=#161970726f64407777772e70616c6f75736573616c65732e636f6d,CN=www
 Company,L=Seattle,ST=Washington,C=US",
      "notBefore":"2018-01-22T20:10:49+0000",
      "notAfter":"2019-01-17T21:10:49+0000",
      "issuedAt":"2018-01-22T21:10:49+0000"
   }
]
```


Automatice la exportación de un certificado renovado

Al Autoridad de certificación privada de AWS crear una CA, puede importarla AWS Certificate Manager y dejar que ACM gestione la emisión y renovación de los certificados. Si un certificado que se está renovando está asociado a un servicio integrado, el servicio se aplica el nuevo certificado sin problemas. Sin embargo, si el certificado se exportó originalmente para usarlo en otro lugar de su entorno de PKI (por ejemplo, en un servidor o dispositivo en las instalaciones), tendrá que volver a exportarlo después de la renovación.

Para ver un ejemplo de solución que automatiza el proceso de exportación de ACM mediante Amazon y EventBridge AWS Lambda, consulte Automatizar la exportación de certificados renovados.

Utilice plantillas de certificados AWS Private CA

Autoridad de certificación privada de AWS utiliza plantillas de configuración para emitir certificados de CA y certificados de entidad final. Al emitir un certificado de CA desde la consola de PCA, se aplica automáticamente la plantilla de certificado de CA raíz o subordinada correspondiente.

Si utiliza la CLI o la API para emitir un certificado, puede proporcionar una plantilla ARN como parámetro de la acción `IssueCertificate`. La plantilla `EndEntityCertificate/V1` se aplica si no proporciona un ARN. Para obtener más información, consulte la documentación sobre la IssueCertificateAPI y los comandos issue-certificate.

> **Note**
>
> AWS Certificate Manager (ACM) los usuarios con acceso compartido entre cuentas a una CA privada pueden emitir certificados gestionados firmados por la CA. Los emisores multicuentas están limitados por una política basada en los recursos y solo tienen acceso a las siguientes plantillas de certificados de entidad final:
>
> - EndEntityCertificate/V1
> - EndEntityClientAuthCertificate/V1
> - EndEntityServerAuthCertificate/V1
> - BlankEndEntityCertificate_ /V1 APIPassthrough
> - BlankEndEntityCertificate_ /V1 APICSRPassthrough
> - Subordinado _ 0/V1 CACertificate PathLen
>
> Para obtener más información, consulte Políticas basadas en recursos.

Temas

- AWS Private CA variedades de plantillas
- AWS Private CA orden de operaciones de la plantilla
- AWS Private CA definiciones de plantillas

AWS Private CA variedades de plantillas

Autoridad de certificación privada de AWS admite cuatro tipos de plantillas.

- Plantillas base

  Plantillas predefinidas en las que no se permiten parámetros de acceso directo.

- CSRPassthrough plantillas

  Plantillas que amplían sus versiones de plantillas base correspondientes al permitir el acceso directo de CSR. Las extensiones de la CSR que se utilizan para emitir el certificado se copian en el certificado emitido. En los casos en que la CSR contenga valores de extensión que entren en conflicto con la definición de la plantilla, esta siempre tendrá la mayor prioridad. Para obtener más información sobre la prioridad, consulte AWS Private CA orden de operaciones de la plantilla.

- APIPassthrough plantillas

  Plantillas que amplían sus versiones de plantillas base correspondientes al permitir el acceso directo de API. Es posible que la entidad que solicita el certificado no conozca los valores dinámicos que el administrador u otros sistemas intermedios conocen, que no se puedan definir en una plantilla y que no estén disponibles en la CSR. Sin embargo, el administrador de la CA puede recuperar información adicional de otro origen de datos, como un Active Directory, para completar la solicitud. Por ejemplo, si una máquina no sabe a qué unidad organizativa pertenece, el administrador puede buscar la información en Active Directory y añadirla a la solicitud de certificado incluyendo la información en una estructura JSON.

  Los valores del parámetro `ApiPassthrough` de la acción `IssueCertificate` se copian en el certificado emitido. En los casos en que el parámetro `ApiPassthrough` contenga valores de extensión que entren en conflicto con la definición de la plantilla, esta siempre tendrá la mayor prioridad. Para obtener más información sobre la prioridad, consulte AWS Private CA orden de operaciones de la plantilla.

- APICSRPassthrough plantillas

  Plantillas que amplían sus versiones de plantillas base correspondientes al permitir el acceso directo de API y CSR. Las extensiones de la CSR utilizadas para emitir el certificado se copian en el certificado emitido y también se copian los valores del parámetro `ApiPassthrough` de la acción `IssueCertificate`. En los casos en que la definición de la plantilla, los valores de transferencia de la API y las extensiones de transferencia de la CSR presentan un conflicto, la definición de la plantilla tiene la máxima prioridad, seguida de los valores de acceso directo de

la API y, a continuación, de las extensiones de acceso directo de la CSR. Para obtener más información sobre la prioridad, consulte AWS Private CA orden de operaciones de la plantilla.

En las tablas siguientes se enumeran todos los tipos de plantillas compatibles Autoridad de certificación privada de AWS con enlaces a sus definiciones.

> **Note**
>
> Para obtener información sobre las plantillas ARNs en GovCloud las regiones, consulte AWS Private Certificate Authorityla Guía del AWS GovCloud (US) usuario.

Plantillas base

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
CodeSigningCertificate/V1	`arn:aws:acm-pca:::template/CodeSigningCertificate/V1`	Firma de código
EndEntityCertificate/V1	`arn:aws:acm-pca:::template/EndEntityCertificate/V1`	Entidad final
EndEntityClientAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate/V1`	Entidad final
EndEntityServerAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate/V1`	Entidad final
OCSPSigningCertificado/V1	`arn:aws:acm-pca:::template/OCSPSigningCertificate/V1`	Firma de OCSP

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
Raíz /V1 CACertificate	`arn:aws:acm-pca:::template/RootCACertificate/V1`	CA
Subordinado _ 0/V1 CACertificate PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1`	CA
Subordinado _ 1/V1 CACertificate PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1/V1`	CA
Subordinado _ 2/V1 CACertificate PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2/V1`	CA
Subordinado _ 3/V1 CACertificate PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3/V1`	CA

CSRPassthrough plantillas

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
BlankEndEntityCertificate_ CSRPassthrough /V1	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_CSRPassthrough/V1`	Entidad final
BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1 CSRPassthrough	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_C`	Entidad final

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
	riticalBasicConstraints_CSRPassthrough/V1	
BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA
CodeSigningCertificate_ /V1 CSRPassthrough	arn:aws:acm-pca:::template/CodeSigningCertificate_CSRPassthrough/V1	Firma de código

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
EndEntityCertificate_ /V1 CSRPassthrough	arn:aws:acm-pca:::template/EndEntityCertificate_CSRPassthrough/V1	Entidad final
EndEntityClientAuthCertificate_ /V1 CSRPassthrough	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_CSRPassthrough/V1	Entidad final
EndEntityServerAuthCertificate_ /V1 CSRPassthrough	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_CSRPassthrough/V1	Entidad final
OCSPSigningCertificado_ /V1 CSRPassthrough	arn:aws:acm-pca:::template/OCSPSigningCertificate_CSRPassthrough/V1	Firma de OCSP
Subordinado _ 0_ /V1 CACertificate PathLen CSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
Subordinado _ 1_ /V1 CACertificate PathLen CSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
Subordinado _ 2_ /V1 CACertificate PathLen CSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
Subordinado _ 3_ /V1 CACertificate PathLen CSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA

APIPassthrough plantillas

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
BlankEndEntityCertificate_ APIPassthrough /V1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_APIPassthrough/V1	Entidad final
BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1 APIPassthrough	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough/V1	Entidad final
CodeSigningCertificate_ /V1 APIPassthrough	arn:aws:acm-pca:::template/CodeSigningCertificate_APIPassthrough/V1	Firma de código
EndEntityCertificate_ /V1 APIPassthrough	arn:aws:acm-pca:::template/EndEntity	Entidad final

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
	Certificate_APIPas sthrough/V1	
EndEntityClientAuthCertificate_ / V1 APIPassthrough	arn:aws:acm-pca::: template/EndEntity ClientAuthCertific ate_APIPassthrough/V1	Entidad final
EndEntityServerAuthCertificate_ / V1 APIPassthrough	arn:aws:acm-pca::: template/EndEntity ServerAuthCertific ate_APIPassthrough/V1	Entidad final
OCSPSigningCertificado_ /V1 APIPassthrough	arn:aws:acm-pca::: template/OCSPSigni ngCertificate_APIP assthrough/V1	Firma de OCSP
Raíz _ /V1 CACertificate APIPassthrough	arn:aws:acm-pca::: template/RootCACer tificate_APIPassth rough/V1	CA
BlankRootCACertificate_ /V1 APIPassthrough	arn:aws:acm-pca::: template/BlankRoot CACertificate_APIP assthrough/V1	CA
BlankRootCACertificate_ PathLen 0_ /V1 APIPassthrough	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len0_APIPassthrough/V1	CA

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
BlankRootCACertificate_ 1_ /V1 PathLen APIPassthrough	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len1_APIPassthrough/V1	CA
BlankRootCACertificate_ 2_ /V1 PathLen APIPassthrough	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len2_APIPassthrough/V1	CA
BlankRootCACertificate_ 3_ /V1 PathLen APIPassthrough	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len3_APIPassthrough/V1	CA
Subordinado _ 0_ /V1 CACertificate PathLen APIPassthrough	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen0_APIPassthrough/ V1	CA
BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen0_APIPas sthrough/V1	CA
Subordinado _ 1_ /V1 CACertificate PathLen APIPassthrough	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen1_APIPassthrough/ V1	CA

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
Subordinado _ 2_ /V1 CACertificate PathLen APIPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
Subordinado _ 3_ /V1 CACertificate PathLen APIPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APIPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1	CA

APICSRPassthrough plantillas

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
BlankEndEntityCertificate_ APICSRPassthrough /V1	`arn:aws:acm-pca::: template/BlankEndE ntityCertificate_A PICSRPassthrough/V1`	Entidad final
BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1 APICSRPassthrough	`arn:aws:acm-pca::: template/BlankEndE ntityCertificate_C riticalBasicConstr aints_APICSRPassth rough/V1`	Entidad final
CodeSigningCertificate_ /V1 APICSRPassthrough	`arn:aws:acm-pca::: template/CodeSigni ngCertificate_APIC SRPassthrough/V1`	Firma de código
EndEntityCertificate_ /V1 APICSRPassthrough	`arn:aws:acm-pca::: template/EndEntity Certificate_APICSR Passthrough/V1`	Entidad final
EndEntityClientAuthCertificate_ / V1 APICSRPassthrough	`arn:aws:acm-pca::: template/EndEntity ClientAuthCertific ate_APICSRPassthrough/ V1`	Entidad final
EndEntityServerAuthCertificate_ / V1 APICSRPassthrough	`arn:aws:acm-pca::: template/EndEntity ServerAuthCertific`	Entidad final

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
	ate_APICSRPassthrough/V1	
OCSPSigningCertificado_ /V1 APICSRPassthrough	arn:aws:acm-pca:::template/OCSPSigningCertificate_APICSRPassthrough/V1	Firma de OCSP
Subordinado _ 0_ /V1 CACertificate PathLen APICSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
Subordinado _ 1_ /V1 CACertificate PathLen APICSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
Subordinado CACertificate _ 2_/3_ V1 PathLen APICSRPassthrough PathLen APIPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APICSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1	CA
Subordinado _ 3_ /V1 CACertificate PathLen APICSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APICSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1	CA

AWS Private CA orden de operaciones de la plantilla

La información contenida en un certificado emitido puede proceder de cuatro fuentes: la definición de la plantilla, el acceso directo de la API, el acceso directo de la CSR y la configuración de la CA.

Los valores de acceso directo de API solo se respetan cuando se utiliza una plantilla de acceso directo de API o de acceso directo de APICSR. La transferencia CSR solo se respeta cuando se utiliza una plantilla de transferencia APICSR CSRPassthrough o APICSR. Cuando estas fuentes de información entran en conflicto, suele aplicarse una regla general: para cada valor de extensión, la definición de la plantilla tiene la máxima prioridad, seguida de los valores de acceso directo de la API y, a continuación, de las extensiones de acceso directo de la CSR.

Ejemplos

1. La definición de plantilla para EndEntityClientAuthCertificate_ APIPassthrough define la ExtendedKeyUsage extensión con un valor de «autenticación de servidor web TLS, autenticación de cliente web TLS». Si ExtendedKeyUsage está definido en la CSR o en el `IssueCertificate ApiPassthrough` parámetro, el `ApiPassthrough` valor de se ExtendedKeyUsage omitirá porque la definición de la plantilla tiene prioridad, y el valor de la CSR se ignorará porque la plantilla no es del tipo de transferencia de CSR. ExtendedKeyUsage

   > **Note**
   >
   > No obstante, la definición de la plantilla copia otros valores de la CSR, como el asunto y el nombre alternativo del asunto. Estos valores siguen tomándose de la CSR, aunque la plantilla no sea de tipo acceso directo de CSR, ya que la definición de la plantilla siempre tiene la máxima prioridad.

2. La definición de plantilla para EndEntityClientAuthCertificate_ APICSRPassthrough define la extensión del nombre alternativo del sujeto (SAN) como una copia de la API o la CSR. Si la extensión SAN se define en la CSR y se proporciona en el parámetro `IssueCertificate ApiPassthrough`, el valor de transferencia de la API tendrá prioridad, ya que los valores de acceso directo de la API tienen prioridad sobre los valores de acceso directo de la CSR.

AWS Private CA definiciones de plantillas

En las siguientes secciones se proporcionan detalles de configuración sobre las plantillas de certificados de Autoridad de certificación privada de AWS compatibles.

BlankEndEntityCertificateDefinición de APIPassthrough _/V1

Con las plantillas de certificados de entidad final en blanco, puede emitir certificados de entidad final solo con las restricciones básicas de X.509. Este es el certificado de entidad final más simple que Autoridad de certificación privada de AWS se puede emitir, pero se puede personalizar mediante la estructura de la API. La extensión de restricciones básicas define si el certificado es o no un certificado de CA. Una plantilla de certificado de entidad final en blanco aplica el valor FALSE a las restricciones básicas para garantizar que se emita un certificado de entidad final y no un certificado de CA.

Puede usar plantillas de transferencia en blanco para emitir certificados de tarjetas inteligentes que requieran valores específicos para el uso de claves (KU) y el uso extendido de claves (EKU). Por ejemplo, el uso prolongado de claves puede requerir la autenticación del cliente y el inicio de sesión con tarjeta inteligente, y el uso de claves puede requerir la firma digital, el no rechazo y el cifrado de la clave. A diferencia de otras plantillas de transferencia, las plantillas de certificados de entidad final en blanco permiten configurar las extensiones KU y EKU, donde KU puede ser cualquiera de los nueve valores admitidos (DigitalSignature, NonRepudiation, KeyEnCipherment, DataEnCipherment keyCertSign, KeyAgreementRLSign, c, EnCipherOnly y DecipherOnly) y EKU puede ser cualquiera de los valores admitidos (ServerAuth, ClientAuth, codesign, EmailProtection, timestamping, y OCSPSigning) además de extensiones personalizadas.

BlankEndEntityCertificateAPIPassthrough_/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	CA:FALSE
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankEndEntityCertificate_ APICSRPassthrough /V1 definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankEndEntityCertificate_ /V1 APICSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	CA:FALSE
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankEndEntityCertificate_ CriticalBasicConstraints _ APICSRPassthrough /V1 definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1 APICSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, CA:FALSE
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]

Parámetro X509v3	Valor
Punto de distribución de CRL*	[Acceso a través de la configuración de CA, la API o la CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankEndEntityCertificate_ CriticalBasicConstraints _ APIPassthrough /V1 definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1 APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, CA:FALSE
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso a través de la configuración de CA o la API]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankEndEntityCertificate_ CriticalBasicConstraints _ CSRPassthrough /V1 definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1 CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, CA:FALSE
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankEndEntityCertificate_ CSRPassthrough /V1 definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankEndEntityCertificate_ /V1 CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	CA:FALSE
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 0`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 0`

Parámetro X509v3	Valor
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 0`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 1`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 1`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 1`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 2`

Parámetro X509v3	Valor
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 2`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 2`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 3`
Identificador de clave de entidad	[SKI del certificado de CA]

Parámetro X509v3	Valor
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 3`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1definición

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de APIPassthrough _/V1.

BlankSubordinateCACertificate_PathLen3_APICSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 3`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

CodeSigningCertificatedefinición de /V1

Esta plantilla se utiliza para crear certificados para la firma de código. Puede utilizar certificados de firma de código Autoridad de certificación privada de AWS con cualquier solución de firma de código que se base en una infraestructura de CA privada. Por ejemplo, los clientes que utilizan Code Signing for AWS IoT pueden generar un certificado de firma de código e importarlo a. Autoridad de certificación privada de AWS AWS Certificate Manager Para obtener más información, consulte ¿Para qué sirve la firma de código? AWS IoT y obtenga e importe un certificado de firma de código.

CodeSigningCertificate/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	`CA:FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]

Parámetro X509v3	Valor
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítica, firma digital
Uso extendido de claves	Crítico, firma de código
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

*Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

CodeSigningCertificate_ APICSRPassthrough /V1 definición

Esta plantilla amplía CodeSigningCertificate /V1 para admitir los valores de transferencia de API y CSR.

CodeSigningCertificateAPICSRPassthrough_ /V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	`CA:FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítica, firma digital
Uso extendido de claves	Crítico, firma de código
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

CodeSigningCertificate_ APIPassthrough /V1 definición

Esta plantilla es idéntica a la `CodeSigningCertificate` plantilla con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales a través de la API al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la API.

CodeSigningCertificate_ APIPassthrough /V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	`CA:FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítica, firma digital
Uso extendido de claves	Crítico, firma de código
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

CodeSigningCertificate_ CSRPassthrough /V1 definición

Esta plantilla es idéntica a la `CodeSigningCertificate` plantilla con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS transfiere extensiones adicionales de la solicitud de firma de certificado (CSR) al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

CodeSigningCertificate_ /V1 CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	`CA:FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítica, firma digital
Uso extendido de claves	Crítico, firma de código
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

*Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityCertificateDefinición de /V1

Esta plantilla se utiliza para crear certificados para entidades finales como sistemas operativos o servidores web.

EndEntityCertificate/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]

Parámetro X509v3	Valor
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación de servidor web de TLS, autenticación de cliente web TLS
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

*Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityCertificate_ APICSRPassthrough /V1 definición

Esta plantilla amplía EndEntityCertificate /V1 para admitir los valores de transferencia de API y CSR.

EndEntityCertificateAPICSRPassthrough_ /V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación de servidor web de TLS, autenticación de cliente web TLS
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityCertificate_ APIPassthrough /V1 definición

Esta plantilla es idéntica a la `EndEntityCertificate` plantilla con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales a través de la API al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la API.

EndEntityCertificate_ APIPassthrough /V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación de servidor web de TLS, autenticación de cliente web TLS
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityCertificate_ CSRPassthrough /V1 definición

Esta plantilla es idéntica a la `EndEntityCertificate` plantilla con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS transfiere extensiones adicionales de la solicitud de firma de certificado (CSR) al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

EndEntityCertificate_ /V1 CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación de servidor web de TLS, autenticación de cliente web TLS
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

*Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityClientAuthCertificateDefinición de /V1

Esta plantilla difiere de la `EndEntityCertificate` solo en el valor de uso de clave extendida, que la restringe a la autenticación de cliente web TLS.

EndEntityClientAuthCertificate/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]

Parámetro X509v3	Valor
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación de cliente web de TLS
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

*Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityClientAuthCertificate_ APICSRPassthrough /V1 definición

Esta plantilla amplía EndEntityClientAuthCertificate /V1 para admitir los valores de transferencia de API y CSR.

EndEntityClientAuthCertificateAPICSRPassthrough_ /V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación de cliente web de TLS
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityClientAuthCertificate_ APIPassthrough /V1 definición

Esta plantilla es idéntica a la plantilla `EndEntityClientAuthCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales a través de la API al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la API.

EndEntityClientAuthCertificate_ APIPassthrough /V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación de cliente web de TLS
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityClientAuthCertificate_ CSRPassthrough /V1 definición

Esta plantilla es idéntica a la plantilla `EndEntityClientAuthCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS transfiere extensiones adicionales de la solicitud de firma de certificado (CSR) al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

EndEntityClientAuthCertificate_ /V1 CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación de cliente web de TLS
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

*Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityServerAuthCertificateDefinición de /V1

Esta plantilla difiere de la `EndEntityCertificate` solo en el valor de uso de clave extendida, que la restringe a la autenticación del servidor web TLS.

EndEntityServerAuthCertificate/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]

Parámetro X509v3	Valor
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación del servidor web de TLS
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

*Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityServerAuthCertificate_ APICSRPassthrough /V1 definición

Esta plantilla amplía EndEntityServerAuthCertificate /V1 para admitir los valores de transferencia de API y CSR.

EndEntityServerAuthCertificateAPICSRPassthrough_ /V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación del servidor web de TLS
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityServerAuthCertificate_ APIPassthrough /V1 definición

Esta plantilla es idéntica a la plantilla `EndEntityServerAuthCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales a través de la API al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la API.

EndEntityServerAuthCertificate_ APIPassthrough /V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación del servidor web de TLS
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

EndEntityServerAuthCertificate_ CSRPassthrough /V1 definición

Esta plantilla es idéntica a la plantilla `EndEntityServerAuthCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS transfiere extensiones adicionales de la solicitud de firma de certificado (CSR) al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

EndEntityServerAuthCertificate_ /V1 CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	entidad de certificación:`FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, cifrado de claves
Uso extendido de claves	Autenticación del servidor web de TLS
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

*Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

OCSPSigningDefinición de certificado/V1

Esta plantilla se utiliza para crear certificados para firmar respuestas OCSP. La plantilla es idéntica a la plantilla `CodeSigningCertificate`, excepto que el valor de uso de clave extendida especifica la firma OCSP en lugar de la firma de código.

OCSPSigningCertificado/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	`CA:FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]

Parámetro X509v3	Valor
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítica, firma digital
Uso extendido de claves	Crítico, firma de OCSP
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

*Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

OCSPSigningDefinición de certificado_/V1 APICSRPassthrough

Esta plantilla amplía el OCSPSigning Certificate/V1 para admitir los valores de transferencia de API y CSR.

OCSPSigningAPICSRPassthroughCertificado_/V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	`CA:FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítica, firma digital
Uso extendido de claves	Crítico, firma de OCSP
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

OCSPSigningDefinición de certificado_ /V1 APIPassthrough

Esta plantilla es idéntica a la plantilla `OCSPSigningCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales a través de la API al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la API.

OCSPSigningCertificado_ /V1 APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	`CA:FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítica, firma digital
Uso extendido de claves	Crítico, firma de OCSP
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

OCSPSigningDefinición de certificado_ /V1 CSRPassthrough

Esta plantilla es idéntica a la plantilla `OCSPSigningCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS transfiere extensiones adicionales de la solicitud de firma de certificado (CSR) al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

OCSPSigningCertificado_ /V1 CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	`CA:FALSE`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítica, firma digital
Uso extendido de claves	Crítico, firma de OCSP
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

*Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de raíz /V1 CACertificate

Esta plantilla se utiliza para emitir certificados de entidad de certificación raíz autofirmados. Los certificados de entidad de certificación incluyen una extensión de restricciones básicas críticas con el campo de entidad de certificación establecido en `TRUE` para designar que el certificado se puede utilizar para emitir certificados de CA. La plantilla no especifica una longitud de ruta (pathLenConstraint) porque esto podría inhibir la futura expansión de la jerarquía. Se excluye el uso extendido de claves para evitar el uso del certificado de entidad de certificación como certificado de servidor o cliente TLS. No se especifica información de CRL porque no se puede revocar un certificado autofirmado.

Raíz CACertificate /V1

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]

Parámetro X509v3	Valor
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Firma digital crítica keyCertSign, signo CRL
Punto de distribución de CRL	N/A

Definición de Root CACertificate APIPassthrough _/V1

Esta plantilla amplía Root CACertificate /V1 para admitir los valores de transferencia de la API.

Root _/V1 CACertificate APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`
Identificador de clave de entidad	[Acceso directo desde la API]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Firma digital crítica keyCertSign, signo CRL
Punto de distribución de CRL*	N/A

BlankRootCACertificateDefinición de APIPassthrough _/V1

Con las plantillas de certificados raíz en blanco, puede emitir certificados raíz solo con las restricciones básicas del formato X.509. Este es el certificado raíz más simple que Autoridad de certificación privada de AWS se puede emitir, pero se puede personalizar mediante la estructura de la API. La extensión de restricciones básicas define si el certificado es o no un certificado de CA.

Una plantilla de certificado raíz en blanco impone un valor de cuatro restricciones básicas TRUE para garantizar que se emita un certificado de CA raíz.

Puede utilizar plantillas raíz pasantes en blanco para emitir certificados raíz que requieran valores específicos para el uso de claves (KU). Por ejemplo, el uso de claves puede requerir `keyCertSign` `ycRLSign`, pero no`digitalSignature`. A diferencia de otras plantillas de certificados de transferencia raíz que no están en blanco, las plantillas de certificados raíz en blanco permiten configurar la extensión KU, donde KU puede ser cualquiera de los nueve valores admitidos (`digitalSignaturenonRepudiationkeyEncipherment`,`dataEncipherment`,`keyAgreement`,`keyCe` `cRLSignencipherOnly`, y`decipherOnly`).

BlankRootCACertificate_ /V1 APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`
Identificador de clave de asunto	[Derivado de CSR]

BlankRootCACertificate_ PathLen 0_ /V1 definición APIPassthrough

Para obtener información general sobre las plantillas de CA raíz vacías, consulte. ???

BlankRootCACertificate_ PathLen 0_ /V1 APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 0`
Identificador de clave de asunto	[Derivado de CSR]

BlankRootCACertificate_ PathLen 1_ /V1 definición APIPassthrough

Para obtener información general sobre las plantillas de CA raíz en blanco, consulte. ???

BlankRootCACertificate_ PathLen 1_ /V1 APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 1`
Identificador de clave de asunto	[Derivado de CSR]

BlankRootCACertificate_ PathLen 2_ /V1 definición APIPassthrough

Para obtener información general sobre las plantillas de CA raíz en blanco, consulte. ???

BlankRootCACertificate_ PathLen 2_ /V1 APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 2`
Identificador de clave de asunto	[Derivado de CSR]

BlankRootCACertificate_ PathLen 3_ /V1 definición APIPassthrough

Para obtener información general sobre las plantillas de CA raíz en blanco, consulte. ???

BlankRootCACertificate_ PathLen 3_ /V1 APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]

Parámetro X509v3	Valor
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 3`
Identificador de clave de asunto	[Derivado de CSR]

Definición de subordinado _ 0/V1 CACertificate PathLen

Esta plantilla se utiliza para emitir certificados de CA subordinados con una longitud de ruta de. `0` Los certificados de entidad de certificación incluyen una extensión de restricciones básicas críticas con el campo de entidad de certificación establecido en `TRUE` para designar que el certificado se puede utilizar para emitir certificados de CA. No se incluye el uso extendido de claves, lo que impide que el certificado de entidad de certificación se utilice como cliente TLS o certificado de servidor.

Para obtener más información acerca de las rutas de certificación, consulte Definición de restricciones de longitud en la ruta de certificación.

Subordinado _ 0/V1 CACertificate PathLen

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 0`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

*Los puntos de distribución de CRL se incluyen en los certificados que se emiten con esta plantilla solo si la CA está configurada con la generación de CRL habilitada.

Definición de subordinado _ 0_ /V1 CACertificate PathLen APICSRPassthrough

Esta plantilla amplía Subordinate CACertificate _ PathLen 0/V1 para admitir los valores de transferencia de API y CSR.

Subordinado _ 0_ /V1 CACertificate PathLen APICSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 0`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 0_ /V1 CACertificate PathLen APIPassthrough

Esta plantilla amplía Subordinate CACertificate _ PathLen 0/V1 para admitir los valores de transferencia de la API.

Subordinado _ 0_ /V1 CACertificate PathLen APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]

Parámetro X509v3	Valor
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 0`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 0_ /V1 CACertificate PathLen CSRPassthrough

Esta plantilla es idéntica a la `SubordinateCACertificate_PathLen0` plantilla con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS transfiere las extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

> **Note**
>
> Se debe crear una CSR que contenga extensiones adicionales personalizadas fuera de Autoridad de certificación privada de AWS.

Subordinado CACertificate _ 0_ /V1 PathLen CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]

Parámetro X509v3	Valor
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 0`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

*Los puntos de distribución de CRL se incluyen en los certificados emitidos con esta plantilla sólo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 1/V1 CACertificate PathLen

Esta plantilla se utiliza para emitir certificados de CA subordinados con una longitud de ruta de. 1 Los certificados de entidad de certificación incluyen una extensión de restricciones básicas críticas con el campo de entidad de certificación establecido en `TRUE` para designar que el certificado se puede utilizar para emitir certificados de CA. No se incluye el uso extendido de claves, lo que impide que el certificado de entidad de certificación se utilice como cliente TLS o certificado de servidor.

Para obtener más información acerca de las rutas de certificación, consulte Definición de restricciones de longitud en la ruta de certificación.

Subordinado _ 1/V1 CACertificate PathLen

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 1`

Parámetro X509v3	Valor
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

*Los puntos de distribución de CRL se incluyen en los certificados emitidos con esta plantilla sólo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado CACertificate _ 1_ /V1 PathLen APICSRPassthrough

Esta plantilla amplía Subordinate CACertificate _ PathLen 1/V1 para admitir los valores de transferencia de API y CSR.

Subordinado _ 1_ /V1 CACertificate PathLen APICSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 1`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 1_ /V1 CACertificate PathLen APIPassthrough

Esta plantilla amplía Subordinate CACertificate _ PathLen 0/V1 para admitir los valores de transferencia de la API.

Subordinado _ 1_ /V1 CACertificate PathLen APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 1`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 1_ /V1 CACertificate PathLen CSRPassthrough

Esta plantilla es idéntica a la `SubordinateCACertificate_PathLen1` plantilla con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS transfiere las extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

> **Note**
>
> Se debe crear una CSR que contenga extensiones adicionales personalizadas fuera de Autoridad de certificación privada de AWS.

Subordinado CACertificate _ 1_ /V1 PathLen CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 1`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

*Los puntos de distribución de CRL se incluyen en los certificados emitidos con esta plantilla sólo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 2/V1 CACertificate PathLen

Esta plantilla se utiliza para emitir certificados de entidad de certificación subordinada con una longitud de ruta de 2. Los certificados de entidad de certificación incluyen una extensión de restricciones básicas críticas con el campo de entidad de certificación establecido en TRUE para designar que el certificado se puede utilizar para emitir certificados de CA. No se incluye el uso extendido de claves, lo que impide que el certificado de entidad de certificación se utilice como cliente TLS o certificado de servidor.

Para obtener más información acerca de las rutas de certificación, consulte Definición de restricciones de longitud en la ruta de certificación.

Subordinado _ 2/V1 CACertificate PathLen

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 2`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

*Los puntos de distribución de CRL se incluyen en los certificados emitidos con esta plantilla sólo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado CACertificate _ 2_ /V1 PathLen APICSRPassthrough

Esta plantilla amplía Subordinate CACertificate _ PathLen 2/V1 para admitir los valores de transferencia de API y CSR.

Subordinado _ 2_ /V1 CACertificate PathLen APICSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 2`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]

Parámetro X509v3	Valor
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 2_ /V1 CACertificate PathLen APIPassthrough

Esta plantilla amplía Subordinate CACertificate _ PathLen 2/V1 para admitir los valores de transferencia de la API.

Subordinado _ 2_ /V1 CACertificate PathLen APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 2`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 2_ /V1 CACertificate PathLen CSRPassthrough

Esta plantilla es idéntica a la `SubordinateCACertificate_PathLen2` plantilla con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS transfiere las extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

> **Note**
>
> Se debe crear una CSR que contenga extensiones adicionales personalizadas fuera de Autoridad de certificación privada de AWS.

Subordinado CACertificate _ 2_ /V1 PathLen CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 2`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

*Los puntos de distribución de CRL se incluyen en los certificados emitidos con esta plantilla sólo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 3/V1 CACertificate PathLen

Esta plantilla se utiliza para emitir certificados de entidad de certificación subordinada con una longitud de ruta de 3. Los certificados de entidad de certificación incluyen una extensión de restricciones básicas críticas con el campo de entidad de certificación establecido en TRUE para designar que el certificado se puede utilizar para emitir certificados de CA. No se incluye el uso extendido de claves, lo que impide que el certificado de entidad de certificación se utilice como cliente TLS o certificado de servidor.

Para obtener más información acerca de las rutas de certificación, consulte Definición de restricciones de longitud en la ruta de certificación.

Subordinado _ 3/V1 CACertificate PathLen

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 3`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

*Los puntos de distribución de CRL se incluyen en los certificados emitidos con esta plantilla sólo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado CACertificate _ 3_ /V1 PathLen APICSRPassthrough

Esta plantilla amplía Subordinate CACertificate _ PathLen 3/V1 para admitir los valores de transferencia de API y CSR.

Subordinado _ 3_ /V1 CACertificate PathLen APICSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 3`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 3_ /V1 CACertificate PathLen APIPassthrough

Esta plantilla amplía Subordinate CACertificate _ PathLen 3/V1 para admitir los valores de transferencia de la API.

Subordinado _ 3_ /V1 CACertificate PathLen APIPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde la API o la CSR]
Asunto	[Acceso directo desde la API o la CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 3`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]

Parámetro X509v3	Valor
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo a través de la configuración de CA]

* Los puntos de distribución de CRL se incluyen en la plantilla solo si la entidad de certificación está configurada con la generación de CRL habilitada.

Definición de subordinado _ 3_ /V1 CACertificate PathLen CSRPassthrough

Esta plantilla es idéntica a la `SubordinateCACertificate_PathLen3` plantilla con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS transfiere las extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no están especificadas en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

> **Note**
>
> Se debe crear una CSR que contenga extensiones adicionales personalizadas fuera de Autoridad de certificación privada de AWS.

Subordinado CACertificate _ 3_ /V1 PathLen CSRPassthrough

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]
Restricciones básicas	Crítico, `CA:TRUE`, `pathlen: 3`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]

Parámetro X509v3	Valor
Uso de clave	Crítico, firma digital, `keyCertSign` , firma de CRL
Punto de distribución de CRL*	[Acceso directo desde la configuración de CA o CSR]

*Los puntos de distribución de CRL se incluyen en los certificados emitidos con esta plantilla sólo si la entidad de certificación está configurada con la generación de CRL habilitada.

Seguridad en AWS Private Certificate Authority

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de los centros de datos y las arquitecturas de red diseñados para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre AWS usted y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede utilizar de forma segura. Los auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los AWS programas de de . Para obtener más información sobre los programas de cumplimiento aplicables AWS Private Certificate Authority, consulte Servicios de AWS Alcance by Compliance Servicios de AWS .
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Autoridad de certificación privada de AWS. Los siguientes temas muestran cómo configurarlo Autoridad de certificación privada de AWS para cumplir sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros Servicios de AWS que le ayuden a supervisar y proteger sus Autoridad de certificación privada de AWS recursos.

Temas

- Identity and Access Management (IAM) para AWS Private Certificate Authority
- Prácticas recomendadas de seguridad para el acceso entre cuentas a redes privadas CAs
- Protección de datos en AWS Private Certificate Authority
- Validación de conformidad para AWS Private Certificate Authority
- Seguridad de la infraestructura en AWS Private Certificate Authority
- AWS Private Certificate Authority Marco CP/CPS del cliente

Identity and Access Management (IAM) para AWS Private Certificate Authority

El acceso a Autoridad de certificación privada de AWS requiere credenciales que AWS puede utilizar para autenticar sus solicitudes. En los siguientes temas se proporciona información detallada sobre cómo puede utilizar AWS Identity and Access Management (IAM) para proteger sus autoridades de certificación privadas (CAs) controlando quién puede acceder a ellas.

En Autoridad de certificación privada de AWS, el recurso principal con el que trabaja es una autoridad de certificación (CA). Todas las entidades de certificación privadas que posea o controle se identifican mediante un nombre de recurso de Amazon (ARN), que tiene el formato siguiente:

```
arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
```

El propietario de un recurso es la entidad principal de la AWS cuenta en la que se crea un AWS recurso. Los siguientes ejemplos ilustran cómo funciona.

- Si utiliza las credenciales de su entidad de certificación Usuario raíz de la cuenta de AWS para crear una entidad de certificación privada, su AWS cuenta es la propietaria de la entidad emisora de certificados.

  > **Important**
  >
  > - No recomendamos utilizar una Usuario raíz de la cuenta de AWS para crear CAs.
  > - Recomendamos encarecidamente el uso de la autenticación multifactor (MFA) cada vez que acceda. Autoridad de certificación privada de AWS

- Si crea un usuario de IAM en su AWS cuenta, puede concederle permiso para crear una CA privada. Sin embargo, la propietaria de la CA será la cuenta a la que pertenece el usuario.
- Si crea una función de IAM en su AWS cuenta y le concede permiso para crear una CA privada, cualquier persona que pueda asumir la función podrá crear la CA. Sin embargo, la propietaria de la CA privada será la cuenta a la que pertenece esa función.

Una política de permisos describe quién tiene acceso a qué. A continuación se explican las opciones disponibles para crear políticas de permisos.

> **Note**
>
> En esta documentación se describe el uso de la IAM en el contexto de. Autoridad de certificación privada de AWS No se proporciona información detallada sobre el servicio de IAM. Para ver la documentación completa de IAM, consulte la Guía del usuario de IAM. Para obtener más información sobre la sintaxis y las descripciones de la política de IAM, consulte AWS Referencia de políticas de IAM.

Autoridad de certificación privada de AWS Operaciones y permisos de la API

Cuando configure el control de acceso y las políticas de permisos que tenga previsto asociar a una identidad de IAM (políticas basadas en identidad), utilice la siguiente tabla como referencia. La primera columna de la tabla muestra cada operación de la Autoridad de certificación privada de AWS API. Usted especifica acciones en un elemento `Action` de la política. El resto de las columnas proporciona información adicional.

Autoridad de certificación privada de AWS Operaciones de API	Permisos necesarios	Recursos
CreateCertificateAuthority	`acm-pca:CreateCertificateAuthority` `acm-pca:TagCertificateAuthority` (Solo se requiere al crear una CA con etiquetas).	`arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566`
CreateCertificateAuthorityAuditReport	`acm-pca:CreateCertificateAuthorityAuditReport`	`arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566`

Autoridad de certificación privada de AWS Operaciones de API	Permisos necesarios	Recursos
CreatePermission	acm-pca:CreatePermission	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DeleteCertificateAuthority	acm-pca:DeleteCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DeletePermission	acm-pca:DeletePermission	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DeletePolicy	acm-pca:DeletePolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

Autoridad de certificación privada de AWS Operaciones de API	Permisos necesarios	Recursos
DescribeCertificateAuthority	acm-pca:DescribeCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DescribeCertificateAuthorityAuditReport	acm-pca:DescribeCertificateAuthorityAuditReport	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
GetCertificate	acm-pca:GetCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
GetCertificateAuthorityCertificate	acm-pca:GetCertificateAuthorityCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

Autoridad de certificación privada de AWS Operaciones de API	Permisos necesarios	Recursos
GetCertificateAuthorityCsr	acm-pca:GetCertificateAuthorityCsr	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
GetPolicy	acm-pca:GetPolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ImportCertificateAuthorityCertificate	acm-pca:ImportCertificateAuthorityCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
IssueCertificate	acm-pca:IssueCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListCertificateAuthorities	acm-pca:ListCertificateAuthorities	N/A

Autoridad de certificación privada de AWS Operaciones de API	Permisos necesarios	Recursos
ListPermissions	acm-pca:ListPermissions	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListTags	acm-pca:ListTags	N/A
PutPolicy	acm-pca:PutPolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
RevokeCertificate	acm-pca:RevokeCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
TagCertificateAuthority	acm-pca:TagCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

Autoridad de certificación privada de AWS Operaciones de API	Permisos necesarios	Recursos
UntagCertificateAuthority	acm-pca:UntagCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
UpdateCertificateAuthority	acm-pca:UpdateCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en AWS IAM Identity Center:

  Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center .

- Usuarios gestionados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.

- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o agregue un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

AWS políticas gestionadas

Autoridad de certificación privada de AWS incluye un conjunto de políticas AWS gestionadas predefinidas para Autoridad de certificación privada de AWS administradores, usuarios y auditores. La comprensión de estas políticas puede ayudarlo a implementar Políticas administradas por el cliente.

Elige cualquiera de las políticas que se indican a continuación para ver los detalles y un ejemplo de código de política.

AWSPrivateCAFullAcceder

Otorga un control administrativo sin restricciones.

Para ver una lista en JSON de los detalles de la política, consulte AWSPrivateCAFullAcceso.

AWSPrivateCAReadSolo

Otorga acceso limitado a las operaciones de la API de solo lectura.

Para ver una lista en JSON de los detalles de la política, consulta AWSPrivateCAReadSolo.

AWSPrivateCAPrivilegedUser

Otorga la capacidad de emitir y revocar certificados de CA. Esta política no tiene otras capacidades administrativas ni capacidad para emitir certificados de entidad final. Los permisos son mutuamente excluyentes con la política Usuario.

Para ver una lista en JSON de los detalles de la política, consulta AWSPrivateCAPrivilegedUsuario.

AWSPrivateCAUser

Otorga la capacidad de emitir y revocar certificados de entidad final. Esta política no tiene capacidades administrativas ni capacidad para emitir certificados de entidad de certificación. Los permisos se excluyen mutuamente de la PrivilegedUserpolítica.

Para ver una lista en JSON de los detalles de la política, consulte AWSPrivateCAUser.

AWSPrivateCAAuditor

Conceda acceso a las operaciones de la API de solo lectura y permiso para generar un informe de auditoría de CA.

Para ver una lista en JSON de los detalles de la política, consulte AWSPrivateCAAuditor.

AWSPrivateCAConnectorForKubernetesPolicy

Otorga los permisos esenciales para el AWS Private CA Conector para Kubernetes.

Para ver una lista en JSON de los detalles de la política, consulte. AWSPrivateCAConnectorForKubernetesPolicy

Actualizaciones de las políticas AWS gestionadas para Autoridad de certificación privada de AWS

En la siguiente tabla, consulte los detalles sobre las actualizaciones de las políticas AWS administradas Autoridad de certificación privada de AWS desde que el servicio comenzó a rastrear estos cambios. Para recibir alertas automáticas sobre todos los cambios que se Autoridad de certificación privada de AWS produzcan, suscríbase a la fuente RSS de la Historial de revisiones del documento página.

Cambios en las políticas administradas

Cambio	Descripción	Fecha
Nueva política: AWS privada CAConnector ForKubern etesPolicy	Se ha introducido una nueva política gestionada para su uso con AWS Private CA Connector for Kubernetes.	19 de mayo de 2025
AWS CAPrivilegedUsuario privado y AWS privadoCA User : política actualizada	`StringLik e ArnLike`Reemplazado por y `StringNotLike` por`ArnNotLike` . Se actualizó la plantilla para incluir comodines `arn:aws:a cm-pca:::template` también. `arn:aws:acm-pca:*:*:template`	22 de enero de 2025
Nuevos nombres de las políticas:	Los prefijos de los nombres de las políticas se cambiaron	13 de febrero de 2023

Cambio	Descripción	Fecha
• `AWS PrivateCA FullAccess` • `AWS PrivateCA ReadOnly` • `AWS PrivateCA PrivilegedUser` • `AWS PrivateCAAuditor` • `AWS PrivateCAUser`	de `AWS CertificateManagerPrivateCA` a `AWS PrivateCA`. La funcionalidad permanece inalterada.	

Políticas administradas por el cliente

Como práctica recomendada, no utilices los tuyos Usuario raíz de la cuenta de AWS para interactuar con ellos AWS, incluso Autoridad de certificación privada de AWS. En su lugar, utilice AWS Identity and Access Management (IAM) para crear un usuario de IAM, un rol de IAM o un usuario federado. Cree un grupo de administradores y añádase usted mismo. Después, inicie sesión como administrador. Puede agregar usuarios adicionales al grupo según sea necesario.

Otra práctica recomendada consiste en crear una política de IAM administrada por el cliente que puede asignar a los usuarios. Las políticas administradas por el cliente son políticas independientes basadas en identidades que usted crea y que puede asociar a varios usuarios, grupos o funciones de la cuenta de AWS . Esta política restringe a los usuarios a realizar solo las acciones de Autoridad de certificación privada de AWS que especifique.

La política administrada por el cliente del siguiente ejemplo permite a un usuario crear un informe de auditoría de la entidad de certificación. Esto es solo un ejemplo. Puede elegir cualquier Autoridad de certificación privada de AWS operación que desee. Para obtener más ejemplos, consulte Políticas insertadas.

Para crear una política administrada por el cliente

1. Inicie sesión en la consola de IAM con las credenciales de un administrador de AWS .
2. En el panel de navegación de la consola, elija Policies (Políticas).
3. Elija Create Policy (Crear política).
4. Seleccione la pestaña JSON.

5. Copie la siguiente política y péguela en el editor.

   JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"acm-pca:CreateCertificateAuthorityAuditReport",
         "Resource":"*"
      }
   ]
}
```

6. Elija Revisar política.
7. En Name (Nombre), escriba `PcaListPolicy`.
8. (Opcional) Ingrese una descripción.
9. Elija Crear política.

Un administrador puede asociar la política a cualquier usuario de IAM para limitar las acciones de Autoridad de certificación privada de AWS que el usuario puede realizar. Para obtener información sobre cómo aplicar una política de permisos, consulte Cambiar los permisos de un usuario de IAM en la Guía del usuario de IAM.

Políticas insertadas

Las políticas insertadas son aquellas que se crean, administran e integran directamente en un usuario, grupo o función. Los siguientes ejemplos de políticas muestran cómo asignar permisos para realizar acciones Autoridad de certificación privada de AWS . Para obtener información general sobre las políticas insertadas, consulte Uso de políticas insertadas en la Guía del usuario de IAM. Puede usar la API Consola de administración de AWS, la AWS Command Line Interface (AWS CLI) o la API de IAM para crear e incrustar políticas integradas.

> **Important**
>
> Recomendamos encarecidamente el uso de la autenticación multifactor (MFA) cada vez que acceda. Autoridad de certificación privada de AWS

Temas

- Anuncio privado CAs
- Recuperar el certificado de una CA privada
- Importar el certificado de una CA privada
- Eliminar una CA privada
- Tag-on-create: Adjuntar etiquetas a una CA en el momento de su creación
- Tag-on-create: Etiquetado restringido
- Controlar el acceso a una CA privada usando etiquetas
- Acceso de solo lectura a Autoridad de certificación privada de AWS
- Acceso completo a Autoridad de certificación privada de AWS

Anuncio privado CAs

La siguiente política permite a un usuario incluir todos los datos privados CAs de una cuenta.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"acm-pca:ListCertificateAuthorities",
         "Resource":"*"
      }
   ]
}
```

Recuperar el certificado de una CA privada

La siguiente política permite al usuario recuperar el certificado de una CA privada.

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:GetCertificateAuthorityCertificate",
      "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID/certificate/certificate_ID"
   }
}
```

Importar el certificado de una CA privada

La siguiente política permite al usuario importar el certificado de una CA privada.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:ImportCertificateAuthorityCertificate",
      "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID/certificate/certificate_ID"
   }
}
```

Eliminar una CA privada

La siguiente política permite al usuario eliminar el certificado de una CA privada.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:DeleteCertificateAuthority",
      "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID/certificate/certificate_ID"   }
}
```

Tag-on-create: Adjuntar etiquetas a una CA en el momento de su creación

La siguiente política permite a los usuarios aplicar etiquetas durante la creación de una CA.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Action": [
            "acm-pca:CreateCertificateAuthority",
            "acm-pca:TagCertificateAuthority"
         ],
         "Effect": "Allow",
         "Resource": "*"
      }
   ]
}
```

Tag-on-create: Etiquetado restringido

La siguiente tag-on-create política impide el uso del par clave-valor Environment=Prod durante la creación de la CA. Se permite etiquetar con otros pares clave-valor.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"acm-pca:*",
         "Resource":"*"
      },
      {
         "Effect":"Deny",
         "Action":"acm-pca:TagCertificateAuthority",
         "Resource":"*",
         "Condition":{
```

```
            "StringEquals":{
               "aws:ResourceTag/Environment":[
                  "Prod"
               ]
            }
         }
      }
   ]
}
```

Controlar el acceso a una CA privada usando etiquetas

La siguiente política permite el acceso únicamente CAs con el par clave-valor Environment=. PreProd También requiere que los nuevos CAs incluyan esta etiqueta.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:*"
         ],
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "aws:ResourceTag/Environment":[
                  "PreProd"
               ]
            }
         }
      }
   ]
}
```

Acceso de solo lectura a Autoridad de certificación privada de AWS

La siguiente política permite al usuario describir y mostrar las entidades de certificación privadas y recuperar el certificado de entidad de certificación privada y la cadena de certificados.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":{
       "Effect":"Allow",
       "Action":[
          "acm-pca:DescribeCertificateAuthority",
          "acm-pca:DescribeCertificateAuthorityAuditReport",
          "acm-pca:ListCertificateAuthorities",
          "acm-pca:ListTags",
          "acm-pca:GetCertificateAuthorityCertificate",
          "acm-pca:GetCertificateAuthorityCsr",
          "acm-pca:GetCertificate"
       ],
       "Resource":"*"
    }
}
```

Acceso completo a Autoridad de certificación privada de AWS

La siguiente política permite a un usuario realizar cualquier Autoridad de certificación privada de AWS acción.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:*"
         ],
         "Resource":"*"
```

```
        }
    ]
}
```

Prácticas recomendadas de seguridad para el acceso entre cuentas a redes privadas CAs

Un Autoridad de certificación privada de AWS administrador puede compartir una CA con los directores (usuarios, funciones, etc.) de otra Cuenta de AWS. Cuando se ha recibido y aceptado una participación, el principal puede usar la CA para emitir certificados de entidad final utilizando nuestros recursos Autoridad de certificación privada de AWS . AWS Certificate Manager El principal puede usar la CA para emitir certificados de CA subordinados utilizando. Autoridad de certificación privada de AWS

> **Important**
>
> Los cargos asociados a un certificado emitido en un escenario de cuentas múltiples se facturan a la AWS cuenta que emite el certificado.

Para compartir el acceso a una CA, Autoridad de certificación privada de AWS los administradores pueden elegir uno de los siguientes métodos:

- Utilice AWS Resource Access Manager (RAM) para compartir la CA como recurso con una entidad principal de otra cuenta o con ella AWS Organizations. La RAM es un método estándar para compartir AWS recursos entre cuentas. Para obtener más información sobre la RAM, consulte la Guía del usuario de AWS RAM. Para obtener más información sobre AWS Organizations, consulte la Guía del usuario de AWS Organizations.
- Utilice la Autoridad de certificación privada de AWS API o la CLI para adjuntar una política basada en recursos a una CA y, de este modo, conceder acceso a un principal de otra cuenta. Para obtener más información, consulte Políticas basadas en recursos.

La Controle el acceso a la CA privada sección de esta guía proporciona flujos de trabajo para conceder el acceso tanto CAs en situaciones de una sola cuenta como de varias cuentas.

Políticas basadas en recursos

Las políticas basadas en recursos son políticas de permisos que se crean y asocian manualmente a un recurso (en este caso, una CA privada) en lugar de a una identidad o un rol de un usuario. O bien, en lugar de crear sus propias políticas, puede utilizar políticas AWS gestionadas para. AWS Private CA Al AWS RAM aplicar una política basada en recursos, un Autoridad de certificación privada de AWS administrador puede compartir el acceso a una CA con un usuario de una AWS cuenta diferente directamente o a través de ella. AWS Organizations Como alternativa, un Autoridad de certificación privada de AWS administrador puede usar la PCA APIs PutPolicyGetPolicy, y o los AWS CLI comandos correspondientes put-policy DeletePolicy, get-policy y delete-policy para aplicar y administrar políticas basadas en recursos.

Para obtener más información acerca de las políticas de acceso basadas en recursos, consulte Políticas basadas en identidad y Políticas basadas en recursos y Control del acceso usando políticas.

Para ver la lista de políticas basadas en recursos AWS gestionados AWS Private CA, vaya a la biblioteca de permisos gestionados de la consola y busque. AWS Resource Access ManagerCertificateAuthority Como ocurre con cualquier política, antes de aplicarla, le recomendamos que la aplique en un entorno de prueba para asegurarse de que cumple sus requisitos.

AWS Certificate Manager Los usuarios (ACM) con acceso compartido entre cuentas a una CA privada pueden emitir certificados gestionados firmados por la CA. Los emisores multicuentas están limitados por una política basada en los recursos y solo tienen acceso a las siguientes plantillas de certificados de entidad final:

- EndEntityCertificate/V1
- EndEntityClientAuthCertificate/V1
- EndEntityServerAuthCertificate/V1
- BlankEndEntityCertificate_ /V1 APIPassthrough
- BlankEndEntityCertificate_ /V1 APICSRPassthrough
- Subordinado _ 0/V1 CACertificate PathLen

Ejemplos de políticas

En esta sección se proporcionan ejemplos de políticas multicuentas para diversas necesidades. En todos los casos, se utiliza el siguiente patrón de comandos para aplicar una política:

```
$ aws acm-pca put-policy \
   --region region \
   --resource-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
   --policy file:///[path]/policyN.json
```

Además de especificar el ARN de una CA, el administrador proporciona un identificador de AWS cuenta o un AWS Organizations identificador al que se concederá acceso a la entidad emisora de certificados. El JSON de cada una de las siguientes políticas tiene el formato de un archivo para facilitar la lectura, pero también se puede proporcionar como argumentos CLI en línea.

> **Note**
>
> Debe seguirse con precisión la estructura de las políticas basadas en recursos de JSON que se muestra a continuación. Los clientes solo ARNs pueden configurar los campos de ID de los principales (el número de AWS cuenta o el ID de la AWS organización) y la CA.

1. Archivo: policy1.json — Compartir el acceso a una CA con un usuario de una cuenta diferente

   *555555555555*Sustitúyalos por el ID de AWS cuenta con el que se comparte la CA.

   Para el ARN del recurso, sustituya lo siguiente por sus propios valores:

   - *aws*- La AWS partición. Por ejemplo, `aws`aws-us-gov,aws-cn,, etc.
   - *us-east-1*- La AWS región en la que está disponible el recurso, por ejemplo`us-west-1`.
   - *111122223333*- El ID de AWS cuenta del propietario del recurso.
   - *11223344-1234-1122-2233-112233445566*- El ID de recurso de la autoridad de certificación.

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [{
               "Sid": "ExampleStatementID",
               "Effect": "Allow",
               "Principal": {
                   "AWS": "555555555555"
               },
   ```

```
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListPermissions",
                "acm-pca:ListTags"
            ],
            "Resource": "arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID"
        },
        {
            "Sid": "ExampleStatementID2",
            "Effect": "Allow",
            "Principal": {
                "AWS": "555555555555"
            },
            "Action": [
                "acm-pca:IssueCertificate"
            ],
            "Resource": "arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID",
            "Condition": {
                "StringEquals": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
EndEntityCertificate/V1"
                }
            }
        }
    ]
}
```

2. Archivo: policy2.json — Compartir el acceso a una CA mediante AWS Organizations

   *o-a1b2c3d4z5*Sustitúyalo por el ID AWS Organizations .

   Para el ARN del recurso, sustituya lo siguiente por sus propios valores:

   - *aws*- La AWS partición. Por ejemplo, `aws`aws-us-gov,aws-cn,, etc.
   - *us-east-1*- La AWS región en la que está disponible el recurso, por ejemplo`us-west-1`.
   - *111122223333*- El ID de AWS cuenta del propietario del recurso.
   - *11223344-1234-1122-2233-112233445566*- El ID de recurso de la autoridad de certificación.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleStatementID3",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "acm-pca:IssueCertificate",
            "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID",
            "Condition": {
                "StringEquals": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
EndEntityCertificate/V1",
                    "aws:PrincipalOrgID": "o-a1b2c3d4z5"
                },
                "StringNotEquals": {
                    "aws:PrincipalAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "ExampleStatementID4",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListPermissions",
                "acm-pca:ListTags"
            ],
            "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "o-a1b2c3d4z5"
                },
                "StringNotEquals": {
                    "aws:PrincipalAccount": "111122223333"
                }
```

```
            }
        }
    ]
}
```

Protección de datos en AWS Private Certificate Authority

El modelo de responsabilidad AWS compartida modelo se aplica a la protección de datos en AWS Private Certificate Authority. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog sobre el Modelo de responsabilidad compartida de AWS y GDPR en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con AWS los recursos. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Cómo trabajar con CloudTrail senderos en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger la información confidencial almacenada en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información

sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con Autoridad de certificación privada de AWS o Servicios de AWS utiliza la consola, la API o. AWS CLI AWS SDKs Cualquier dato que introduzca en etiquetas o campos de formato libre utilizados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Cumplimiento de las normas de seguridad y almacenamiento de las claves Autoridad de certificación privada de AWS privadas

Las claves privadas de private CAs se almacenan en módulos de seguridad de hardware AWS gestionados (HSMs). HSMs Cumplen con los requisitos de seguridad de nivel 3 de la norma FIPS PUB 140-2 para los módulos criptográficos.

Cifrado de datos en AWS Private CA Connector for Active Directory

AWS Private CA Connector para AD almacena los datos de configuración del cliente relacionados con los conectores, las plantillas, los registros de directorios, los nombres principales de los servicios y las entradas de control de acceso de los grupos de plantillas. Los datos se cifran en tránsito y en reposo. La información sobre los certificados emitidos a través de Connector for AD se puede encontrar mediante la GetCertificateacción de la AWS Private CA API. No almacena información sobre los certificados emitidos ni sobre el cliente o la máquina que solicita un certificado AWS.

Validación de conformidad para AWS Private Certificate Authority

Los auditores externos evalúan la seguridad y el cumplimiento AWS Private Certificate Authority como parte de varios programas de AWS cumplimiento. Estos incluyen SOC, PCI, FedRAMP, HIPAA y otros.

Para obtener una lista de AWS los servicios incluidos en el ámbito de los programas de cumplimiento específicos, consulte AWS Servicios incluidos Servicios de AWS en el . Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Autoridad de certificación privada de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- En el caso de las organizaciones que necesitan cifrar sus buckets de Amazon S3, los temas siguientes describen cómo configurar el cifrado para acomodar los activos de Autoridad de certificación privada de AWS :
  - Cifrado de los informes de auditoría
  - Cifrando su CRLs
- Guías de inicio rápido de seguridad y conformidad : estas guías de implementación analizan consideraciones sobre arquitectura y proporcionan los pasos para implementar los entornos de referencia centrados en la seguridad y la conformidad en AWS.
- Documento técnico sobre cómo diseñar una arquitectura basada en la seguridad y el cumplimiento de la HIPAA: en este documento técnico se describe cómo pueden utilizar las empresas para crear aplicaciones que cumplan con la HIPAA. AWS
- AWS Recursos de cumplimiento Recursos de de trabajo y guías puede aplicarse a su sector y ubicación.
- Evaluación de los recursos con las reglas de la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub CSPM— Este AWS servicio proporciona una visión integral del estado de su seguridad AWS que le ayuda a comprobar su conformidad con los estándares y las mejores prácticas del sector de la seguridad.

Utilice los informes de auditoría con su CA privada

Puede crear un informe de auditoría para mostrar todos los certificados que la CA privada ha emitido o revocado. El informe se guarda en un bucket de S3 nuevo o existente que se especifique en la entrada.

Para obtener información sobre cómo agregar protección de cifrado a los informes de auditoría, consulte Cifrado de los informes de auditoría.

El archivo del informe de auditoría tiene la siguiente ruta y nombre de archivo. El ARN de un bucket de Amazon S3 es el valor de `amzn-s3-demo-bucket`. `CA_ID` es el identificador único de una CA emisora. `UUID` es el identificador único de un informe de auditoría.

```
amzn-s3-demo-bucket/audit-report/CA_ID/UUID.[json|csv]
```

Puede generar un nuevo informe cada 30 minutos y descargarlo en el bucket. En el ejemplo siguiente se muestra un informe CSV.

```
awsAccountId,requestedByServicePrincipal,certificateArn,serial,subject,notBefore,notAfter,issue
123456789012,,arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/
certificate_ID,00:11:22:33:44:55:66:77:88:99:aa:bb:cc:dd:ee:ff,"2.5.4.5=#012345678901,2.5.4.44=
 Company,L=Seattle,ST=Washington,C=US",2020-03-02T21:43:57+0000,2020-04-07T22:43:57+0000,2020-0
pca:::template/EndEntityCertificate/V1
123456789012,acm.amazonaws.com,arn:aws:acm-pca:region:account:certificate-
authority/CA_ID/
certificate/
certificate_ID,ff:ee:dd:cc:bb:aa:99:88:77:66:55:44:33:22:11:00,"2.5.4.5=#012345678901,2.5.4.44=
 Company,L=Seattle,ST=Washington,C=US",2020-03-02T20:53:39+0000,2020-04-07T21:53:39+0000,2020-0
pca:::template/EndEntityCertificate/V1
```

En el ejemplo siguiente, se muestra un informe en formato JSON.

```
[
   {
      "awsAccountId":"123456789012",
      "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
      "serial":"00:11:22:33:44:55:66:77:88:99:aa:bb:cc:dd:ee:ff",

 "subject":"2.5.4.5=#012345678901,2.5.4.44=#0a1b3c4d,2.5.4.65=#0a1b3c4d5e6f,2.5.4.43=#0a1b3c4d5
 Company,L=Seattle,ST=Washington,C=US",
      "notBefore":"2020-02-26T18:39:57+0000",
      "notAfter":"2021-02-26T19:39:57+0000",
      "issuedAt":"2020-02-26T19:39:58+0000",
      "revokedAt":"2020-02-26T20:00:36+0000",
      "revocationReason":"UNSPECIFIED",
      "templateArn":"arn:aws:acm-pca:::template/EndEntityCertificate/V1"
   },
   {
      "awsAccountId":"123456789012",
```

```
      "requestedByServicePrincipal":"acm.amazonaws.com",
      "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
      "serial":"ff:ee:dd:cc:bb:aa:99:88:77:66:55:44:33:22:11:00",

 "subject":"2.5.4.5=#012345678901,2.5.4.44=#0a1b3c4d,2.5.4.65=#0a1b3c4d5e6f,2.5.4.43=#0a1b3c4d5
 Company,L=Seattle,ST=Washington,C=US",
      "notBefore":"2020-01-22T20:10:49+0000",
      "notAfter":"2021-01-17T21:10:49+0000",
      "issuedAt":"2020-01-22T21:10:49+0000",
      "templateArn":"arn:aws:acm-pca:::template/EndEntityCertificate/V1"
   }
]
```

> **Note**
>
> Al AWS Certificate Manager renovar un certificado, el informe de auditoría privado de CA rellena el `requestedByServicePrincipal` campo con. `acm.amazonaws.com` Esto indica que el AWS Certificate Manager servicio solicitó a `IssueCertificate` la Autoridad de certificación privada de AWS API la renovación del certificado en nombre de un cliente.

Prepare un bucket de Amazon S3 para los informes de auditoría

> **Important**
>
> AWS Private CA no admite el uso de Amazon S3 Object Lock. Si habilitas Object Lock en tu bucket, AWS Private CA no podrá escribir informes de auditoría en el bucket.

Para almacenar sus informes de auditoría, debe preparar un bucket de Amazon S3. Para obtener más información, consulte ¿Cómo puedo crear un bucket de S3?

El bucket de S3 debe estar protegido por una política de permisos que permita acceder AWS Private CA al bucket de S3 que especifique y escribir en él. Los usuarios autorizados y los directores de servicio necesitan `Put` permiso Autoridad de certificación privada de AWS para poder colocar objetos en el depósito y `Get` para recuperarlos. Le recomendamos que aplique la política que se muestra a continuación, que restringe el acceso tanto a una AWS cuenta como al ARN de una CA privada. Como alternativa, puede usar la clave de condición aws: SourceOrg ID para restringir el acceso a

una organización específica. AWS Organizations Para obtener más información sobre las políticas de depósitos, consulte Políticas de depósitos de Amazon Simple Storage Service.

Cree un informe de auditoría

Puede crear un informe de auditoría desde la consola o la AWS CLI.

Para crear un informe de auditoría (consola)

1. Inicia sesión en tu AWS cuenta y abre la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/casa.
2. En la página Private certificate authorities (Entidades de certificación privada), elija una CA privada de la lista.
3. En el menú Acciones, elija Generar informe de auditoría.
4. En Audit report destination (Destino de informe de auditoría), para Create a new S3 bucket? ¿Crear un nuevo bucket de S3?, elija Yes (Sí) y escriba un nombre de bucket único o elija No y elija un bucket existente de la lista.

   Si eliges Sí, Autoridad de certificación privada de AWS crea y adjunta la política predeterminada a tu bucket. La política predeterminada incluye una clave de `aws:SourceAccount` condición que limita el acceso a una AWS cuenta específica. Si desea restringir aún más el acceso, puede añadir otras claves de condición a la política, como en el ejemplo anterior.

   Si selecciona No, deberá asociar la siguiente política al bucket antes de poder generar el informe de auditoría. Utilice el patrón de políticas descrito en Prepare un bucket de Amazon S3 para los informes de auditoría. Para obtener información sobre cómo adjuntar una política, consulte Añadir una política de bucket mediante la consola Amazon S3.
5. En Formato de salida, elija JSON para la notación de JavaScript objetos o CSV para valores separados por comas.
6. Seleccione Generate audit report (Generar informe de auditoría).

Para crear un informe de auditoría (AWS CLI)

1. Si no dispone de un bucket S3, cree uno.
2. Adjunte la política a su bucket. Utilice el patrón de políticas descrito en Prepare un bucket de Amazon S3 para los informes de auditoría. Para obtener más información sobre cómo adjuntar una política, consulte Agregar una política de bucket mediante la consola de Amazon S3

3. Utilice el comando create-certificate-authority-audit-report para crear el informe de auditoría y colocarlo en el depósito de S3 preparado.

```
$ aws acm-pca create-certificate-authority-audit-report \
--certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
--s3-bucket-name amzn-s3-demo-bucket \
--audit-report-response-format JSON
```

Recupere un informe de auditoría

Para obtener un informe de auditoría para su inspección, utilice la consola de Amazon S3, la API, la CLI o el SDK. Para obtener más información, consulte Descargar un objeto en la Guía del usuario de Amazon Simple Storage Service.

Cifrado de los informes de auditoría

Si lo desea, puede configurar el cifrado en el bucket de Amazon S3 que contiene sus informes de auditoría. Autoridad de certificación privada de AWS admite dos modos de cifrado para los activos de S3:

- Cifrado del lado del servidor automático con claves AES-256 administradas por Amazon S3.
- El cifrado gestionado por el cliente utiliza AWS Key Management Service y AWS KMS key configura según sus especificaciones.

> **Note**
>
> Autoridad de certificación privada de AWS no admite el uso de claves KMS predeterminadas generadas automáticamente por S3.

Los siguientes procedimientos describen cómo configurar cada una de las opciones de cifrado.

Para configurar el cifrado automático

Complete los siguientes pasos para habilitar el cifrado del lado del servidor de S3.

1. Abra la consola de Amazon S3 en https://console.aws.amazon.com/s3/.

2. En la tabla Buckets, elija el bucket que contendrá sus Autoridad de certificación privada de AWS activos.
3. En la página del bucket, elija la pestaña Properties (Propiedades).
4. Elija la tarjeta Default encryption (Cifrado predeterminado) .
5. Seleccione Habilitar.
6. Elija la Amazon S3 key (SSE-S3) (Clave Amazon S3 (SSE-S3)).
7. Elija Save changes (Guardar cambios).

Para configurar el cifrado personalizado

Complete los siguientes pasos para habilitar el cifrado mediante una clave personalizada.

1. Abra la consola de Amazon S3 en https://console.aws.amazon.com/s3/.
2. En la tabla de cubos, elige el grupo en el que se guardarán tus Autoridad de certificación privada de AWS activos.
3. En la página del bucket, elija la pestaña Properties (Propiedades).
4. Elija la tarjeta Default encryption (Cifrado predeterminado) .
5. Seleccione Habilitar.
6. Elija la AWS Key Management Service clave (SSE-KMS).
7. Elige entre tus AWS KMS claves o Ingresa el AWS KMS key ARN.
8. Elija Save changes (Guardar cambios).
9. (Opcional) Si aún no tiene una clave de KMS, cree una con el siguiente comando create-key de AWS CLI :

   ```
   $ aws kms create-key
   ```

   La salida contiene la ID de clave y el nombre de recurso de Amazon (ARN) de la clave de KMS. El siguiente es un ejemplo de salida:

   ```
   {
       "KeyMetadata": {
           "KeyId": "01234567-89ab-cdef-0123-456789abcdef",
           "Description": "",
           "Enabled": true,
           "KeyUsage": "ENCRYPT_DECRYPT",
   ```

```
        "KeyState": "Enabled",
        "CreationDate": 1478910250.94,
        "Arn": "arn:aws:kms:us-west-2:123456789012:key/01234567-89ab-
cdef-0123-456789abcdef",
        "AWSAccountId": "123456789012"
    }
}
```

10. Mediante los siguientes pasos, usted otorga al principal del Autoridad de certificación privada de AWS servicio permiso para usar la clave KMS. De forma predeterminada, todas las claves de KMS son privadas; solo el propietario del recurso puede utilizarlo para cifrar y descifrar datos. Sin embargo, el propietario del recurso puede conceder permisos para que otros usuarios y recursos accedan a la clave de KMS. Esta entidad principal del servicio debe estar en la misma región en la que está almacenada la clave de KMS.

    a. En primer lugar, guarde la política predeterminada para su clave KMS `policy.json` mediante el siguiente get-key-policycomando:

    ```
    $ aws kms get-key-policy --key-id key-id --policy-name default --output text
     > ./policy.json
    ```

    b. Abra el archivo `policy.json` en un editor de texto. Seleccione una de las siguientes declaraciones de política y agréguela a la política existente.

    Si su clave de bucket de Amazon S3 está habilitada, utilice la siguiente declaración:

    ```
    {
       "Sid":"Allow ACM-PCA use of the key",
       "Effect":"Allow",
       "Principal":{
          "Service":"acm-pca.amazonaws.com"
       },
       "Action":[
          "kms:GenerateDataKey",
          "kms:Decrypt"
       ],
       "Resource":"*",
       "Condition":{
          "StringLike":{
             "kms:EncryptionContext:aws:s3:arn":"arn:aws:s3:::bucket-name"
          }
       }
    ```

```
}
```

Si su clave de bucket de Amazon S3 está deshabilitada, utilice la siguiente declaración:

```
{
   "Sid":"Allow ACM-PCA use of the key",
   "Effect":"Allow",
   "Principal":{
      "Service":"acm-pca.amazonaws.com"
   },
   "Action":[
      "kms:GenerateDataKey",
      "kms:Decrypt"
   ],
   "Resource":"*",
   "Condition":{
      "StringLike":{
         "kms:EncryptionContext:aws:s3:arn":[
            "arn:aws:s3:::bucket-name/acm-pca-permission-test-key",
            "arn:aws:s3:::bucket-name/acm-pca-permission-test-key-private",
            "arn:aws:s3:::bucket-name/audit-report/*",
            "arn:aws:s3:::bucket-name/crl/*"
         ]
      }
   }
}
```

c. Por último, aplique la política actualizada mediante el siguiente put-key-policycomando:

```
$ aws kms put-key-policy --key-id key_id --policy-name default --policy file://
policy.json
```

Seguridad de la infraestructura en AWS Private Certificate Authority

Como servicio gestionado, AWS Private Certificate Authority está protegido por la seguridad de la red AWS global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte Seguridad AWS en la nube. Para diseñar su AWS entorno utilizando las mejores prácticas de seguridad de la infraestructura, consulte Protección de infraestructuras en un marco de buena AWS arquitectura basado en el pilar de la seguridad.

Utiliza las llamadas a la API AWS publicadas para acceder a AWS Private CA través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Autoridad de certificación privada de AWS Puntos finales de VPC ()AWS PrivateLink

Puede crear una conexión privada entre su VPC y configurar un punto final de Autoridad de certificación privada de AWS la VPC de interfaz. Los puntos finales de la interfaz funcionan con una tecnología para acceder de forma privada a Autoridad de certificación privada de AWS las operaciones de la API. AWS PrivateLink AWS PrivateLink enruta todo el tráfico de red entre su VPC y Autoridad de certificación privada de AWS a través de la red de Amazon, lo que evita la exposición en Internet abierta. Cada punto de enlace de la VPC está representado por una o varias interfaces de red elástica con direcciones IP privadas en las subredes de la VPC.

El punto final de la interfaz de la VPC conecta su VPC directamente Autoridad de certificación privada de AWS sin una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o una conexión. Direct Connect Las instancias de la VPC no necesitan direcciones IP públicas para comunicarse con la API de Autoridad de certificación privada de AWS .

Para usarlo Autoridad de certificación privada de AWS a través de la VPC, debe conectarse desde una instancia que esté dentro de la VPC. Como alternativa, puede conectar la red privada a la VPC mediante un AWS Virtual Private Network (Site-to-Site VPN) o. Direct Connect Para obtener más información Site-to-Site VPN, consulte Conexiones VPN en la Guía del usuario de Amazon VPC. Para obtener información acerca de Direct Connect, consulte Creación de una conexión en la Guía del usuario de Direct Connect .

Autoridad de certificación privada de AWS no requiere su uso AWS PrivateLink, pero lo recomendamos como capa de seguridad adicional. Para obtener más información sobre AWS PrivateLink los puntos de enlace de la VPC, consulte Acceder a los servicios mediante. AWS PrivateLink

Consideraciones sobre los puntos Autoridad de certificación privada de AWS finales de VPC

Antes de configurar los puntos finales de la VPC de la interfaz Autoridad de certificación privada de AWS, tenga en cuenta las siguientes consideraciones:

- Autoridad de certificación privada de AWS es posible que no sea compatible con los puntos de conexión de VPC en algunas zonas de disponibilidad. Cuando cree un punto de conexión de VPC, compruebe primero el soporte en la consola de administración. Las zonas de disponibilidad no compatibles aparecen marcadas como “Service not supported in this Availability Zone” (El servicio no es compatible en esta zona de disponibilidad).
- Los puntos de conexión de VPC no admiten las solicitudes entre regiones. Asegúrese de crear su punto de conexión en la misma región en la que tiene previsto enviar llamadas a la API de Autoridad de certificación privada de AWS.
- Los puntos de conexión de VPC solo admiten DNS proporcionadas por Amazon a través de Amazon Route 53. Si desea utilizar su propio DNS, puede utilizar el enrutamiento de DNS condicional. Para obtener más información, consulte Conjuntos de opciones de DHCP en la Guía del usuario de Amazon VPC.
- El grupo de seguridad asociado al punto de conexión de la VPC debe permitir las conexiones entrantes en el puerto 443 desde la subred privada de la VPC.

Autoridad de certificación privada de AWS Actualmente, la API admite los puntos finales de VPC en los siguientes aspectos: Regiones de AWS

- Este de EE. UU. (Ohio)
- Este de EE. UU. (Norte de Virginia)
- Oeste de EE. UU. (Norte de California)
- Oeste de EE. UU. (Oregón)
- África (Ciudad del Cabo)
- Asia-Pacífico (Hong Kong)
- Asia-Pacífico (Hyderabad)
- Asia-Pacífico (Yakarta)
- Asia-Pacífico (Melbourne)
- Asia-Pacífico (Mumbai)

- Asia-Pacífico (Osaka)
- Asia-Pacífico (Seúl)
- Asia-Pacífico (Singapur)
- Asia-Pacífico (Sídney)
- Asia-Pacífico (Tokio)
- Canadá (centro)
- Oeste de Canadá (Calgary)
- Europa (Fráncfort)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Milán)
- Europa (París)
- Europa (España)
- Europa (Estocolmo)
- Europa (Zúrich)
- Israel (Tel Aviv)
- Middle East (Bahrain)
- Medio Oriente (EAU)
- América del Sur (São Paulo)

Creación de los puntos finales de VPC para Autoridad de certificación privada de AWS

Puede crear un punto de enlace de VPC para el Autoridad de certificación privada de AWS servicio mediante la consola de VPC en o el. https://console.aws.amazon.com/vpc/ AWS Command Line Interface Para obtener más información, consulte el procedimiento de creación de un punto final de interfaz en la Guía del usuario de Amazon VPC. Autoridad de certificación privada de AWS admite realizar llamadas a todas sus operaciones de API dentro de su VPC.

Si ha habilitado nombres de host de DNS privados para el punto de conexión, el punto de conexión Autoridad de certificación privada de AWS predeterminado se resuelve ahora en el punto de conexión de VPC. Para obtener una lista completa de los puntos de conexión de servicio predeterminados, consulte Puntos de enlace de servicio y cuotas.

Si no ha habilitado nombres de host de DNS privados, Amazon VPC proporciona un nombre de punto de conexión de DNS que puede utilizar en el siguiente formato:

```
vpc-endpoint-id.acm-pca.region.vpce.amazonaws.com
```

> **Note**
>
> El valor *region* representa el identificador de región de una AWS región admitida por Autoridad de certificación privada de AWS, por ejemplo, la región EE.UU. Este (Ohio). `us-east-2` Para obtener una lista de Autoridad de certificación privada de AWS, consulte Puntos de conexión y cuotas de la autoridad de certificación privada de AWS.

Para obtener más información, consulte los puntos de enlace de la Autoridad de certificación privada de AWS VPC (AWS PrivateLink) en la Guía del usuario de Amazon VPC.

Cree una política de puntos de conexión de VPC para Autoridad de certificación privada de AWS

Puede crear una política para los puntos de enlace de Amazon VPC Autoridad de certificación privada de AWS para especificar lo siguiente:

- La entidad principal que puede realizar acciones
- Las acciones que se pueden realizar
- Los recursos en los que se pueden llevar a cabo las acciones

Para obtener más información, consulte Control del acceso a los servicios con puntos de enlace de VPC en la Guía de Amazon VPC.

Ejemplo: política de puntos finales de VPC para acciones Autoridad de certificación privada de AWS

Cuando se conecta a un punto final, la siguiente política otorga acceso a todos los principales a las Autoridad de certificación privada de AWS accionesIssueCertificate,,DescribeCertificateAuthority, GetCertificate GetCertificateAuthorityCertificateListPermissions, y. ListTags El recurso en cada estrofa es una entidad de certificación privada. La primera estrofa autoriza la creación de certificados de entidad final utilizando la entidad de certificación privada y la plantilla de certificado especificados.

Si no desea controlar la plantilla que se está utilizando, la sección `Condition` no es necesaria. Sin embargo, al eliminar esto, todas las entidades principales pueden crear certificados de entidad de certificación, así como certificados de entidad final.

```
{
      "Statement":[
         {
            "Principal":"*",
            "Effect":"Allow",
            "Action":[
               "acm-pca:IssueCertificate"
            ],
            "Resource":[
               "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
            ],
            "Condition":{
               "StringEquals":{
                  "acm-pca:TemplateArn":"arn:aws:acm-pca:::template/
EndEntityCertificate/V1"
               }
            }
         },
         {
            "Principal":"*",
            "Effect":"Allow",
            "Action":[
               "acm-pca:DescribeCertificateAuthority",
               "acm-pca:GetCertificate",
               "acm-pca:GetCertificateAuthorityCertificate",
               "acm-pca:ListPermissions",
               "acm-pca:ListTags"
            ],
            "Resource":[
               "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
            ]
         }
      ]
   }
```

Compatibilidad con puntos de conexión de doble pila

AWS Private Certificate Authority proporciona un punto final público de doble pila que admite tanto a los clientes como a IPv4 los IPv6 clientes. Un punto final de doble pila permite a los clientes comunicarse AWS Private CA con cualquiera IPv4 de las direcciones. IPv6 AWS Private CA para Active Directory y AWS Private CA Connector for SCEP también admiten terminales de doble pila.

El terminal público AWS Private CA de doble pila `https://acm-pca.your-region.api.aws` es compatible con ambos clientes. IPv4 IPv6 AWS Private CA también es accesible de forma privada a través IPv4 y IPv6 desde su nube privada virtual (VPC) mediante. AWS PrivateLink Para obtener más información sobre la creación de puntos de enlace de VPC de interfaz privada para AWS Private CA, consulte. Autoridad de certificación privada de AWS Puntos finales de VPC ()AWS PrivateLink

Para obtener más información, consulte los siguientes recursos:

- Direccionamiento IP para sus VPCs subredes y
- IPv6 soporte para su VPC

Uso de IPv6 direcciones en IAM y AWS Private CA

Antes de intentar acceder IPv6, asegúrese AWS Private Certificate Authority de que todas las políticas de IAM que contengan restricciones de direcciones IP estén actualizadas para incluir los rangos de IPv6 direcciones. Las políticas basadas en IP que no se actualizan para gestionar IPv6 las direcciones pueden provocar que los clientes pierdan o obtengan el acceso de forma incorrecta cuando empiecen a IPv6 utilizarlas. Para obtener más información sobre AWS Private CA la compatibilidad con doble pila, consulteCompatibilidad con puntos de conexión de doble pila.

> **Important**
>
> Estas declaraciones no permiten ninguna acción. Utilice estas declaraciones en combinación con otras declaraciones que permiten acciones específicas.

La siguiente declaración deniega explícitamente el acceso a todos los AWS Private CA permisos a las solicitudes que se originen en el `192.0.2.*` rango de IPv4 direcciones. A las direcciones IP que se encuentren fuera de este rango no se les deniegan AWS Private CA los permisos de forma explícita. Como todas IPv6 las direcciones están fuera del rango denegado, esta declaración no deniega explícitamente AWS Private CA los permisos para ninguna IPv6 dirección.

```
{
    "Sid": "DenyPrivateCAPermissions",
    "Effect": "Deny",
    "Action": [
        "acm-pca:*"
    ],
    "Resource": "*",
    "Condition": {
        "NotIpAddress": {
            "aws:SourceIp": [
                "192.0.2.0/24"
            ]
        }
    }
}
```

Puede modificar el `Condition` elemento para denegar los rangos de direcciones IPv4 (`192.0.2.0/24`) y IPv6 (`2001:db8::/32`), como se muestra en el siguiente ejemplo:

```
{
    "Sid": "DenyPrivateCAPermissions",
    "Effect": "Deny",
    "Action": [
        "acm-pca:*"
    ],
    "Resource": "*",
    "Condition": {
        "NotIpAddress": {
            "aws:SourceIp": [
                "192.0.2.0/24",
                "2001:db8::/32"
            ]
        }
    }
}
```

AWS Private Certificate Authority Marco CP/CPS del cliente

AWS Private Certificate Authority proporciona servicios de infraestructura que permiten crear jerarquías de autoridades de certificación (CA), incluidas las principales y las subordinadas CAs, sin los costos de inversión y mantenimiento que implica operar una CA local. Al AWS Private CA crear sus jerarquías de CA, existe una responsabilidad compartida entre usted y. AWS Private CA El modelo de responsabilidad compartida puede ayudarle a aliviar su carga operativa al AWS operar, gestionar y controlar la seguridad física de las instalaciones en las que opera el servicio. Usted asume la responsabilidad y la gestión de la autoridad de certificación (incluida la creación y eliminación de los recursos de CA; la distribución de las bases de confianza; la creación de jerarquías entre las PKI; las políticas y prácticas de certificación; la configuración para permitir o denegar el intercambio de CA Cuentas de AWS; las políticas de uso de plantillas; las auditorías; los controles de acceso, incluida la separación de funciones; y otras configuraciones y políticas de CA). Debe considerar detenidamente los servicios que elija, ya que sus responsabilidades varían según los servicios utilizados, la integración de esos servicios en su entorno de TI y las leyes y reglamentos aplicables. Para obtener más información, consulte el modelo de responsabilidad compartida en materia de Nube de AWS seguridad.

Crear una política de certificación (CP) o una declaración de prácticas de certificación (CPS) para su entidad de certificación privada es una parte fundamental de la administración de su infraestructura de clave pública (PKI). Un CP define todos los requisitos y reglas de su PKI y el CPS explica cómo se cumplen los requisitos del CP. Usted es responsable de crear un CP y un CPS como autoridad de certificación de su PKI. AWS Private CA le proporciona documentación de AWS control y cumplimiento, como el informe de controles de AWS sistemas y organizaciones (SOC) 2, que puede utilizar para crear su CP y CPS y para llevar a cabo los procedimientos de evaluación y verificación del control según sea necesario. AWS Los informes SOC son informes de examen independientes de terceros que demuestran cómo se AWS logran los principales controles y objetivos de cumplimiento. El objetivo de los informes es ayudarlo a usted y a sus auditores a comprender los AWS controles establecidos para respaldar las operaciones y el cumplimiento.

Este documento presenta un marco que se ajusta al RFC 3647 para ayudarlo a redactar su CP y CPS, e identifica la responsabilidad compartida entre usted y. AWS Private CA Las secciones de los requisitos de CP/CPS en las que AWS Private CA tiene una responsabilidad de cumplimiento se identifican como «Compartidas» o «» y se proporciona la correspondiente «información complementaria» para ayudarle a comprender cómo se cumple el requisito de CP/CPS asociado.AWS Private Certificate Authority AWS Private CA Por ejemplo, el requisito 5 (4.5.1) es una AWS Private CA responsabilidad y puede encontrar el lenguaje de control correspondiente

en la sección D.6 del informe sobre el AWS SOC 2 para ayudarle a completar su CP/CPS. Para obtener más información sobre los informes del AWS SOC y sobre cómo puede solicitar acceso a los informes del SOC, visite nuestra página sobre el SOC. FAQs

Requisitos y responsabilidades del CP/CPS

Requisito de CP/CPS	Responsabilidad	Información complementaria
1. Introducción (todos)	Usted	Usted es responsable de documentar la descripci ón general, el nombre y la identificación del documento , los participantes de la PKI, el uso de los certificados, la administración de las políticas y las definiciones y acrónimos relacionados con su PKI.
2. Responsabilidades de publicación y repositorio (todas)	Usted	Usted es responsable de documentar las definiciones relacionadas con su PKI.
3. Identificación y autentica ción (todas)	Usted	Usted es responsable de documentar los procedimi entos utilizados para autentica r la identidad u otros atributos del solicitante de un certificado de usuario final ante una CA o una autoridad de registro (RA) antes de emitir el certificado.
4. Requisitos operativos del ciclo de vida de los certifica dos (4.4.1 — 4.4.6, 4.4.9 — 4.4.11)	Compartido	Usted es responsable de especificar los requisitos impuestos a la CA emisora, al sujeto CAs RAs, a los suscriptores u otros participa ntes con respecto al ciclo de vida de un certificado.

Requisito de CP/CPS	Responsabilidad	Información complementaria
		AWS Private CA le proporcio na dos mecanismos totalment e gestionados para ayudar a comprobar el estado de las revocaciones: el Protocolo de estado de los certificados en línea (OCSP) y las listas de revocación de certifica dos (CRLs) para ayudarle a cumplir las disposiciones 4.4.9 y 4.4.10.
4. Requisitos operativos del ciclo de vida de los certifica dos (4.4.7, 4.4.8, 4.4.12)	N/A	AWS Private CA no admite el cambio de claves, la modificación de certificados ni el depósito en garantía y la recuperación de claves.

Requisito de CP/CPS	Responsabilidad	Información complementaria
5. Controles de instalaciones, gestión y operaciones (4.5.1)	AWS Private CA	Usted hereda los controles de acceso que le ayudan a cumplir los requisitos de esta sección que entran dentro del ámbito del informe de tipo 2 sobre el AWS Private CA SOC 2 (consulte la sección D.6 Seguridad física y protección ambiental). **Note** Usted es responsable de la seguridad física y la clasificación de los datos de CA exportados o transferidos fuera del AWS entorno, pero no de la seguridad física de los datos de CA almacenados en él. AWS
5. Controles de instalaciones, administración y operaciones (4.5.2)	Compartido	Usted es responsable de cumplir los requisitos de esta sección específicos para definir las funciones de confianza para las operaciones de su entorno de PKI. AWS Private CA mantiene las funciones de confianza específicas del acceso físico a los módulos criptográficos.

Requisito de CP/CPS	Responsabilidad	Información complementaria
5. Controles de instalaciones, gestión y operaciones (4.5.3)	Compartido	Usted es responsable de cumplir con los requisitos de esta sección específicos para los procedimientos de verificación de anteceden tes, capacitación y medidas disciplinarias para sus personas de confianza. Usted hereda los controles relacionados con la verificac ión de antecedentes, la formación y los procedimi entos disciplinarios de AWS los empleados que entran en el ámbito del informe AWS Private CA SOC 2 de tipo 2 (consulte la sección A. Políticas, A.1 Entorno de control, B. Comunicaciones y D.1 Organización de la seguridad y D.2 Acceso de los usuarios de los empleados).

Requisito de CP/CPS	Responsabilidad	Información complementaria
5. Controles de instalaciones, gestión y operaciones (4.5.4)	Compartido	Usted es responsable de habilitar, configurar la retención y proteger y auditar los registros CloudTrail y CloudWatch alertas de informes. Además, es responsable de crear procedimientos de procesamiento de registros y realizar evaluaciones de vulnerabilidad del uso del AWS Private CA servicio que cumplan con los requisitos de esta sección. Usted hereda los controles relacionados con la disponibilidad de sus registros, la gestión de la access/site security, CA/RA configuración física, la seguridad de AWS los registros de la AWS infraestructura y las evaluaciones de vulnerabilidad de la infraestructura que entran dentro del ámbito del informe AWS Private CA SOC 2 tipo 2 (consulte la sección A.1 Entorno de control, la sección C.1 Compromisos de servicio, D.2 Acceso a los usuarios de los empleados, D.3 Seguridad lógica, D.6 Seguridad física y protección ambiental, D.7 Gestión de cambios, D.8 Integridad, disponibilidad y

Requisito de CP/CPS	Responsabilidad	Información complementaria
		redundancia de los datos y E.1 Actividades de supervisión).
5. Controles de instalaciones, administración y operaciones (4.5.5)	Compartido	Usted es responsable de configurar los períodos de respaldo y retención que cumplan con los requisitos de esta sección. Usted hereda los controles relacionados con la disponibilidad de sus registros (al configurarlos) que entran dentro del ámbito del informe AWS Private CA SOC 2 de tipo 2 (consulte D.8 Integridad, disponibilidad y redundancia de los datos).
5. Controles de instalaciones, administración y operaciones (4.5.6)	N/A	AWS Private CA no es compatible con Key Changeover.

Requisito de CP/CPS	Responsabilidad	Información complementaria
5. Controles de instalaciones, gestión y operaciones (4.5.7)	Compartido	Usted es responsable de implementar los procedimientos de gestión de incidentes y compromisos específicos para su uso AWS Private CA que cumplan con los requisitos de esta sección. Usted hereda los procedimientos de gestión de incidentes, gestión de riesgos, continuidad empresarial y recuperación ante desastres específicos para las operaciones de infraestructura y alojamiento de sitios físicos que le ayudan a cumplir los requisitos de esta sección que entran dentro del ámbito del informe de privacidad de tipo 2 del AWS Private CA SOC 2 (consulte D.8 Integridad, disponibilidad y redundancia de los datos y la sección D.10 Privacidad).
5. Controles de instalaciones, gestión y operaciones (4.5.8)	Usted	Debe documentar los requisitos relacionados con los procedimientos de rescisión y rescisión de una CA o RA, incluida la identidad del custodio de los registros de archivo de CA y RA.

Requisito de CP/CPS	Responsabilidad	Información complementaria
6. Controles técnicos (4.6.1)	Compartido	Usted es responsable de documentar las necesidades de generación e instalación de claves para su PKI. AWS Private CA le proporcio na módulos criptográficos con certificación FIPS 140-3 de nivel 3 para la generación de claves de CA.
6. Controles técnicos (4.6.2)	Compartido	Usted es responsable de documentar la protección por clave privada y los controles de ingeniería de los módulos criptográficos, como los requisitos de los estándares criptográficos y los controles de varias personas. AWS Private CA le proporcio na módulos criptográficos con certificación FIPS 140-3 de nivel 3 para la generación de claves de CA y controles de acceso físico bipartitos también. HSMs
6. Controles técnicos (4.6.3)	Usted	Usted es responsable de documentar otros aspectos de la administración de los pares de claves, como el archivo de su clave pública y el período de funcionamiento de los certificados.

Requisito de CP/CPS	Responsabilidad	Información complementaria
6. Controles técnicos (4.6.4)	N/A	AWS Autoridad de certificación privada de AWS HSMs están siempre en línea y no tienen la noción de «datos de activación». **Note** Usted es responsable de implementar controles de acceso de los usuarios a su CA privada para restringir adecuadamente la capacidad de crear CA y emitir certificados.

Requisito de CP/CPS	Responsabilidad	Información complementaria
6. Controles técnicos (4.6.5)	Compartido	Usted es responsable de documentar los controles de seguridad informática para el uso de su CA privada. Usted hereda los controles relacionados con el acceso lógico de AWS los empleados , los controles de seguridad de red e informática de la AWS infraestructura y los controles de los parámetros de contraseña de las cuentas de los AWS empleados que entran dentro del ámbito del informe AWS Private CA SOC 2 de tipo 2 (consulte la sección D.2 Acceso de los usuarios de los empleados, D.3 Seguridad lógica y D.6 Seguridad física y protección ambiental).

Requisito de CP/CPS	Responsabilidad	Información complementaria
6. Controles técnicos (4.6.6)	Compartido	Usted es responsable de documentar los controles de gestión de la seguridad relacionados con el uso de su CA privada. Usted hereda los controles relacionados con los controles de desarrollo del sistema del AWS Private CA servicio que entran dentro del ámbito del informe de tipo 2 del AWS Private CA SOC 2 (consulte la sección D.7 Gestión de cambios).
6. Controles técnicos (4.6.7)	Compartido	Usted es responsable de documentar los controles de seguridad de la red para el uso de Private CA, si corresponde a su entorno de PKI. Usted hereda los controles relacionados con los controles de seguridad de la red de la AWS infraestructura incluidos en el informe de tipo 2 del AWS Private CA SOC 2 (consulte la sección C.1, los compromisos de servicio, la sección D.3, la seguridad lógica y las actividades de supervisión de la sección E.1).

Requisito de CP/CPS	Responsabilidad	Información complementaria
6. Controles técnicos (4.6.8)	AWS Private CA	AWS Private CA utiliza fuentes horarias confiables para marcar la fecha y hora de los datos de CA.
7. Perfiles de certificado, CRL y OCSP (todos)	Compartido	Usted es responsable de documentar los requisito s del perfil y la entrada de certificados que satisfagan las necesidades de su entorno de PKI. AWS Private CA le proporcio na plantillas de perfil para ayudarle a cumplir los requisitos de su perfil.
8. Auditoría de cumplimiento y otras evaluaciones (todas)	Compartido	Usted es responsable de documentar las auditorías de cumplimiento y otras evaluacio nes. AWS Private CA le proporcio na un informe SOC 2 para ayudarlo a usted y a sus auditores a comprender los AWS controles establecidos para respaldar las operaciones y el cumplimiento.
9. Otros asuntos comerciales y legales	Usted	Usted es responsable de documentar los asuntos comerciales y legales generales relacionados con su CA privada.

Supervise AWS Private CA los recursos

La supervisión es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de Autoridad de certificación privada de AWS AWS las demás soluciones. AWS proporciona las siguientes herramientas de monitoreo para observar Autoridad de certificación privada de AWS, informar cuando algo anda mal y tomar medidas automáticas cuando sea apropiado:

- Amazon CloudWatch monitorea tus AWS recursos y las aplicaciones en las que AWS ejecutas en tiempo real. Puede recopilar métricas y realizar un seguimiento de las métricas, crear paneles personalizados y definir alarmas que le advierten o que toman medidas cuando una métrica determinada alcanza el umbral que se especifique. Por ejemplo, puede CloudWatch hacer un seguimiento del uso de la CPU u otras métricas de sus instancias de Amazon EC2 y lanzar automáticamente nuevas instancias cuando sea necesario. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.
- Amazon CloudWatch Logs le permite supervisar, almacenar y acceder a sus archivos de registro desde instancias de Amazon EC2 y otras fuentes. CloudTrail CloudWatch Los registros pueden monitorear la información de los archivos de registro y notificarle cuando se alcancen ciertos umbrales. También se pueden archivar los datos del registro en un almacenamiento de larga duración. Para obtener más información, consulta la Guía del usuario CloudWatch de Amazon Logs.
- AWS CloudTrail captura las llamadas a la API y otros eventos relacionados que realiza la Cuenta de AWS o que se realizan en nombre de esta. Además, entrega los archivos de registro a un bucket de Amazon S3 especificado. También pueden identificar qué usuarios y cuentas llamaron a AWS, la dirección IP de origen de las llamadas y el momento en que estas se realizaron. Para obtener más información, consulte la Guía del usuario de AWS CloudTrail.
- Amazon EventBridge es un servicio de bus de eventos sin servidor que facilita la conexión de sus aplicaciones con datos de diversas fuentes. EventBridge ofrece un flujo de datos en tiempo real desde sus propias aplicaciones, aplicaciones Software-as-a-Service (SaaS) y AWS servicios, y dirige esos datos a destinos como Lambda. Esto le permite monitorear los eventos que ocurren en los servicios y crear arquitecturas basadas en eventos. Para obtener más información, consulta la Guía del EventBridge usuario de Amazon.

En los siguientes temas se describen las herramientas de AWS monitoreo de la nube disponibles para su uso con Autoridad de certificación privada de AWS.

AWS Private CA CloudWatch métricas

Amazon CloudWatch es un servicio de monitorización de AWS recursos. Puede usarlo CloudWatch para recopilar métricas y realizar un seguimiento, configurar alarmas y reaccionar automáticamente ante los cambios en sus AWS recursos. CloudWatch las métricas se publican al menos una vez.

Autoridad de certificación privada de AWS admite las siguientes CloudWatch métricas.

Métrica	Description (Descripción)
`CRLGenerated`	Se crea una lista de revocación de certificados (CRL). Esta métrica solo se aplica a una CA privada.
`MisconfiguredCRLBucket`	El bucket de S3 especificado para la CRL no está configurado correctamente. Compruebe la política del bucket. Esta métrica solo se aplica a una CA privada.
`Time`	El tiempo en milisegundos que transcurre entre una solicitud de emisión y la finalización (o el fracaso) de la emisión. Esta métrica se aplica únicamente a la IssueCertificateoperación.
`Success`	Se emitió correctamente un certificado. Esta métrica solo se aplica a la IssueCert ificateoperación.
`Failure`	Una operación ha devuelto un error. Esta métrica solo se aplica a la IssueCert ificateoperación.

Para obtener más información sobre CloudWatch las métricas, consulte los siguientes temas:

- Uso de Amazon CloudWatch Metrics
- Creación de Amazon CloudWatch Alarms

AWS Private CA Supervise con CloudWatch eventos

Puede usar Amazon CloudWatch Events para automatizar sus AWS servicios y responder automáticamente a eventos del sistema, como problemas de disponibilidad de aplicaciones o cambios en los recursos. Los eventos de AWS los servicios se envían a CloudWatch Events prácticamente en tiempo real. Puedes escribir reglas sencillas para indicar qué eventos te interesan y las acciones automatizadas que debes tomar cuando un evento coincide con una regla. CloudWatch Los eventos se publican al menos una vez. Para obtener más información, consulte Crear una regla de CloudWatch eventos que se active en un evento.

CloudWatch Los eventos se convierten en acciones con Amazon EventBridge. Con EventBridge, puedes usar eventos para activar objetivos que incluyen AWS Lambda funciones, AWS Batch trabajos, temas de Amazon SNS y muchos otros. Para obtener más información, consulta ¿Qué es Amazon EventBridge?

Éxito o error al crear una CA privada

Estos eventos los desencadena la CreateCertificateAuthorityoperación.

Success

En caso de éxito, la operación devuelve el ARN de la nueva entidad de certificación.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA Creation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T19:14:56Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   ],
   "detail":{
      "result":"success"
   }
}
```

Failure

En caso de error, la operación devuelve un ARN para la entidad de certificación. Mediante el ARN, puede llamar DescribeCertificateAuthoritypara determinar el estado de la CA.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA Creation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T19:14:56Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   ],
   "detail":{
      "result":"failure"
   }
}
```

Éxito o error al emitir un certificado

Estos eventos los desencadena la IssueCertificateoperación.

Success

En caso de éxito, la operación devuelve el certificado ARNs de la CA y el nuevo certificado.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA Certificate Issuance",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T19:57:46Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
   ],
   "detail":{
```

```
      "result":"success"
   }
}
```

Failure

En caso de error, la operación devuelve un certificado ARN y el ARN de la entidad de certificación. Con el certificado ARN, puede llamar GetCertificatepara ver el motivo del error.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA Certificate Issuance",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T19:57:46Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
   ],
   "detail":{
      "result":"failure"
   }
}
```

Éxito al revocar un certificado

Este evento lo desencadena la RevokeCertificateoperación.

No se envía ningún evento si la revocación devuelve un error o si el certificado ya ha sido revocado.

Correcto

En caso de éxito, la operación devuelve el certificado ARNs de la CA y el certificado revocado.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA Certificate Revocation",
```

```
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-05T20:25:19Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
   ],
   "detail":{
      "result":"success"
   }
}
```

Éxito o error al generar una CRL

Estos eventos los desencadena la RevokeCertificateoperación, lo que debería dar lugar a la creación de una lista de revocación de certificados (CRL).

Success

En caso de éxito, la operación devuelve el ARN de la entidad de certificación asociada a la CRL.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA CRL Generation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T21:07:08Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   ],
   "detail":{
      "result":"success"
   }
}
```

Error 1: no se pudo guardar la CRL en Amazon S3 debido a un error de permiso

Compruebe los permisos de bucket de Amazon S3 si se produce este error.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA CRL Generation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-07T23:01:25Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   ],
   "detail":{
      "result":"failure",
      "reason":"Failed to write CRL to S3. Check your S3 bucket permissions."
   }
}
```

Error 2: no se ha podido guardar la CRL en Amazon S3 debido a un error interno

Vuelva a intentar la operación si se produce este error.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA CRL Generation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-07T23:01:25Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   ],
   "detail":{
      "result":"failure",
      "reason":"Failed to write CRL to S3. Internal failure."
   }
}
```

Error 3: Autoridad de certificación privada de AWS no se pudo crear una CRL

Para solucionar este error, compruebe las métricas de CloudWatch .

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA CRL Generation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-07T23:01:25Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   ],
   "detail":{
      "result":"failure",
      "reason":"Failed to generate CRL. Internal failure."
   }
}
```

Éxito o Error al crear un informe de auditoría de CA

Estos eventos los desencadena la CreateCertificateAuthorityAuditReportoperación.

Success

En caso de éxito, la operación devuelve el ARN de la entidad de certificación y el ID del informe de auditoría.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA Audit Report Generation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T21:54:20Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "audit_report_ID"
   ],
   "detail":{
```

```
      "result":"success"
   }
}
```

Failure

Un informe de auditoría puede fallar Autoridad de certificación privada de AWS si no tiene PUT permisos en su bucket de Amazon S3, cuando el cifrado está activado en el bucket o por otros motivos.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA Audit Report Generation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T21:54:20Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "audit_report_ID"
   ],
   "detail":{
      "result":"failure"
   }
}
```

Registro de llamadas a la AWS Private Certificate Authority API mediante AWS CloudTrail

AWS Private Certificate Authority está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en AWS Private CA. CloudTrail captura las llamadas a la API y las operaciones de firma para eventos AWS Private CA como eventos. Las llamadas capturadas incluyen llamadas desde la AWS Private CA consola y llamadas en código a las operaciones de la AWS Private CA API. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos para AWS Private CA. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por usted CloudTrail, puede determinar a

AWS Private CA qué dirección IP se realizó la solicitud, quién la realizó, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la Guía AWS CloudTrail del usuario.

AWS Private CA información en CloudTrail

CloudTrail está habilitada en tu cuenta Cuenta de AWS al crear la cuenta. Cuando se produce una actividad en AWS Private CA, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Puede ver, buscar y descargar eventos recientes en su Cuenta de AWS. Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para tener un registro continuo de tus eventos Cuenta de AWS, incluidos los eventos para AWS Private CA ti, crea una ruta. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las Regiones de AWS. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para más información, consulte los siguientes temas:

- Introducción a la creación de registros de seguimiento
- CloudTrail servicios e integraciones compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones y recibir archivos de CloudTrail registro de varias cuentas

Todas AWS Private CA las acciones se registran CloudTrail y se documentan en la referencia de la AWS Private CA API. Por ejemplo, las llamadas a las `ImportCACertificate` `CreateAuditReport` acciones `IssueCertificate` y las llamadas generan entradas en los archivos de CloudTrail registro.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de usuario root o AWS Identity and Access Management (IAM).

- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Para obtener más información, consulte el elemento userIdentity de CloudTrail .

AWS Private CA eventos de gestión

AWS Private CA se integra CloudTrail para registrar las acciones de la API realizadas por un usuario, un rol o un AWS servicio en AWS Private CA. Se puede utilizar CloudTrail para supervisar las solicitudes de AWS Private CA API en tiempo real y almacenar registros en Amazon Simple Storage Service, Amazon CloudWatch Logs y Amazon CloudWatch Events. AWS Private CA admite el registro de las siguientes acciones y operaciones como eventos en archivos de CloudTrail registro:

- CreateCertificateAuthority
- CreateCertificateAuthorityAuditReport
- CreatePermission
- DeleteCertificateAuthority
- DeletePermission
- DeletePolicy
- DescribeCertificateAuthority
- DescribeCertificateAuthorityReport
- GetCertificate
- GetCertificateAuthorityCertificate
- GetCertificateAuthorityCsr
- GetPolicy
- ImportCertificateAuthorityCertificate
- IssueCertificate
- ListCertificateAuthorities
- ListPermissions
- ListTags
- PutPolicy
- RestoreCertificateAuthority

- RevokeCertificate
- TagCertificateAuthority
- UntagCertificateAuthority
- UpdateCertificateAuthority
- `GenerateOCSPResponse`- Se activa cuando se AWS Private CA genera una respuesta OCSP.
- `SignCertificate`- Se genera cuando tu cliente llama IssueCertificate.
- `SignOCSPResponse`- Se genera cuando se AWS Private CA firma una respuesta de OCSP.
- `GenerateCRL`- Se genera cuando se AWS Private CA genera una lista de revocación de certificados (CRL).
- `SignCACSR`- Se genera al firmar una solicitud de AWS Private CA firma de certificado (CSR) de una autoridad certificadora (CA).
- `SignCRL`- Se genera al AWS Private CA firmar una CRL.

Ejemplos AWS Private CA de eventos

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

Los siguientes son ejemplos de AWS Private CA CloudTrail eventos.

Ejemplo 1: Evento de gestión, **`IssueCertificate`**

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `IssueCertificate` acción.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA Certificate Issuance",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T19:57:46Z",
   "region":"region",
```

```
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
   ],
   "detail":{
      "result":"success"
   }
}
```

Ejemplo 2: Evento de administración, **`ImportCertificateAuthorityCertificate`**

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `ImportCertificateAuthorityCertificate` acción.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T21:53:28Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"ImportCertificateAuthorityCertificate",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "certificate":{
         "hb":[
            45,
            45,
            ...10
         ],
         "offset":0,
         "isReadOnly":false,
         "bigEndian":true,
```

```
            "nativeByteOrder":false,
            "mark":-1,
            "position":1257,
            "limit":1257,
            "capacity":1257,
            "address":0
         },
         "certificateChain":{
            "hb":[
               45,
               45,
               ...10
            ],
            "offset":0,
            "isReadOnly":false,
            "bigEndian":true,
            "nativeByteOrder":false,
            "mark":-1,
            "position":1139,
            "limit":1139,
            "capacity":1139,
            "address":0
         }
      },
      "responseElements":null,
      "requestID":"request_ID",
      "eventID":"event_ID",
      "eventType":"AwsApiCall",
      "recipientAccountId":"account"
}
```

Solucionar problemas con AWS Private Certificate Authority

Consulte los siguientes temas si tiene problemas cuando utiliza AWS Private Certificate Authority.

Temas

- Solucione problemas de revocación AWS Private CA de certificados
- Solucione los problemas de los mensajes de excepción AWS Private Certificate Authority
- Solucione los errores de los AWS Private CA certificados compatibles con Matter

Solucione problemas de revocación AWS Private CA de certificados

Latencia de respuesta del OCSP

La capacidad de respuesta del OCSP puede ser más lenta si la persona que llama se encuentra geográficamente alejada de una caché periférica regional o de la región de la CA emisora. Para obtener más información sobre la disponibilidad de la caché periférica regional, consulte Global Edge Network. Recomendamos emitir los certificados en una región cercana a donde se vayan a utilizar.

Revocación de certificados autofirmados

No puede revocar un certificado de CA autofirmado. Para revocar el certificado de forma funcional, elimine la CA.

Solucione los problemas de los mensajes de excepción AWS Private Certificate Authority

Un Autoridad de certificación privada de AWS comando puede fallar por varios motivos. Para obtener información sobre cada excepción y recomendaciones para resolverlas, consulte la tabla siguiente.

Autoridad de certificación privada de AWS Excepciones

Excepción devuelta por Autoridad de certificación privada de AWS	Description (Descripción)	Corrección
`AccessDeniedException`	Los permisos necesarios para usar el comando dado no los ha delegado a la cuenta de llamada una entidad de certificación privada.	Para obtener información sobre la delegación de permisos en Autoridad de certificación privada de AWS, consulteAsignación de permisos de renovación de certificados a ACM.
`InvalidArgsException`	Se ha realizado una solicitud de creación o renovación de certificados con parámetros no válidos.	Compruebe la documentación individual del comando para asegurarse de que los parámetros de entrada son válidos. Si va a crear un nuevo certificado, asegúrese de que el algoritmo de firma solicitado se puede utilizar con el tipo de clave de la entidad de certificación.
`InvalidStateException`	La entidad de certificación privada asociada no puede renovar el certificado porque no está en el estado `ACTIVE`.	Intente restaurar la entidad de certificación privada. Si la entidad de certificación privada está fuera de su período de restauración, la entidad de certificación no se puede restaurar y el certificado no se puede renovar.
`LimitExceededException`	Cada entidad de certificación (CA) tiene una cuota de certificados que puede emitir. La entidad de certificación	Póngase en contacto con el AWS Support Centro para solicitar un aumento de cuota.

Excepción devuelta por Autoridad de certificación privada de AWS	Description (Descripción)	Corrección
	privada asociada al certifica do designado ha alcanzado su cuota. Para obtener más información, consulte Service Quotas en la Guía de Refere ncia general de AWS .	
`MalformedCSRExcept ion`	La solicitud de firma de certificados (CSR) que se envió a Autoridad de certifica ción privada de AWS no se puede verificar ni validar.	Confirme que su CSR se generó y configuró correctam ente.
`OtherException`	Un error interno ha provocado un error en la solicitud.	Intente volver a ejecutar el comando. Si el problema persiste, póngase en contacto con el AWS Support Centro.
`RequestFailedExcep tion`	Un problema de red en su AWS entorno provocó el error de la solicitud.	Intente realizar de nuevo la solicitud . Si el error persiste, compruebe la configuración de Amazon VPC (VPC).
`ResourceNotFoundEx ception`	La entidad de certificación privada que emitió el certifica do se eliminó y ya no existe.	Solicite un nuevo certificado de otra entidad de certificación activa.

Excepción devuelta por Autoridad de certificación privada de AWS	Description (Descripción)	Corrección
`ThrottlingException`	Una acción de API solicitad a devolvió un error porque superó una cuota.	Confirme que no está emitiendo más llamadas de las permitidas por Autoridad de certificación privada de AWS. También puede producirse un error de `ThrottlingException` porque ha encontrado una condición transitoria en lugar de una cuota superada. Si encuentra el error y no ha superado el número de llamadas de la cuota, vuelva a intentar la solicitud. Si se ve limitado por una cuota, puede solicitar un aumento. Para obtener más información, consulte Service Quotas en la Referencia general de AWS Guía.

Excepción devuelta por Autoridad de certificación privada de AWS	Description (Descripción)	Corrección
`ValidationException`	A los parámetros de entrada de la solicitud se les ha aplicado un formato incorrecto o el período de validez del certificado raíz finaliza antes del período de validez del certificado solicitado.	Compruebe los requisitos de sintaxis de los parámetros de entrada del comando, así como el período de validez del certificado raíz de su entidad de certificación. Para obtener información sobre cómo cambiar el período de validez, consulte Actualice una CA privada en AWS Private Certificate Authority.

Solucione los errores de los AWS Private CA certificados compatibles con Matter

El estándar de conectividad Matter especifica las configuraciones de certificados que mejoran la seguridad y la coherencia de los dispositivos de Internet de las cosas (IoT). Puede encontrar ejemplos de Java para crear certificados de CA raíz, CA intermedia y entidad final compatibles con Matter en Se utiliza Autoridad de certificación privada de AWS para implementar los certificados Matter.

Para facilitar la solución de problemas, los desarrolladores de Matter proporcionan una herramienta de verificación de certificados llamada chip-cert. Los errores de los que informa la herramienta se enumeran en la tabla siguiente con las correcciones.

Código de error	Significado	Corrección
0x00000035	Las extensiones `BasicConstraints` , `KeyUsage` y `ExtensionKeyUsage`	Asegúrese de haber seleccionado la plantilla correcta para su c uso.

Código de error	Significado	Corrección
	deben marcarse como críticas.	
0x000000 0	La extensión del identificador de clave de autoridad debe estar presente.	Autoridad de certificación privada de AWS no establece la exte identificador de clave de autoridad en los certificados raíz. Debe un AuthorityKeyIdentifier valor codificado en Base64 mediante l y, a continuación, pasarlo por un. CustomExtension Para obten información, consulte Active una CA raíz para los certificados o s de nodo (NOC). y Activar una autoridad de certificación de pr (PAA).
0x000000 E	El certificado ha caducado.	Asegúrese de que el certificado que utiliza no haya caducado.

Código de error	Significado	Corrección
0x000000 4	Error en la validación de la cadena de certificados.	Este error se puede producir si intenta crear un certificado de e final compatible con Matter sin utilizar los ejemplos de Java pro nados, que utilizan la Autoridad de certificación privada de AWS para transmitir un certificado configurado correctamente. KeyUs De forma predeterminada, Autoridad de certificación privada de genera valores de KeyUsage extensión de nueve bits, y el nove bit genera un byte adicional. Matter ignora el byte adicional dura las conversiones de formato, lo que provoca errores en la valida de la cadena. Sin embargo, se puede usar a CustomExtensione `APIPassthrough` plantilla para establecer el número exacto del valor. `KeyUsage` Para ver un ejemplo, consulta Crear un ce operativo de nodo (NOC). Si modifica el código de ejemplo o utiliza una utilidad X.509 alte va, como OpenSSL, debe realizar una verificación manual para errores de validación en cadena. Para comprobar que las conversiones no tienen pérdidas 1. Utilice openssl para comprobar que un certificado de nodo final) contiene una cadena válida. En este ejemplo, `rcac.` certificado de CA raíz, `icac.pem` es el certificado de CA in y `noc.pem` es el certificado de nodo. ``` openssl verify -verbose -CAfile <(cat rcac.pem icac noc.pem ``` 2. Utilice chip-cert para convertir el certificado de nodo con fo PEM al formato TLV (etiqueta, longitud, valor) y viceversa. ``` ./chip-cert convert-cert noc.pem noc.chip -c ./chip-cert convert-cert noc.chip noc_converted.pem ```

Código de error	Significado	Corrección
		Los archivos `noc.pem` y `noc_converted.pem` deben se exactamente los mismos que los confirmados por una herra de comparación de cadenas.

Proteja Kubernetes con AWS Private Certificate Authority

Puede utilizarlos AWS Private Certificate Authority para proporcionar certificados de autenticación y cifrado seguros a través de TLS y mTLS. AWS Private CA proporciona un complemento de código abierto, AWS Private CA Connector for Kubernetes, (`aws-privateca-issuer`) para el complemento administrador de certificados de Kubernetes, ampliamente utilizado, que solicita certificados, los distribuye entre los secretos de Kubernetes y automatiza la renovación de los certificados.

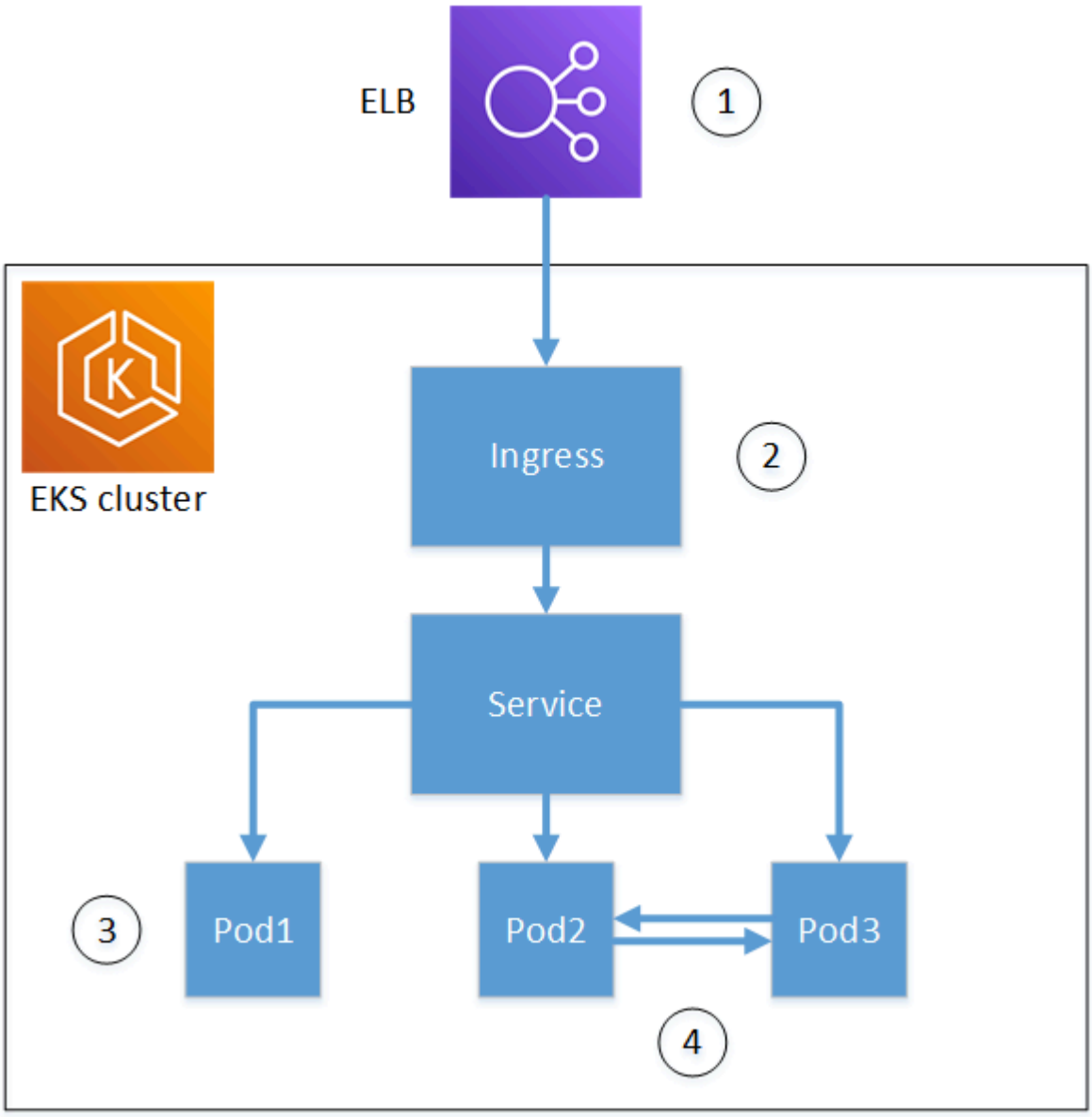
El `aws-privateca-issuer` `cert-manager` complemento te permite emitir certificados a través de él. AWS Private CA Puede usar el complemento con Amazon Elastic Kubernetes Service (Amazon EKS), un clúster de Kubernetes autogestionado en o en un clúster de Kubernetes local. AWS El complemento funciona tanto en arquitecturas x86 como ARM.

AWS Private CA tiene claves respaldadas por HSM que no se pueden exportar. Si tiene requisitos normativos para controlar el acceso y auditar sus operaciones de CA, puede utilizarlos AWS Private CA para mejorar la auditabilidad y respaldar el cumplimiento.

> **Note**
>
> Si utiliza Amazon EKS, le recomendamos que utilice los `aws-privateca-connector-for-kubernetes` complementos `cert-manager` y para disfrutar de una experiencia de instalación gestionada. Para obtener más información, consulte los AWS complementos.

Conceptos

El siguiente diagrama muestra algunas de las opciones disponibles para usar TLS en un clúster de Amazon EKS. El clúster de ejemplo se encuentra detrás de un balanceador de carga. Los números identifican los posibles puntos finales de las comunicaciones protegidas por TLS.



1. Terminación en el balanceador de carga

   Elastic Load Balancing (Elastic Load Balancing) está integrado con el AWS Certificate Manager servicio. No es necesario instalarlo `cert-manager` en el balanceador de carga. Puede aprovisionar ACM con una CA privada, firmar un certificado con la CA privada e instalar el certificado mediante la consola Elastic Load Balancing. AWS Private CA los certificados se renuevan automáticamente.

   Como alternativa, puedes proporcionar un certificado privado a un proveedor que no sea un balanceador de AWS cargas para finalizar el TLS.

Esto proporciona una comunicación cifrada entre un cliente remoto y el equilibrador de carga. Los datos después de que el balanceador de carga se pasen sin cifrar al clúster de Amazon EKS.

2. Terminación en el controlador de entrada de Kubernetes

   El controlador de ingreso se encuentra dentro del clúster de Amazon EKS y actúa como balanceador de carga y enrutador. Para utilizar el controlador de entrada como punto final del clúster para la comunicación externa, debe:

   - Instale ambos y `cert-manager` `aws-privateca-issuer`
   - Proporcione al controlador un certificado privado TLS del AWS Private CA.

   Las comunicaciones entre el balanceador de cargas y el controlador de entrada están cifradas y los datos pasan sin cifrar a los recursos del clúster.

3. Terminación en un pod

   Cada pod es un grupo de uno o más contenedores que comparten recursos de almacenamiento y red. Si instalas ambos `cert-manager` `aws-privateca-issuer` y aprovisionas el clúster con una CA privada, Kubernetes puede instalar un certificado privado TLS firmado en los pods según sea necesario. De forma predeterminada, una conexión TLS que termina en un pod no está disponible para otros pods del clúster.

4. Proteja las comunicaciones entre los pods.

   Puedes aprovisionar varios pods con certificados para que puedan comunicarse entre sí. Los siguientes escenarios son posibles:

   - El aprovisionamiento con certificados autofirmados generó Kubernetes. Esto protege las comunicaciones entre los módulos, pero los certificados autofirmados no cumplen con los requisitos de la HIPAA o el FIPS.
   - Aprovisionamiento con certificados firmados por. AWS Private CA Esto requiere la instalación de ambos`cert-manager`. `aws-privateca-issuer` Luego, Kubernetes puede instalar certificados mTLS firmados en los pods según sea necesario.

Consideraciones

Cuando lo utilices AWS Private Certificate Authority con Kubernetes, ten en cuenta las siguientes consideraciones.

Uso multicuenta del administrador de certificados

Los administradores con acceso multicuenta a una CA pueden usar el `cert-manager` complemento de Kubernetes para aprovisionar certificados para un clúster mediante la CA compartida. Para obtener más información, consulta Prácticas recomendadas de seguridad para el acceso entre cuentas a redes privadas CAs.

Solo puede usar determinadas plantillas de AWS Private CA certificados en escenarios con varias cuentas.

En la siguiente tabla se enumeran las Autoridad de certificación privada de AWS plantillas que puede usar con cert-manager para aprovisionar un clúster de Kubernetes.

Plantillas compatibles con Kubernetes	Compatible con la modalidad entre cuentas
BlankEndEntityCertificate_ CSRPassthrough / V1 definición	No
CodeSigningCertificatedefinición de /V1	No
EndEntityCertificateDefinición de /V1	Sí
EndEntityClientAuthCertificateDefinición de /V1	Sí
EndEntityServerAuthCertificateDefinición de / V1	Sí
OCSPSigningDefinición de certificado/V1	No

Comience con AWS Private CA Connector for Kubernetes.

En los siguientes temas, se muestra cómo utilizarlos AWS Private CA para proteger las comunicaciones en un clúster de Kubernetes. Para ver otro ejemplo, consulta Encryption in transit for Kubernetes on. GitHub

Puede utilizar una autoridad de certificación privada para proteger las comunicaciones con sus clústeres de Amazon EKS. Antes de comenzar, asegúrese de que dispone de lo siguiente:

- Una AWS cuenta con los permisos adecuados dentro del ámbito de sus políticas de seguridad.

Amazon EKS clusters

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAM",
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:AttachRolePolicy",
        "iam:GetRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "EKS",
      "Effect": "Allow",
      "Action": [
        "eks:CreateAddon",
        "eks:DescribeAddon",
        "eks:CreatePodIdentityAssociation",
        "eks:DescribeCluster"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::*:role/CertManagerPrivateCARole"
    }
  ]
}
```

Other clusters

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GetAndIssuePCACertificates",
      "Effect": "Allow",
      "Action": [
        "acm-pca:GetCertificate",
        "acm-pca:IssueCertificate"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RolesAnywhere",
      "Effect": "Allow",
      "Action": [
        "rolesanywhere:CreateProfile"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAM",
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:AttachRolePolicy"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::*:role/CertManagerPrivateCARole"
    }
  ]
}
```

- Un clúster de Kubernetes. Para crear un clúster de Amazon Elastic Kubernetes Service, consulte la guía de inicio rápido de Amazon EKS. Para simplificar, cree una variable de entorno que contenga el nombre del clúster:

```
export CLUSTER=aws-privateca-demo
```

- La Región de AWS ubicación de su clúster de CA y Amazon EKS. Para simplificar, cree una variable de entorno que contenga la región:

```
export REGION=aws-region
```

- El nombre de recurso de Amazon (ARN) de una entidad de certificación AWS Private CA privada. Para simplificar, cree una variable de entorno que contenga el ARN de CA privado:

```
export CA_ARN="arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
```

  Para crear una CA privada, consulte https://docs.aws.amazon.com/privateca/latest/userguide/create-CA.html Crear una CA privada en AWS Private CA

- Un ordenador con el siguiente software instalado:
  - AWS CLI v2 configurado
  - kubectl v1.13+
  - Para clústeres que no son de Amazon EKS, Helm v3

Instale cert-manager

Para usar una CA privada, debe instalar el `cert-manager>` complemento que solicita los certificados, los distribuye y automatiza la renovación de los certificados. También debe instalar el `aws-private-ca-issuer` complemento que le permite emitir certificados privados desde. AWS Private CA Siga los siguientes pasos para instalar el complemento y el complemento.

Amazon EKS clusters

Instálelo `cert-manager` como complemento de Amazon EKS:

```
aws eks create-addon \
  --cluster-name $CLUSTER \
```

```
  --addon-name cert-manager \
  --region $REGION
```

Other clusters

Instálelo `cert-manager` con Helm:

```
helm repo add jetstack https://charts.jetstack.io
helm repo update

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --set crds.enabled=true
```

Configurar los permisos de IAM

El `aws-privateca-issuer` complemento requiere permiso para interactuar con él AWS Private CA. Para los clústeres de Amazon EKS, utilice la identidad del pod. Para otros clústeres, utilice AWS Identity and Access Management Roles Anywhere.

En primer lugar, cree una política de IAM. La política utiliza la política `AWSPrivateCAConnectorForKubernetesPolicy` gestionada. Para obtener más información sobre la política, consulte la guía AWSPrivateCAConnectorForKubernetesPolicyde referencia de políticas AWS gestionadas.

Amazon EKS clusters

1. Cree un archivo con el nombre `trust-policy.json` que contenga la siguiente política de confianza:

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Sid": "TrustPolicyForEKSClusters",
         "Effect": "Allow",
         "Principal": {
   ```

```
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

2. Ejecute los siguientes comandos para crear un rol de IAM:

```
ROLE_ARN=$(aws iam create-role \
  --role-name CertManagerPrivateCARole \
  --assume-role-policy-document file://trust-policy.json \
  --region $REGION \
  --output text \
  --query "Role.Arn")

 aws iam attach-role-policy \
  --role-name CertManagerPrivateCARole \
  --policy-arn arn:aws:iam::aws:policy/AWSPrivateCAConnectorForKubernetesPolicy
```

Other clusters

1. Cree un ancla de confianza que confíe en la CA privada en CA_ARN la que está almacenada. Para obtener instrucciones, consulte Cómo empezar con IAM Roles Anywhere. Cree una variable de entorno para almacenar el ARN del ancla de confianza:

```
export TRUST_ANCHOR_ARN=trustAnchorArn
```

2. Cree un archivo `trust-policy.json` que contenga la siguiente política de confianza:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "TrustPolicyForSelfManagedOrOnPremiseClusters",
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity",
                "sts:TagSession"
            ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": [
                        "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TRUST_ANCHOR_ARN"
                    ]
                },
                "StringEquals": {
                    "aws:PrincipalTag/x509Subject/CN": "aws-privateca-
issuer"
                }
            }
        }
    ]
}
```

3. Ejecute los siguientes comandos para crear un rol de IAM:

```
ROLE_ARN=$(aws iam create-role \
  --role-name CertManagerPrivateCARole \
  --assume-role-policy-document file://trust-policy.json \
  --query "Role.Arn" \
  --region $REGION \
  --output text)

 aws iam attach-role-policy \
  --role-name CertManagerPrivateCARole \
  --region $REGION \
  --policy-arn arn:aws:iam::aws:policy/AWSPrivateCAConnectorForKubernetesPolicy
```

Instale y configure el emisor del AWS Private CA clúster

Para instalar el `aws-privateca-connector-for-kubernetes` complemento, utilice los siguientes comandos:

Amazon EKS clusters

Crea el complemento:

```
aws eks create-addon --region $REGION \
  --cluster-name $CLUSTER \
  --addon-name aws-privateca-connector-for-kubernetes \
  --pod-identity-associations "[{
    \"serviceAccount\": \"aws-privateca-issuer\",
    \"roleArn\": \"$ROLE_ARN\"
  }]"
```

Luego espera a que el complemento esté activo:

```
aws eks describe-addon \
  --cluster-name $CLUSTER \
  --addon-name aws-privateca-connector-for-kubernetes \
  --region $REGION \
  --query 'addon.status'
```

Other clusters

1. Crea un perfil en IAM Roles Anywhere:

   ```
   PROFILE_ARN=$(aws rolesanywhere create-profile \
     --name "privateca-profile" \
     --role-arns "$ROLE_ARN" \
     --region "$REGION" \
     --query 'profile.profileArn' \
     --enabled \
     --output text)
   ```

2. Genere un certificado de cliente para usarlo con el Connector for Kubernetes y para IAM Roles Anywhere autenticarse con: AWS Private CA

   a. Genere una clave privada para el certificado de cliente:

   ```
   openssl genrsa -out client.key 2048
   ```

   b. Genere una solicitud de firma de certificado (CSR) para el certificado del cliente:

   ```
   openssl req -new \
   ```

```
  -key client.key \
  -out client.csr \
  -subj "/CN=aws-privateca-issuer"
```

c. Emita el certificado de cliente desde AWS Private CA:

```
CERT_ARN=$(aws acm-pca issue-certificate \
  --signing-algorithm SHA256WITHRSA \
  --csr fileb://client.csr \
  --validity Value=1,Type=DAYS \
  --certificate-authority-arn "$CA_ARN" \
  --region "$REGION" \
  --query 'CertificateArn' \
  --output text)
```

d. Guarde el certificado de cliente localmente:

```
aws acm-pca get-certificate \
  --certificate-authority-arn $CA_ARN \
  --certificate-arn $CERT_ARN \
  --region $REGION \
  --query 'Certificate'
  --output text > pca-issuer-client-cert.pem
```

3. Instale el AWS Private CA emisor en el clúster con el certificado de cliente:

a. Agregue el repositorio `awspca` de Helm:

```
helm repo add awspca https://cert-manager.github.io/aws-privateca-issuer
helm repo update
```

b. Cree un espacio de nombres:

```
kubectl create namespace aws-privateca-issuer
```

c. Guarde el certificado creado anteriormente en secreto:

```
kubectl create secret tls aws-privateca-credentials \
  -n aws-privateca-issuer \
  --cert=pca-issuer-client-cert.pem \
  --key=client.key
```

4. Instale el AWS Private CA emisor con IAM Roles Anywhere:

a. Cree un archivo con el nombre `values.yaml` necesario para configurar el complemento AWS Private CA emisor para usarlo con: IAM Roles Anywhere

```
cat > values.yaml <<EOF
env:
  AWS_EC2_METADATA_SERVICE_ENDPOINT: "http://127.0.0.1:9911"

extraContainers:
  - name: "rolesanywhere-credential-helper"
    image: "public.ecr.aws/rolesanywhere/credential-helper:latest"
    command: ["aws_signing_helper"]
    args:
      - "serve"
      - "--private-key"
      - "/etc/cert/tls.key"
      - "--certificate"
      - "/etc/cert/tls.crt"
      - "--role-arn"
      - "$ROLE_ARN"
      - "--profile-arn"
      - "$PROFILE_ARN"
      - "--trust-anchor-arn"
      - "$TRUST_ANCHOR_ARN"
    volumeMounts:
      - name: cert
        mountPath: /etc/cert/
        readOnly: true

volumes:
  - name: cert
    secret:
      secretName: aws-privateca-credentials
EOF
```

b. Instale el AWS Private CA emisor con: IAM Roles Anywhere

```
helm install aws-privateca-issuer awspca/aws-privateca-issuer \
  -n aws-privateca-issuer \
  -f values.yaml
```

Espere a que el emisor esté listo. Utilice el siguiente comando :

```
kubectl wait --for=condition=ready pods --all -n aws-privateca-issuer --timeout=120s
```

Y, a continuación, verifica la instalación para asegurarte de que todos los pods hayan alcanzado el `READY` estado siguiente:

```
kubectl -n aws-privateca-issuer get all
```

Para configurarlos`aws-private-ca-cluster-issuer`, crea un archivo YAML con un nombre `cluster-issuer.yaml` que contenga la configuración del emisor:

```
cat > cluster-issuer.yaml <<EOF
apiVersion: awspca.cert-manager.io/v1beta1
kind: AWSPCAClusterIssuer
metadata:
  name: aws-privateca-cluster-issuer
spec:
  arn: "$CA_ARN"
  region: "$REGION"
EOF
```

A continuación, aplique la configuración del clúster:

```
kubectl apply -f cluster-issuer.yaml
```

Compruebe el estado del emisor:

```
kubectl describe awspcaclusterissuer aws-privateca-cluster-issuer
```

Verá una respuesta parecida a la siguiente:

```
Status:
  Conditions:
    Last Transition Time:  2025-08-13T21:00:00Z
    Message:               AWS PCA Issuer is ready
    Reason:                Verified
    Status:                True
    Type:                  Ready
```

Administre el certificado del AWS Private CA cliente con cert-manager

Si no utiliza un clúster de Amazon EKS, después de iniciar manualmente un certificado de confianza, `aws-privateca-issuer` puede realizar la transición a un certificado de autenticación de cliente gestionado por`cert-manager`. Esto permite `cert-manager` renovar automáticamente el certificado de autenticación del cliente.

1. Cree un archivo llamado`pca-auth-cert.yaml`:

```
cat > pca-auth-cert.yaml <<EOF
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: aws-privateca-client-cert
  namespace: aws-privateca-issuer
spec:
  secretName: aws-privateca-credentials
  duration: 168h
  renewBefore: 48h
  commonName: aws-privateca-issuer
  privateKey:
    algorithm: ECDSA
    size: 256
    rotationPolicy: Always
  usages:
    - client auth
  issuerRef:
    name: aws-privateca-cluster-issuer
    kind: AWSPCAClusterIssuer
    group: awspca.cert-manager.io
EOF
```

2. Cree el nuevo certificado de autenticación de cliente administrado:

```
kubectl apply -f pca-auth-cert.yaml
```

3. Valide que se haya creado el certificado:

```
kubectl get certificate aws-privateca-client-cert -n aws-privateca-issuer
```

   Verá una respuesta parecida a la siguiente:

```
NAME                          READY   SECRET                          AGE
aws-privateca-client-cert     True    aws-privateca-credentials       19m
```

Emita su primer certificado TLS

Ahora que los `cert-manager` y `aws-privateca-issuer` están instalados, puede emitir un certificado.

Cree un archivo YAML con el nombre `certificate.yaml` que contenga el recurso del certificado:

```
cat > certificate.yaml <<EOF
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: example-certificate
  namespace: default
spec:
  secretName: example-certificate-tls
  issuerRef:
    name: aws-privateca-cluster-issuer
    kind: AWSPCAClusterIssuer
    group: awspca.cert-manager.io
  commonName: example.internal
  dnsNames:
    - example.internal
    - api.example.internal
  duration: 2160h # 90 days
  renewBefore: 360h # 15 days
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

Aplique el certificado mediante el siguiente comando:

```
kubectl apply -f certificate.yaml
```

A continuación, puede comprobar el estado del certificado con los siguientes comandos:

```
kubectl get certificate example-certificate
kubectl describe certificate example-certificate
```

Deberías ver una respuesta similar a esta:

```
NAME                  READY   SECRET                     AGE
example-certificate   True    example-certificate-tls    30s
```

Puede inspeccionar el certificado emitido con el siguiente comando:

```
kubectl get secret example-certificate-tls -o yaml
```

También puede decodificar y examinar el certificado con el siguiente comando:

```
kubectl get secret example-certificate-tls -o jsonpath='{.data.tls\.crt}' | base64 -d |
 openssl x509 -text -noout
```

Ejemplos

Los siguientes ejemplos muestran cómo se AWS Private CA puede utilizar con los clústeres de Kubernetes.

- Ejemplo: Cifrado en tránsito para Kubernetes
- Clústeres de Kubernetes habilitados para TLS con Autoridad de certificación privada de AWS y Amazon EKS
- Configuración del cifrado end-to-end TLS en Amazon EKS con el nuevo AWS Load Balancer Controller

Supervise Kubernetes con AWS Private CA

Para monitorear la CA privada del clúster de Kubernetes, utilice las técnicas descritas en. Supervise AWS Private CA los recursos Puede utilizar lo siguiente para supervisar una CA privada:

- AWS Private CA CloudWatch métricas
- AWS Private CA Supervise con CloudWatch eventos
- Registro de llamadas a la AWS Private Certificate Authority API mediante AWS CloudTrail

Solucione problemas de Kubernetes con AWS Private CA

Puede obtener los registros mediante `aws-private-ca-issuer` el siguiente procedimiento:

1. Obtenga el nombre del pod:

```
kubectl get pods -A
```

2. Para ver los registros de los emisores, usa el siguiente comando:

```
kubectl logs -n aws-privateca-issuer <pod-name> aws-privateca-issuer
```

3. Para ver los IAM Roles Anywhere registros, utilice el siguiente comando:

```
kubectl logs -n aws-privateca-issuer <pod-name> rolesanywhere-credentials-helper
```

Para comprobar el estado de su AWS Private CA emisor, utilice una de las siguientes opciones:

Para comprobar que su emisor está preparado, utilice el siguiente comando:

```
kubectl get AWSPCAClusterIssuers -o json | jq '.items[].status
```

La respuesta debería ser similar a la siguiente:

```
{
  "conditions": [
    {
      "lastTransitionTime": "2024-07-03T13:56:37Z",
      "message": "Issuer verified",
      "reason": "Verified",
      "status": "True",
      "type": "Ready"
    }
  ]
}
```

Si el emisor no se encuentra en el `Ready` estado, el `message` campo proporciona información sobre los motivos por los que el emisor no ha podido comunicarse con el `Ready` estado.

Para comprobar que el certificado está listo, utilice el siguiente comando:

```
kubectl get certificates -o json | jq '.items[].status'
```

La respuesta debe ser similar a la siguiente:

```
{
  "conditions": [
    {
      "lastTransitionTime": "2024-07-03T13:58:13Z",
      "message": "Certificate is up to date and has not expired",
      "observedGeneration": 1,
      "reason": "Ready",
      "status": "True",
      "type": "Ready"
    }
  ],
  "notAfter": "2024-10-01T13:58:12Z",
  "notBefore": "2024-07-03T12:58:12Z",
  "renewalTime": "2024-09-16T13:58:12Z",
  "revision": 1
}
```

Si el certificado no está en el `Ready` estado, el `message` campo proporciona información sobre los motivos por los que el certificado no ha podido llegar al `Ready` estado.

Autoridad de certificación privada de AWS Conector para Active Directory

AWS Private CA puede emitir y administrar los certificados requeridos por AWS Managed Microsoft AD. Con el Autoridad de certificación privada de AWS Connector para Active Directory (Connector para AD), puede sustituir las instalaciones empresariales o de terceros CAs por una CA privada gestionada de su propiedad, lo que permite la inscripción de certificados a los usuarios, grupos y máquinas que administra su AD.

Puede usar el Connector para AD con el AWS Managed Microsoft AD fin de eliminar la infraestructura local migrando su infraestructura de AD y de clave pública a la nube. Para los clientes que desean utilizarla AWS Private CA con su AD local, esta función también se integra con Connector. AWS Managed Microsoft AD

Temas

- ¿Es la primera vez que utiliza el Conector para AD?
- Configuración de Conector de para AD
- Comience a utilizar AWS Private CA Connector for Active Directory
- AWS Private CA conectores para Active Directory
- Integración de Connector for AD en aplicaciones basadas en eventos mediante Amazon EventBridge
- Solución de problemas con AWS Private CA Connector for Active Directory

¿Es la primera vez que utiliza el Conector para AD?

Si es la primera vez que usa el Conector para AD, le recomendamos que empiece leyendo las siguientes secciones:

- ¿Qué es Autoridad de certificación privada de AWS?
- ¿Qué es Directory Service?

Conector de acceso para AD

Puede acceder a Connector for AD a través de la consola AWS CLI, y APIs. Puedes acceder al conector de la consola desde la AWS Private CA consola, desde tu Directory Service consola o buscando Connector for AD en la barra de Consola de administración de AWS búsqueda.

Precios

El conector para AD se ofrece como una función sin Autoridad de certificación privada de AWS coste adicional. Solo paga por las entidades de certificación privadas y los certificados que emita a través de ellas.

Para obtener la información Autoridad de certificación privada de AWS de precios más reciente, consulte AWS Private Certificate Authority Precios. También puede utilizar la calculadora de precios de AWS para estimar los costos.

Configuración de Conector de para AD

Los pasos de esta sección son requisitos previos para usar Connector para AD. Se supone que ya ha creado una AWS cuenta. Tras completar los pasos de esta página, puedes empezar a crear un conector para AD.

Paso 1: Cree una CA privada mediante AWS Private CA

Configure una autoridad de certificación (CA) privada para emitir certificados para los objetos de su directorio. Para obtener más información, consulte Autoridades de certificación en AWS Private CA.

La entidad emisora de certificados privada debe estar en el `Active` estado para poder crear un conector para AD. El nombre de asunto de la CA privada debe incluir un nombre común. La creación del conector fallará si intenta crear un conector con una CA privada sin un nombre común.

Paso 2: Configurar un Active Directory

Además de una CA privada, necesita un Active Directory en una nube privada virtual (VPC). El Conector para AD admite los siguientes tipos de directorios que ofrece Directory Service:

- AWS Microsoft Active Directory administrado: con Directory Service él puede ejecutar Microsoft Active Directory (AD) como un servicio administrado. AWS Directory Service for Microsoft Active Directory también conocido como AWS Managed Microsoft AD, funciona con Windows Server 2019. Con AWS Managed Microsoft AD, puede ejecutar cargas de trabajo compatibles con

directorios en, Nube de AWS incluidas Microsoft Sharepoint y aplicaciones personalizadas basadas en .Net y SQL Server.

- Conector Active Directory: el Conector AD es una puerta de enlace del directorio que puede utilizar para redirigir solicitudes del directorio a su Microsoft Active Directory en las instalaciones sin almacenar información en caché en la nube. El Conector AD también admite la conexión a un dominio alojado en Amazon EC2

(Solo para Active Directory Connector) Paso 3: delegue los permisos a la cuenta de servicio

> **Note**
>
> Si utiliza AWS Managed Microsoft AD los permisos adicionales, se delegan automáticamente al autorizar el servicio Connector for AD en su directorio. Puede omitir este paso de requisito previo.

Cuando usa el Conector AD de Directory Service, debe delegar permisos adicionales a la cuenta de servicio. Configure la lista de control de acceso (ACL) en la cuenta de servicio para permitir la capacidad:

- Agregue y elimine un nombre de la entidad principal del servicio (SPN) para sí mismo
- Cree y actualice las entidades de certificación en los siguientes contenedores:

```
#containers
CN=Public Key Services,CN=Services,CN=Configuration
CN=AIA,CN=Public Key Services,CN=Services,CN=Configuration
CN=Certification Authorities,CN=Public Key Services,CN=Services,CN=Configuration
```

- Cree y actualice un objeto de entidad emisora de NTAuth certificados (CA). Nota: si el objeto CA de NTAuth certificados existe, debe delegar los permisos para él. Si el objeto no existe, debe delegar la capacidad de crear objetos secundarios en el contenedor de servicios de clave pública.

```
#objects
CN=NTAuthCertificates,CN=Public Key Services,CN=Services,CN=Configuration
```

El PowerShell script disponible en el repositorio oficial de Connector for Active Directory se puede utilizar para delegar los permisos adicionales necesarios para la cuenta de servicio AD Connector de Directory Service.

Este script crea el objeto NTAuth de entidad emisora de certificados.

Para obtener la versión más reciente del script y los detalles de uso, consulte el archivo README del GitHub repositorio.

Paso 4: Crear una política de IAM

Para crear un conector para AD, necesita una política de IAM que le permita crear recursos de conectores, compartir su CA privada con el servicio del Conector para AD y autorizar el servicio del Conector para AD con su directorio.

Este es un ejemplo de política administrada por el usuario:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "pca-connector-ad:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListCertificateAuthorities",
                "acm-pca:ListTags",
                "acm-pca:PutPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "acm-pca:IssueCertificate",
```

```
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APIPassthrough/V*"
                },
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "pca-connector-ad.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ds:AuthorizeApplication",
                "ds:DescribeDirectories",
                "ds:ListTagsForResource",
                "ds:UnauthorizeApplication",
                "ds:UpdateAuthorizedApplication"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ec2:DeleteTags",
                "ec2:CreateTags"
            ],
            "Resource": "arn:*:ec2:*:*:vpc-endpoint/*"
        }
    ]
```

```
}
```

Connector for AD requiere AWS RAM permisos adicionales, tanto para su uso en la consola como en la línea de comandos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ram:CreateResourceShare",
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "ram:Principal": "pca-connector-ad.amazonaws.com",
                    "ram:RequestedResourceType": "acm-pca:CertificateAuthority"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:GetResourcePolicies",
                "ram:GetResourceShareAssociations",
                "ram:GetResourceShares",
                "ram:ListPrincipals",
                "ram:ListResources",
                "ram:ListResourceSharePermissions",
                "ram:ListResourceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

Paso 5: Comparta su CA privada con Connector para AD

Deberá compartir su CA privada con el servicio de conectores mediante el uso compartido del principal del AWS Resource Access Manager servicio.

Al crear un conector en la AWS consola, el recurso compartido se crea automáticamente.

Al crear un recurso compartido mediante el AWS CLI, utilizará el AWS RAM create-resource-share comando.

Ejecute los siguientes comandos para crear un recurso compartido:

```
$  aws ram create-resource-share \
    --region us-east-1 \
    --name MyPcaConnectorAdResourceShare \
    --permission-arns arn:aws:ram::aws:permission/
AWSRAMBlankEndEntityCertificateAPIPassthroughIssuanceCertificateAuthority \
    --resource-arns arn:aws:acm-pca:region:account:certificate-authority/CA_ID \
    --principals pca-connector-ad.amazonaws.com \
    --sources account
```

El director del servicio que llama CreateConnector tiene permisos de emisión de certificados en la PCA. Para evitar que los directores de servicio que utilizan el Conector para AD tengan acceso general a sus recursos de Autoridad de certificación privada de AWS , restrinja sus permisos mediante `CalledVia`.

Paso 6: Crear un registro de directorio

Usted autoriza el servicio del Conector para AD con su directorio para que el conector pueda comunicarse con su directorio. Para autorizar el servicio del Conector para AD, debe crear un registro de directorio. Para obtener más información acerca de la creación de un registro de directorio, consulte Administrar registros de directorios

Paso 7: Configurar grupos de seguridad

La comunicación entre la VPC y el conector Connector for AD es directa AWS PrivateLink, lo que requiere un grupo o grupos de seguridad con reglas de entrada que abran el puerto 443 TCP de la VPC. Se le solicitará este grupo de seguridad cuando cree un conector. Puede especificar la fuente como personalizada y seleccionar el bloque CIDR de su VPC. Puede optar por restringirlo aún más (es decir, IP, CIDR e ID de grupo de seguridad).

Paso 8: Configurar el acceso a la red para los objetos de directorio

Los objetos de directorio requieren acceso público a Internet para validar el Protocolo de estado de certificados en línea (OCSP) y las listas de revocación de certificados (CRLs) de los siguientes dominios:

```
*.windowsupdate.com
*.amazontrust.com
```

Reglas de acceso mínimas requeridas:

- Necesario para la comunicación entre OCSP y CRL:

  ```
  TCP 80: (HTTP) to 0.0.0.0/0
  ```

- Necesario para el conector AD:

  ```
  TCP 443: (HTTPS) to 0.0.0.0/0
  ```

- Necesario para Active Directory:

  ```
  TCP 88: (Kerberos) to Domain Controller IP range
  TCP/UDP 389/636: (LDAP/LDAPS) to Domain Controller IP range, depending on Domain
   Controller configuration
  TCP/UDP 53: (DNS) to 0.0.0.0/0
  ```

Si los dispositivos no tienen acceso público a Internet, la emisión del certificado fallará de forma intermitente con el código de error `WS_E_OPERATION_TIMED_OUT.`

> **Note**
>
> Si está configurando un grupo de seguridad para una instancia de Amazon EC2, no tiene por qué ser el mismo del paso 7.

Comience a utilizar AWS Private CA Connector for Active Directory

Con AWS Private CA Connector for Active Directory, puede emitir certificados desde su entidad de certificación privada a sus objetos de Active Directory para su autenticación y cifrado. Al crear

un conector, AWS Private Certificate Authority crea un punto de conexión en la VPC para que los objetos del directorio soliciten certificados.

Para emitir certificados, debe crear un conector y plantillas compatibles con AD para el conector. Al crear una plantilla, puede establecer los permisos de inscripción para sus grupos de AD.

Temas

- Antes de empezar
- Paso 1: Crear un conector
- Paso 2: Configurar las políticas de Microsoft Active Directory
- Paso 3: Crea una plantilla
- Paso 4: Configurar los permisos de grupo de Microsoft

Antes de empezar

El siguiente tutorial le guía por el proceso de creación de un conector para AD y una plantilla de conector. Para seguir este tutorial, primero debe cumplir los requisitos previos que se enumeran en la sección.

Paso 1: Crear un conector

Para crear un conector, consulteCreación de un conector para Active Directory.

Paso 2: Configurar las políticas de Microsoft Active Directory

El Conector para AD no puede ver ni administrar la configuración del objeto de política de grupo (Group Policy Object, GPO) del cliente. El GPO controla el enrutamiento de las solicitudes de AD al servidor del cliente Autoridad de certificación privada de AWS o a otros servidores de autenticación o venta de certificados. Una configuración de GPO no válida puede provocar que sus solicitudes se enruten incorrectamente. Los clientes deben configurar y probar la configuración del Conector para AD.

Las políticas de grupo están asociadas a un conector y puede optar por crear varios conectores para un único AD. Depende de usted administrar el control de acceso a cada conector si sus configuraciones de políticas de grupo son diferentes.

La seguridad de las llamadas al plano de datos depende de Kerberos y de la configuración de la VPC. Cualquier persona con acceso a la VPC puede realizar llamadas al plano de datos siempre

que esté autenticada en el AD correspondiente. Esto existe fuera de los límites AWSAuth y la administración de la autorización y la autenticación depende de usted, el cliente.

Cuando AWS Managed Microsoft AD lo utilices, usa el Directory Service enable-ca-enrollment-policy comando para GPOs configurarlo en el controlador de dominio de la AWS Managed Microsoft AD instancia.

El siguiente comando permite inscribir controladores de dominio:

```
$  aws ds enable-ca-enrollment-policy \
    --pca-connector-arn MyPcaConnectorAdArn \
    --directory-id MyDirectoryId
```

Cuando utilices AD Connector, sigue estos pasos para crear un GPO que apunte al URI generado al crear un conector. Este paso es necesario para usar el Conector para AD desde la consola o la línea de comandos.

Configurar GPOs.

1. Abra el Administrador de servidores en el DC
2. Acceda a Herramientas y elija Administración de políticas de grupo en la esquina superior derecha de la consola.
3. Acceda a Bosque > Dominios. Seleccione su nombre de dominio y haga clic con el botón derecho en su dominio. Seleccione Crear un GPO en este dominio y vincúlelo aquí... e introduzca `PCA GPO` para el nombre.
4. El GPO recién creado aparecerá ahora bajo su nombre de dominio.
5. Elija PCA GPO y seleccione Editar. Si se abre un cuadro de diálogo con el mensaje de alerta Este es un enlace y los cambios se propagarán globalmente, confirme el mensaje para continuar. Debería abrirse el Editor de administración de políticas de grupo.
6. En el Editor de administración de políticas de grupo, acceda a Configuración del equipo > Políticas > Configuración de Windows > Configuración de seguridad > Políticas de clave pública (elija la carpeta).
7. Acceda al tipo de objeto y elija Cliente de servicios de certificación: política de inscripción de certificados
8. En las opciones, cambie el Modelo de configuración a Habilitado.
9. Confirme que la Política de inscripción de Active Directory esté marcada y habilitada. Elija Añadir.

10. Debería abrirse la ventana Servidor de políticas de inscripción de certificados.
11. Introduzca el punto de conexión del servidor de políticas de inscripción de certificados que se generó al crear el conector en el campo Introduzca el URI de la política del servidor de inscripciones.
12. Deje el tipo de autenticación como Windows integrado.
13. Elija Validar. Una vez que la validación se haya realizado correctamente, seleccione Agregar. Se cierra el cuadro de diálogo.
14. Vuelva a la Cliente de servicios de certificación: Política de inscripción de certificados y marque la casilla situada junto al conector recién creado para asegurarse de que se trata de la política de inscripción predeterminada
15. Elija la Política de inscripción de Active Directory y seleccione Eliminar.
16. En el cuadro de diálogo de confirmación, elija Sí para eliminar la autenticación basada en el LDAP.
17. Seleccione Aplicar y Aceptar en la ventana Cliente de servicios de certificación > Política de inscripción de certificados y ciérrela.
18. Vaya a la carpeta Políticas de clave pública y elija Cliente de servicios de certificación: inscripción automática.
19. Cambie el Modelo de configuración a Habilitado.
20. Confirme que las opciones Renovar certificados caducados y Actualizar certificados estén marcadas. Deje las otras opciones como están.
21. Seleccione Aplicar, luego Aceptar y cierre el cuadro de diálogo.

A continuación, configure las políticas de claves públicas para la configuración del usuario. Acceda a Configuración de usuario > Políticas > Configuración de Windows > Configuración de seguridad > Políticas de claves públicas. Siga los procedimientos descritos desde el paso 6 hasta el paso 21 para configurar las políticas de claves públicas para la configuración del usuario.

Cuando haya terminado de configurar GPOs las políticas de clave pública, los objetos del dominio solicitarán los certificados de Autoridad de certificación privada de AWS Connector for AD y recibirán los certificados emitidos por él Autoridad de certificación privada de AWS.

Paso 3: Crea una plantilla

Para crear una plantilla, consulteCrear una plantilla de conector.

Paso 4: Configurar los permisos de grupo de Microsoft

Para configurar los permisos de grupo de Microsoft, consulteAdministre las entradas de control de acceso de la plantilla Connector for AD.

AWS Private CA conectores para Active Directory

Los procedimientos de esta sección describen cómo crear conectores de Active Directory (AD), configurar plantillas e integrarlos con Autoridad de certificación privada de AWS Active Directory. Puede realizar estas operaciones desde la consola de Autoridad de certificación privada de AWS Connector for AD, mediante la sección Connector para AD de la AWS CLI API Connector for AD o mediante la API de Autoridad de certificación privada de AWS Connector for AD.

> **Note**
>
> Si bien Autoridad de certificación privada de AWS Connector for AD está estrechamente integrado con Autoridad de certificación privada de AWS ella, los dos servicios son independientes APIs. Para obtener más información, consulte la referencia de la API de AWS Private Certificate Authority de la Referencia de la API del Conector Autoridad de certificación privada de AWS para Active Directory.

Creación de un conector para Active Directory

Utilice los siguientes procedimientos para crear un conector mediante la consola, la línea de comandos o la API de AWS Private CA Connector for Active Directory.

Console

Para crear un conector mediante la consola

Inicie sesión en su AWS cuenta y abra la consola de AWS Private CA Connector for Active Directory en**`https://console.aws.amazon.com/pca-connector-ad/home`**.

1. En la página de inicio del primer servicio o en la página Conectores para Active Directory, elija Crear conector.
2. En la página Crear un conector CA privado para Active Directory, proporcione información en la sección Active Directory.

- En Seleccionar el tipo de Active Directory, elija uno de los dos tipos disponibles:
  - AWS Directory Service for Microsoft Active Directory— Especifica un Active Directory administrado por Directory Service.
  - Active Directory en las instalaciones con conector AD AWS : utiliza el Conector AD para acceder a un Active Directory que se aloja en las instalaciones.
- En Seleccionar su directorio, elija su directorio de la lista.

  Como alternativa, puede elegir Crear directorio, que abre la Directory Service consola en una ventana nueva. Cuando termine de crear un directorio nuevo, vuelva a la consola de AWS Private CA Connector for Active Directory y actualice la lista de directorios. El nuevo directorio debería estar disponible para su selección.

  > **Note**
  >
  > Al crear un directorio, tenga en cuenta que el Conector para AD solo admite los siguientes tipos de directorios que se ofrecen en la consola de Directory Service :
  >
  > - AWS Microsoft AD gestionado
  > - Conector de AD

- En Seleccionar grupos de seguridad para el punto de conexión de VPC, elija un grupo de seguridad de la lista.

  Como alternativa, puedes elegir Crear grupo de seguridad, que abre la EC2 consola de Amazon en la página Crear grupo de seguridad en una ventana nueva. Cuando termine de crear un grupo de seguridad, vuelva a la consola de AWS Private CA Connector for Active Directory y actualice la lista de grupos de seguridad. El nuevo grupo de seguridad debería estar disponible para su selección.

3. En la sección de tipo de dirección IP, elija una de las siguientes opciones:

   - IPv4- Habilita la IPv4 conectividad con el servicio. Elija esta opción solo si todas las subredes que alojan su directorio tienen rangos de IPv4 direcciones.
   - Dualstack: habilita tanto la IPv6 conectividad con el IPv4 servicio como la misma. Elija esta opción solo si todas las subredes que alojan su directorio tienen ambos IPv4 rangos de direcciones. IPv6

4. En la sección Autoridad de certificación privada, elija una CA privada de la lista.

Como alternativa, puede elegir Crear una CA privada, que abre la Autoridad de certificación privada de AWS consola en la página de autoridades de certificación privadas en una ventana nueva. Cuando termine de crear una CA, vuelva a la consola de AWS Private CA Connector for Active Directory y actualice la lista de CAs. El nuevo CA debería estar disponible para su selección.

5. En el panel Etiquetas: opcional, puede aplicar y eliminar metadatos en su recurso de AD. Las etiquetas son pares de cadenas clave-valor en las que la clave debe ser exclusiva del recurso y el valor es opcional. El panel muestra todas las etiquetas existentes para el recurso en una tabla. Se admiten las siguientes acciones.

   - Seleccione Administrar etiquetas para abrir la página Administrar etiquetas.
   - Para crear una etiqueta, elija Crear etiqueta nueva. Complete el campo Clave y, si lo desea, el campo Valor. Para aplicar la etiqueta, elija Guardar los cambios.
   - Pulse el botón Eliminar situado junto a una etiqueta para marcarla para su eliminación y seleccione Guardar cambios para confirmar.

6. Tras proporcionar la información requerida y revisar las opciones, seleccione Crear conector. Se abrirá la página de detalles de Conectores para Active Directory, donde podrá ver el progreso del conector a medida que se vaya creando.

   Una vez finalizado el proceso de creación de un conector, asígnele un nombre de entidad principal de servicio.

API

Para crear un conector mediante la API

Para crear un conector para Active Directory con la API, utilice la CreateConnectoracción de la API AWS Private CA Connector for Active Directory.

CLI

Para crear un conector mediante AWS CLI

Para crear un conector para Active Directory con la CLI, utilice el comando create-connector de la sección AWS Private CA Conector para Active Directory de. AWS CLI

Crear una plantilla de conector

Una plantilla es una lista de configuraciones sobre el aspecto que debe tener el certificado una vez emitido y el modo en que el cliente debe gestionarlos. Los siguientes procedimientos explican cómo crear una plantilla.

Console

Para crear una plantilla mediante la consola

1. Inicie sesión en su AWS cuenta y abra la consola de AWS Private CA Connector for Active Directory en**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Elija un conector de la lista de Conectores para Active Directory y, a continuación, seleccione Ver detalles.
3. En la página de detalles del conector, busque la sección Plantillas y, a continuación, elija Crear plantilla.
4. En la página Crear plantilla, en la sección Método de creación de plantillas, elija una de las opciones del método.

   - Comenzar con una plantilla predefinida (predeterminada): elija de una lista de plantillas predefinidas para las aplicaciones de AD:
     - Firma de código
     - Computadora
     - Autenticación del controlador de dominio
     - Agente de recuperación EFS
     - Agente de inscripción
     - Agente de inscripción (computadora)
     - IPSec
     - Autenticación Kerberos
     - Servidor RAS e IAS
     - Inicio de sesión en Smartcard
     - Firma de la lista de confianza
     - Firma de usuario
     - Autenticación de la estación de trabajo

- Comenzar a partir de una plantilla existente que creó: elija de una lista de plantillas personalizadas que haya creado anteriormente.
- Comenzar desde una plantilla en blanco: elija esta opción para empezar a crear una plantilla completamente nueva.

5. En la sección Configuración de certificados, defina la siguiente configuración para los certificados basados en esta plantilla.

- Tipo de certificado: especifique si desea crear certificados de usuario o de equipo.
- Inscripción automática: elija si desea activar la inscripción automática para los certificados basados en esta plantilla.
- Período de validez: especifique un periodo de validez como un valor entero de horas, días, semanas o años. El valor mínimo es 2 horas.
- Período de renovación: especifique un periodo de renovación del certificado como un valor entero de horas, días, semanas o años. El período de renovación no debe superar el 75% del período de validez.
- Nombre del asunto: elija una o más opciones para incluirlas en el nombre del asunto conforme a la información contenida en Active Directory.

> **Note**
>
> Debe especificarse al menos un nombre de asunto o una opción de nombre alternativo del asunto.

  - Nombre común
  - DNS como nombre común
  - Ruta del directorio
  - Correo electrónico
- Nombre alternativo del asunto: elija una o más opciones para incluirlas en el nombre alternativo del asunto conforme a la información contenida en Active Directory.

> **Note**
>
> Debe especificarse al menos un nombre de asunto o una opción de nombre alternativo del asunto.

- GUID de directorio
- Nombre de DNS
- DNS de dominio
- Correo electrónico
- Nombre de la entidad principal del servicio (SPN)
- Nombre de la entidad principal del usuario (UPN)

6. En la sección Opciones de inscripción y gestión de solicitudes de certificados, especifique el propósito de los certificados en función de la plantilla y elija una de las siguientes opciones.

   - Signature
   - Cifrado
   - Firma y cifrado
   - Inicio de sesión con firma y smartcard

   A continuación, elija cuál de las siguientes funciones desea activar. Las opciones varían en función del propósito del certificado.

   - Elimine los certificados no válidos (no los archive)
   - Incluya algoritmos simétricos
   - Clave privada exportable

   Por último, elija una opción de inscripción de certificados. Las opciones varían en función del propósito del certificado.

   - No es necesaria la entrada del usuario
   - Avise al usuario durante la inscripción
   - Indique al usuario durante la inscripción y solicite la entrada del usuario

7. En la sección Políticas de aplicación, elija todas las políticas de aplicación que correspondan. Las políticas disponibles se enumeran en varias páginas. Es posible que algunas políticas estén preseleccionadas debido a la configuración anterior.

8. En la sección Políticas de aplicación personalizadas, puede añadir políticas personalizadas OIDs a la plantilla y especificar si las extensiones de las políticas de aplicación son imprescindibles.
9. En la sección Configuración de criptografía, elija las siguientes categorías de configuración de criptografía para los certificados basados en esta plantilla.
10. En la sección Grupos y permisos, puede ver las plantillas, los grupos existentes y los permisos para la inscripción o puede pulsar el botón Añadir nuevos grupos y permisos para agregar nuevos grupos y permisos. El botón abre un formulario que requiere la siguiente información:

    - Display name (Nombre de visualización)
    - Identificador de seguridad (SID)
    - Inscríbase con las opciones PERMITIR | DENEGAR | NO ESTABLECER
    - Inscríbase automáticamente con las opciones PERMITIR | DENEGAR | NO ESTABLECER
11. En la sección Reemplazar plantillas, puede notificar a Active Directory que la plantilla actual reemplaza a una o más plantillas creadas en AD. Para aplicar la plantilla sustitutiva, seleccione Añadir plantilla de Active Directory para sustituirla y especifique el nombre común de la plantilla sustitutiva.
12. En el panel Etiquetas: opcional, puede aplicar y eliminar metadatos en su recurso de AD. Las etiquetas son pares de cadenas clave-valor en las que la clave debe ser exclusiva del recurso y el valor es opcional. El panel muestra todas las etiquetas existentes para el recurso en una tabla. Se admiten las siguientes acciones.

    - Seleccione Administrar etiquetas para abrir la página Administrar etiquetas.
    - Para crear una etiqueta, elija Crear etiqueta nueva. Complete el campo Clave y, si lo desea, el campo Valor. Para aplicar la etiqueta, elija Guardar los cambios.
    - Pulse el botón Eliminar situado junto a una etiqueta para marcarla para su eliminación y seleccione Guardar cambios para confirmar.
13. Tras proporcionar la información requerida y revisar las opciones, seleccione Crear plantilla. Se abrirá Detalles de la plantilla, donde podrá revisar la configuración de la nueva plantilla, editarla o eliminarla, gestionar los grupos y los permisos, gestionar las plantillas sustituidas, gestionar las etiquetas y configurar la reinscripción automática para los titulares de certificados.

API

Para crear una plantilla de conector mediante la API

Utilice la CreateTemplateacción en la API de AWS Private CA Connector for Active Directory.

CLI

Para crear una plantilla de conector mediante el AWS CLI

Utilice el comando create-template de la sección AWS Private CA Connector para Active Directory del. AWS CLI

Actualizar una plantilla para Active Directory

Utilice los siguientes procedimientos para actualizar una plantilla mediante la consola, la línea de comandos o la API de AWS Private CA Connector for Active Directory.

Console

Para actualizar una plantilla mediante la consola

Inicie sesión en su AWS cuenta y abra la consola de AWS Private CA Connector for Active Directory en**https://console.aws.amazon.com/pca-connector-ad/home**.

1. En la lista de conectores para Active Directory, seleccione el conector cuya plantilla desee actualizar. Elija Editar para ver y modificar las plantillas del conector.
2. En la página de detalles de la plantilla del conector, seleccione Editar. Siga las instrucciones para realizar las actualizaciones. Cuando hayas terminado de editar un área, selecciona Guardar para guardar los cambios.

API

Para actualizar una plantilla mediante la API

Para actualizar una plantilla de Active Directory con la API, utilice la UpdateTemplateacción de la API AWS Private CA Connector for Active Directory.

CLI

Para actualizar una plantilla mediante la AWS CLI

Para actualizar un conector para Active Directory con la CLI, utilice el comando update-template de la sección AWS Private CA Conector para Active Directory de la. AWS CLI

Cómo propaga Connector for Active Directory los cambios de plantilla

AWS Private CA aplica la plantilla a la política cuando el cliente actualiza la caché de la política, es decir, cada ocho horas. Esto incluye cambios en las entradas de control de acceso del grupo de plantillas. Cuando el cliente actualiza la memoria caché, consulta el conector para ver las plantillas disponibles. En el caso de la actualización de la inscripción automática, el cliente emite certificados que cumplen una o ambas de las siguientes condiciones:

- El certificado se encuentra dentro del período de renovación.
- El certificado no está presente en el dispositivo cliente.

Para la actualización manual, el cliente consultará el conector y usted deberá configurar la plantilla para que se emita.

Si está depurando, puede borrar manualmente la caché de políticas para ver inmediatamente los cambios en la plantilla. Para ello, ejecuta el siguiente comando de Powershell en tu cliente.

```
certutil -f -user -policyserver * -policycache delete
```

Listar conectores para Active Directory

Puede usar la consola AWS Private CA Connector for Active Directory o AWS CLI para enumerar los conectores que le pertenecen.

Console

Para enumerar los conectores mediante la consola

1. Inicie sesión en su AWS cuenta y abra la consola de AWS Private CA Connector for Active Directory en**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Revise la información de la lista de Conectores para Active Directory. Puede explorar diferentes páginas de conectores utilizando los números de página de la parte superior derecha. Cada conector ocupa una fila que muestra las siguientes columnas de información de forma predeterminada.

- ID del conector: el ID único del conector.
- Nombre del directorio: el recurso de Active Directory asociado al conector.
- Estado del conector: estado del conector. Los valores posibles son Crear | Activo | Eliminar | Fallido.
- Estado del nombre de la entidad principal del servicio: estado del nombre principal del servicio (SPN) asociado al conector. Los valores posibles son Crear | Activo | Eliminar | Fallido.
- Estado de registro en el directorio: estado de registro del director asociado. Los valores posibles son Crear | Activo | Eliminar | Fallido.
- Creado en: marca de tiempo en el momento de la creación del conector.

Mediante el icono de engranaje situado en la esquina superior derecha de la consola, puede personalizar el número de conectores que se muestran en una página mediante la preferencia de tamaño de página.

API

Para enumerar los conectores mediante la API

Utilice la ListConnectorsacción en la API de AWS Private CA Connector for Active Directory.

CLI

Para enumerar los conectores mediante el AWS CLI

Utilice el comando list-connectors para enumerar los conectores.

Listar plantillas de conectores

Puede usar la consola AWS Private CA Connector para Active Directory o AWS CLI para enumerar las plantillas de los conectores de su propiedad. Las plantillas de conectores se basan en las plantillas AWS Private CA _BlankEndEntityCertificateAPIPassthrough_/V1.

Console

Para actualizar el oyente desde la consola

1. Inicie sesión en su AWS cuenta y abra la consola de AWS Private CA Connector for Active Directory en**https://console.aws.amazon.com/pca-connector-ad/home**.

2. Elija un conector de la lista de Conectores para Active Directory y, a continuación, seleccione Ver detalles.
3. En la página de detalles del conector, revise la información de la sección Plantillas. Puede explorar diferentes páginas de certificados utilizando los números de página de la parte superior derecha. Cada plantilla ocupa una fila que muestra las siguientes columnas de información.

- Nombre de la plantilla: el nombre de la plantilla en lenguaje natural.
- Estado de la plantilla: estado de la plantilla. Los valores posibles son: Activo | Eliminar.
- ID de plantilla: el identificador único de la plantilla.

API

Para enumerar los conectores mediante la API

Utilice la ListTemplatesacción de la API de AWS Private CA Connector for Active Directory para mostrar las plantillas del conector especificado.

CLI

Para enumerar los conectores mediante el AWS CLI

Utilice el comando list-templates para enumerar las plantillas del conector especificado.

Ver los detalles del conector

Utilice los siguientes procedimientos para ver los detalles de configuración de un conector en la consola, la línea de comandos o la API de AWS Private CA Connector for Active Directory.

Console

Para ver los detalles de un conector mediante la consola

1. Inicie sesión en su AWS cuenta y abra la consola de AWS Private CA Connector for Active Directory en**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Elija un conector de la lista de Conectores para Active Directory y, a continuación, seleccione Ver detalles.

3. En la página de detalles del conector, revise la información del panel de detalles del conector, que incluye lo siguiente:
   - ID del conector
   - Estado del conector
   - Detalles de estado adicionales
   - Conector ARN
   - Punto de conexión del servidor de políticas de inscripción de certificados
   - Nombre del directorio
   - ID de directorio
   - Autoridad de certificación privada de AWS asunto
   - Autoridad de certificación privada de AWS estado
   - Tipo de dirección IP
   - Punto de conexión de VPC y grupos de seguridad
4. En el panel Plantillas, puede crear o administrar las plantillas asociadas al conector.
5. En el panel Nombre de la entidad principal del servicio (Service principal name, SPN), puede ver el nombre principal del servicio asociado al conector.
6. En el panel de Registro de directorios, puede ver o cambiar el registro de directorio asociado al conector.
7. En el panel Etiquetas: opcional, puede crear o administrar las etiquetas asociadas al conector.

API

Para enumerar los conectores mediante la API

Utilice la GetConnectoracción en la API de AWS Private CA Connector for Active Directory.

CLI

Para enumerar los conectores mediante el AWS CLI

Utilice el comando get-connector de la sección AWS Private CA Connector para Active Directory del. AWS CLI

Ver detalles de la plantilla de conectores

Utilice los siguientes procedimientos para ver los detalles de configuración de una plantilla de conector mediante la consola, la línea de comandos o la API de AWS Private CA Connector for Active Directory

Console

Para ver los detalles de una plantilla de conector mediante la consola

1. Inicie sesión en su AWS cuenta y abra la consola de AWS Private CA Connector for Active Directory en**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Elija un conector de la lista de Conectores para Active Directory y, a continuación, seleccione Ver detalles.
3. En la página de detalles del conector, revise la información de la sección Plantillas y seleccione la plantilla que desee inspeccionar. Elija Ver detalles.
4. En la página de detalles, el panel Detalles de la plantilla muestra la siguiente información sobre la plantilla:

   - Nombre de la plantilla
   - ID de plantilla
   - Estado de la plantilla
   - Versión del esquema de la plantilla
   - Versión de plantilla
   - ARN de plantilla
   - Tipo de certificado
   - La inscripción automática está activada
   - Periodo de validez
   - Período de renovación
   - Requisitos de nombre de asunto
   - Requisitos de nombre alternativo del asunto
   - Configuración de solicitud e inscripción de certificados
   - Categoría de proveedor de criptografía
   - Algoritmo clave

- Tamaño mínimo de clave (bits)
- Algoritmo hash
- Proveedores de criptografía
- Configuración de extensión de uso de clave

Desde este panel, también puede realizar las siguientes acciones mediante los botones Editar, Eliminar y Acciones.

- Editar
- Delete
- Administrar grupos y permisos: para obtener más información, consulte Configurar grupos y permisos.
- Administrar plantillas sustituidas: para obtener más información, consulte Revisar y crear.
- Administrar etiquetas: para obtener más información, consulte Etiquetado de los recursos del Conector para AD.
- Volver a inscribir a todos los titulares de certificados: esta configuración permite aumentar automáticamente la versión principal de una plantilla. Todos los miembros de los grupos de Active Directory a los que se les permita inscribirse con una plantilla recibirán un nuevo certificado emitido con esa plantilla. Para obtener más información, consulte la API UpdateTemplate.

5. El panel inferior muestra una fila de pestañas que permiten realizar cambios en la configuración de la plantilla.

   - Grupos y permisos: vea y administre los permisos de los grupos de Active Directory para inscribir certificados con esta plantilla. Para obtener más información, consulte Configurar grupos y permisos
   - Políticas de aplicación: vea y administre las políticas de aplicaciones de plantilla. Para obtener más información, consulte Asignaciones de políticas de aplicaciones.
   - Plantillas reemplazadas: vea y administre las plantillas reemplazadas. Para obtener más información, consulte Revisar y crear.
   - Etiquetaopcional: ver y gestionar el etiquetado en esta plantilla. Para obtener más información, consulte Etiquetado de los recursos del Conector para AD.

API

Para enumerar los conectores mediante la API

Utilice la GetTemplateacción en la API de AWS Private CA Connector for Active Directory.

CLI

Para enumerar los conectores mediante el AWS CLI

Utilice el comando get-template de la sección AWS Private CA Connector para Active Directory del. AWS CLI

Administrar registros de directorios

Console

Para administrar los registros de directorios mediante la consola

Los registros de directorios para los conectores se pueden administrar desde el nivel superior de la consola del Conector AWS Private CA para Active Directory. En este tema se explican las opciones de administración disponibles.

1. Inicie sesión en su AWS cuenta y abra la consola de AWS Private CA Connector for Active Directory en**https://console.aws.amazon.com/pca-connector-ad/home**.
2. En el área de navegación de la izquierda, seleccione Registros en el directorio.
3. La página de registros del directorio muestra una tabla de directorios registrados con los siguientes campos:
   - ID de directorio: el ID único del directorio
   - Nombre del directorio: el nombre del sitio del dominio del directorio
   - Tipo de directorio
   - Registrado: el estado del registro. Los valores admitidos son CREAR | ACTIVO | ELIMINAR | FALLIDO.
   - Estado del directorio: el estado del directorio

   El usuario puede utilizar el directorio de registro para crear un registro nuevo.

4. Puede seleccionar uno de los registros de la lista para gestionarlo. Esto habilita los botones Ver detalles de registro y Anular el registro del directorio. El botón Ver detalles de registro abre la página de detalles del registro.
5. El panel Detalles de registro de directorio muestra la siguiente información:

   - Nombre del sitio del dominio del directorio
   - ID de directorio: el ID único del directorio. Si selecciona el enlace, accederá a la AWS Directory Service consola.
   - Tipo de directorio
   - Estado: estado del directorio
   - ARN de registro de directorio: el nombre del recurso de Amazon del registro de directorio
   - Información de estado adicional
6. En el panel Conectores y nombre principal del servicio (SPNs), puede administrar SPNs el conector. Para obtener más información, consulte Ver detalles del conector.
7. En el panel Etiquetas: opcional, puede aplicar y eliminar metadatos en su recurso de AD. Las etiquetas son pares de cadenas clave-valor en las que la clave debe ser exclusiva del recurso y el valor es opcional. El panel muestra todas las etiquetas existentes para el recurso en una tabla. Se admiten las siguientes acciones.

   - Seleccione Administrar etiquetas para abrir la página Administrar etiquetas.
   - Para crear una etiqueta, elija Crear etiqueta nueva. Complete el campo Clave y, si lo desea, el campo Valor. Para aplicar la etiqueta, elija Guardar los cambios.
   - Pulse el botón Eliminar situado junto a una etiqueta para marcarla para su eliminación y seleccione Guardar cambios para confirmar.

API

Para gestionar los registros de directorios mediante la API

Crear: CreateDirectoryRegistrationacción en la API de AWS Private CA Connector for Active Directory.

Recuperar: GetDirectoryRegistrationacción en la API de AWS Private CA Connector for Active Directory.

Lista: ListDirectoryRegistrationsacción en la API de AWS Private CA Connector for Active Directory.

Eliminar: DeleteDirectoryRegistrationacción en la API de AWS Private CA Connector for Active Directory.

CLI

Para administrar los registros de directorios mediante la CLI

Crear: utilice el create-directory-registrationcomando de la sección AWS Private CA Connector para Active Directory del AWS CLI.

Recuperar: get-directory-registrationcomando de la sección AWS Private CA Connector para Active Directory del AWS CLI.

Lista: list-directory-registrationscomando de la sección AWS Private CA Connector para Active Directory de la AWS CLI.

Eliminar: delete-directory-registrationcomando de la sección AWS Private CA Connector para Active Directory del AWS CLI.

Administre las entradas de control de acceso de la plantilla Connector for AD

Una entrada de control de acceso permite controlar qué grupos de Active Directory pueden o no inscribir certificados para una plantilla específica de Connector para AD. Cuando pueda crear o administrar grupos y permisos en Connector for AD, debe proporcionar el identificador de seguridad (SID) del objeto de grupo de Active Directory. Puede obtener el SID mediante el siguiente PowerShell comando. Para obtener más información SIDs, consulte Cómo funcionan los identificadores de seguridad en la documentación de los Servicios de dominio de Microsoft Directory.

```
$ Get-ADGroup -Identity "my_active_directory_group_name"
```

Los siguientes procedimientos ilustran cómo crear y administrar las entradas de los grupos de acceso a plantillas de Connector for AD.

Console

Para administrar los permisos de los grupos de plantillas mediante la consola

Puede administrar los grupos y los permisos de una plantilla existente desde la página de detalles de una plantilla. Para obtener más información, consulte Ver detalles de la plantilla de conectores.

Establezca los permisos para que los grupos puedan o no puedan inscribir certificados para la plantilla específica. Usted brinda el identificador del grupo de seguridad (SID). A continuación, configura los permisos de inscripción e inscripción automática para el grupo. Para la inscripción automática, tanto la inscripción como la inscripción automática deben estar configuradas en "Permitir".

API

Para gestionar los permisos de los grupos de plantillas mediante la API

Crear: CreateTemplateGroupAccessControlEntryacción en la API de AWS Private CA Connector for Active Directory.

Actualización: UpdateTemplateGroupAccessControlEntryacción en la API de AWS Private CA Connector for Active Directory.

Recuperar: GetTemplateGroupAccessControlEntryacción en la API de AWS Private CA Connector for Active Directory.

Lista: ListTemplateGroupAccessControlEntriesacción en la API de AWS Private CA Connector for Active Directory.

Eliminar: DeleteTemplateGroupAccessControlEntryacción en la API de AWS Private CA Connector for Active Directory.

CLI

Para administrar los permisos de los grupos de plantillas mediante la CLI

Cree el comando: create-template-group-access-control-entry en la sección AWS Private CA Connector para Active Directory del. AWS CLI

Actualización: comando update-template-group-access-control-entry en la sección AWS Private CA Connector para Active Directory del. AWS CLI

Recupere el comando: get-template-group-access-control-entry en la sección AWS Private CA Connector para Active Directory del. AWS CLI

Comando List: list-template-group-access-control-entries en la sección AWS Private CA Connector para Active Directory del. AWS CLI

Elimine el comando: delete-template-group-access-control-entries de la sección AWS Private CA Connector para Active Directory del. AWS CLI

Configuración del nombre de la entidad principal del servicio

Aprenda a configurar el nombre de la entidad principal del servicio para el conector.

Console

Para administrar los nombres principales de los servicios mediante la consola

El nombre de la entidad principal del servicio (SPN) de un conector AD existente se puede administrar desde la página de detalles del conector. Para obtener más información, consulte Administrar el registro del directorio Ver detalles del conector

API

Para administrar los nombres principales de los servicios mediante la API

Crear: CreateServicePrincipalNameacción en la API de AWS Private CA Connector for Active Directory.

Recuperar: GetServicePrincipalNameacción en la API de AWS Private CA Connector for Active Directory.

Lista: ListServicePrincipalNamesacción en la API de AWS Private CA Connector for Active Directory.

Eliminar: DeleteServicePrincipalNameacción en la API de AWS Private CA Connector for Active Directory.

CLI

Para administrar los nombres principales de los servicios mediante la CLI

Crear: create-service-principal-namecomando en la sección AWS Private CA Connector para Active Directory del AWS CLI.

Recupere: get-service-principal-namecomando en la sección AWS Private CA Connector para Active Directory del AWS CLI.

Lista: list-service-principal-namescomando de la sección AWS Private CA Connector para Active Directory de la AWS CLI.

Eliminar: delete-service-principal-namecomando de la sección AWS Private CA Connector para Active Directory del AWS CLI.

Etiquetado de los recursos del Conector para AD

Puede aplicar etiquetas a los conectores, plantillas y registros de directorios. El etiquetado agrega metadatos a un recurso que puede ayudar a la organización y la administración.

Console

Para administrar el etiquetado de recursos mediante la consola

El etiquetado de los recursos existentes se gestiona en la página de detalles del recurso. Para obtener más información, consulte los siguientes procedimientos:

- Ver detalles de la plantilla de conectores
- Administrar los registros de directorios

API

Para gestionar el etiquetado de recursos mediante la API

Etiqueta: TagResourceacción en la API de AWS Private CA Connector for Active Directory.

Listar etiquetas: ListTagsForResourceacción en la API de AWS Private CA Connector for Active Directory.

Desetiquetar: UntagResourceacción en la API de AWS Private CA Connector for Active Directory.

Importante: es aceptable usar etiquetas para etiquetar objetos que contengan información confidencial. No obstante, las etiquetas en sí no deberían contener información de identificación personal (PII) o información confidencial.

CLI

Para administrar el etiquetado de recursos mediante la CLI

Etiqueta: comando tag-resource de la sección AWS Private CA Connector para Active Directory del. AWS CLI

Listar etiquetas: list-tags-for-resourcecomando en la sección AWS Private CA Connector para Active Directory del. AWS CLI

Comando Untag: untag-resource de la sección AWS Private CA Connector for Active Directory del. AWS CLI

Integración de Connector for AD en aplicaciones basadas en eventos mediante Amazon EventBridge

Puede incorporar Connector for AD en aplicaciones basadas en eventos (EDAs) que utilizan los eventos que se producen en Connector for AD para comunicarse entre los componentes de la aplicación e iniciar procesos posteriores.

Por ejemplo, puede invocar otros AWS servicios o componentes personalizados cuando se produzcan en su cuenta los siguientes eventos de Connector para AD:

- Se crea un certificado o cuando se produce un error en la creación.
- Se ha inscrito un certificado o se produce un error en la inscripción.

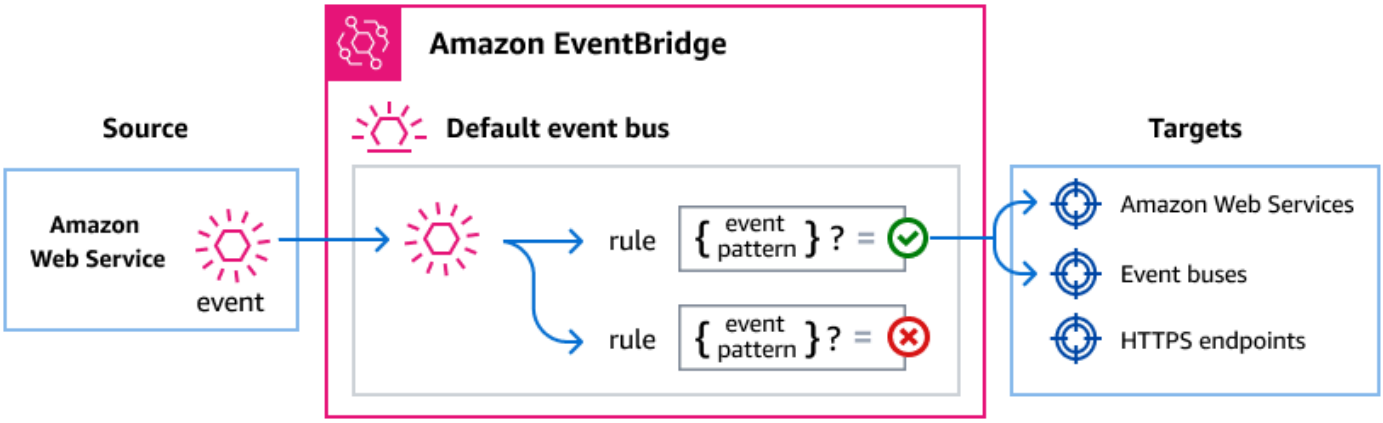
Para ello, utilice Amazon EventBridge para enrutar eventos desde Connector for AD a otros componentes de software. Amazon EventBridge es un servicio sin servidor que utiliza eventos para conectar los componentes de la aplicación entre sí, lo que facilita la integración de AWS servicios como Connector for AD en arquitecturas basadas en eventos sin necesidad de código ni operaciones adicionales.

¿Cómo direcciona los eventos de EventBridge Connector for AD

Así es como EventBridge funciona con los eventos de Connector for AD:

Como ocurre con muchos AWS servicios, Connector for AD genera y envía eventos al bus de eventos EventBridge predeterminado. Un bus de eventos es un enrutador que recibe eventos y los envía a destinos u objetivos, que se especifican. Los objetivos pueden incluir otros AWS servicios, aplicaciones personalizadas y aplicaciones de socios de SaaS.

EventBridge enruta los eventos de acuerdo con las reglas que usted crea en el bus de eventos. Para cada regla, se debe especificar un filtro o patrón de eventos para seleccionar solo los eventos que desee. Cada vez que se envía un evento al bus de eventos, lo EventBridge compara con cada regla. Si el evento coincide con la regla, lo EventBridge enruta a los objetivos especificados.



Conector para eventos de AD

Para obtener una lista de los eventos de Connector para AD enviados a EventBridge, consulte el tema Connector para AD en la Referencia de EventBridge eventos.

Estructura de evento

Todos los eventos de AWS los servicios contienen dos tipos de datos:

- Todos los eventos de los servicios de tienen un conjunto común de campos que contienen metadatos sobre el evento, como el servicio de AWS que es el origen del evento, la hora en que se generó el evento, la cuenta y la región en las que tuvo lugar el evento, etc. Para ver las definiciones de estos campos generales, consulte Estructura de eventos en Amazon EventBridge Events Reference.
- Un campo `detail` que contiene los datos específicos de ese evento en particular.

Crear patrones de eventos que coincidan con los eventos de Connector for AD

Los patrones de eventos son filtros que especifican qué datos deben contener los eventos que desea seleccionar.

Cada patrón de eventos es un objeto JSON que contiene:

- Un atributo `source` que identifica el servicio que envía el evento. En el caso de los eventos de Connector for AD, la fuente es`aws.pca-connector-ad`.

- (Opcional): un atributo `detail-type` que contiene una matriz de los nombres de eventos que deben coincidir.
- (Opcional): un atributo `detail` que contiene cualquier otro dato de evento con el que coincidir.

Por ejemplo, el siguiente patrón de eventos seleccionaría todos los eventos de Connector for AD que se hayan registrado correctamente en la política de certificados:

```
{
  "source": ["aws.pca-connector-ad"],
  "detail-type": ["Certificate Policy Enrollment Succeeded"]
}
```

Para obtener más información sobre la escritura de los patrones de eventos, consulte Patrones de eventos en la Guía del usuario de EventBridge .

Recibir eventos de EventBridge

Puede especificar los certificados de Connector for AD como destino de una regla. Esto permite a Connector for AD recibir eventos de una amplia variedad de fuentes, incluidos otros AWS servicios, aplicaciones personalizadas y socios de SaaS. Para obtener más información, consulte Crear reglas que reaccionen a los eventos en la Guía del EventBridge usuario.

Para obtener una lista completa de los AWS servicios que puede especificar como objetivos, consulte los tipos de objetivos en la Referencia de EventBridge eventos.

Solución de problemas con AWS Private CA Connector for Active Directory

Utilice la información que aparece aquí para diagnosticar y solucionar los problemas de AWS Private Certificate Authority Connector for AD.

Temas

- Solucione los problemas de los códigos de error de Connector for AD
- Solucione los errores de creación del conector AD en el conector
- Solucione el error de creación del SPN de AD en el conector
- Soluciona problemas de actualización de plantillas de Connector for AD

Solucione los problemas de los códigos de error de Connector for AD

Connector para AD envía mensajes de error por varios motivos. Para obtener información sobre cada error y recomendaciones para resolverlos, consulte la siguiente tabla. Puede recibir estos errores suscribiéndose a los eventos de Amazon EventBridge Scheduler (fuente del evento:`aws.pca-connector-ad`) o mediante la inscripción manual en Windows.

Código de error	Causa raíz	Corrección
0x8FFFA000	Error en la autenticación de Kerberos.	Asegúrese de que se pueda acceder a su directorio y de que el cliente sea un usuario o un ordenador. Si utilizas la inscripción automática, corrige el principal de tu servicio AWS de recursos. Si utiliza la interfaz de usuario de Active Directory para obtener un certificado, ejecute `gpupdate /force` .
0x8FFFA001	El mensaje SOAP debe contener un encabezado de acción.	Agregue una acción de encabezado.
0x8FFFA002	El conector no tiene acceso a la CA privada a la que está conectado.	Comparta su CA privada con el conector creando un Resource Access Manager (RAM) de AWS para compartir entre su CA privada y el servicio del Conector para AD.
0x8FFFA003	La CA privada de este conector no está activa.	Mueva la CA privada al estado Activo. Si su CA privada está en estado de certificado pendiente, instale el certificado de CA.
0x8FFFA004	La CA privada de este conector no existe.	Mueva la entidad de certificación al estado Activo si se encuentra

Código de error	Causa raíz	Corrección
		en el estado Eliminado. Si su CA privada se elimina permanentemente, cree un nuevo conector con una CA diferente.
0x8FFFA005	La plantilla especificó el atributo `directoryGuid` del asunto del certificado o su nombre alternativo, pero el atributo no se encontró en el objeto AD del solicitante.	Active Directory no generó un `directoryGuid` para su directorio. Solución de problemas de Active Directory.
0x8FFFA006	La plantilla especificó el atributo `dnsHostName` del asunto del certificado o su nombre alternativo, pero el atributo no se encontró en el objeto AD del solicitante.	Agregue el atributo de `dnsHostName` a su objeto de AD.
0x8FFFA007	La plantilla especificó el atributo de correo electrónico que se debe incluir en el asunto del certificado o su nombre alternativo, pero el atributo no se encontró en el objeto AD para el solicitante.	Agregue el atributo de correo electrónico a su objeto de AD

Código de error	Causa raíz	Corrección
0x8FFFA008	El mensaje SOAP debe tener un encabezado de acción igual a `http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy/IPolicy/GetPolicies` o `http://schemas.microsoft.com/windows/pki/2009/01/enrollment/RST/wstep` .	Actualice el encabezado de la acción para usar uno de los valores especificados.
0x8FFFA009	BinarySecurityToken Debe estar codificado en`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary` .	Actualice el tipo de token de seguridad binario.
0x8FFFA00A	El no BinarySecurityToken es válido.	Compruebe que la CSR se generó correctamente.
0x8FFFA00B	BinarySecurityToken Debe tener un tipo de valor igual `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#PKCS7` o igual a`http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10` .	Actualice el tipo de valor del token de seguridad binario a un valor válido.

Código de error	Causa raíz	Corrección
0x8FFFA00C	El CMS no válido BinarySecurityToken contenido.	El Base64 es válido, pero la sintaxis del mensaje criptográfico (CMS) no es válida. Revise la sintaxis del CMS.
0x8FFFA00D	BinarySecurityToken Contenía una CSR no válida.	Compruebe que la CSR se haya generado correctamente.
0x8FFFA00E	La CA privada no pudo emitir un certificado con la plantilla específica.	Revise la excepción de validación de AWS Private CA. Puedes ver la excepción de validación en Amazon EventBridge o AWS CloudTrail.
0x8FFFA00F	El mensaje de SOAP debe tener un tipo de solicitud de `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .	Establezca el tipo de solicitud en `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .
0x8FFFA010	El mensaje SOAP debe tener un encabezado de destino en el campo `CertificateEnrollmentPolicyServerEndpoint` del conector o en el campo URI de la respuesta de XCEP.	Defina el encabezado del token de seguridad de la solicitud en el campo `CertificateEnrollmentPolicyServerEndpoint` o en el campo URI de la respuesta de XCEP.
0x8FFFA011	El mensaje SOAP debe tener solo un encabezado de acción.	Revise el encabezado del mensaje SOAP del token de seguridad de la solicitud y configúrelo correctamente.
0x8FFFA012	El mensaje SOAP debe tener solo un encabezado `messageId` .	Revise el encabezado del mensaje SOAP del token de seguridad de la solicitud y configúrelo correctamente.

Código de error	Causa raíz	Corrección
0x8FFFA013	El mensaje SOAP debe tener solo un encabezado .	Revise el encabezado del mensaje SOAP del token de seguridad de la solicitud y configúrelo correctamente.
0x8FFFA014	El solicitante no tiene acceso a la plantilla solicitada.	Cree una entrada de control de acceso para permitir que el grupo del solicitante se inscriba utilizando la plantilla solicitada.
0x8FFFA015	La extensión `CertificateTemplateInformation` o la `CertificateTemplateName` extensión deben estar presentes en BinarySecurityToken.	Añada la extensión de seguridad a su CSR.
0x8FFFA016	No se encontró la plantilla solicitada para el conector indicado.	Las plantillas son recursos secundarios para cada conector. Cree la plantilla para el conector utilizando `createTemplate` .
0x8FFFA017	La solicitud fue denegada debido a una limitación de la solicitud.	Reduzca la taza de solicitudes.
0x8FFFA018	El mensaje SOAP debe contener un encabezado de `to`.	Revise el encabezado del mensaje SOAP.
0x8FFFA019	No se pudo procesar el mensaje SOAP debido a un encabezado no reconocido.	Revise el encabezado del mensaje SOAP.

Código de error	Causa raíz	Corrección
0x8FFFA01A	La plantilla especificó el atributo UPN que se debe incluir en el asunto del certificado o su nombre alternativo, pero el atributo no se encontró en el objeto AD para el solicitante.	Agregue un UPN al objeto de Active Directory.

Solucione los errores de creación del conector AD en el conector

La creación de un conector para AD puede fallar por varios motivos. Si se produce un error en la creación del conector, recibirás el motivo del error en la respuesta de la API. Si utilizas la consola, el motivo del error se muestra en la página de detalles del conector, en el campo Detalles de estado adicionales del contenedor de detalles del conector. En la siguiente tabla se describen los motivos del error y los pasos recomendados para su resolución.

Estado del fallo	Descripción	Corrección
`CA_CERTIFICATE_REGISTRATION_FAILED`	Connector for AD no puede importar los certificados de CA a su directorio.	Revise la página de requisitos previos y compruebe que su cuenta de servicio tiene los permisos correctos. Tras delegar los permisos correctos en su cuenta de servicio, elimine el conector defectuoso y cree uno nuevo. Para obtener información sobre la delegación de permisos, consulte Delegar privilegios en su cuenta de servicio en la Guía de AWS Directory Service administración.
`DIRECTORY_ACCESS_DENIED`	Connector para AD no puede acceder a su directorio.	

Estado del fallo	Descripción	Corrección
		Debe conceder a Connector for AD el acceso a su directorio. Revise la Paso 4: Crear una política de IAM sección para asegurarse de que la política de IAM asociada a su AWS cuenta le permite acceder a los directorios y describirlos. Tras conceder los permisos correctos a su AWS función, elimine el conector defectuoso y cree uno nuevo. Si utiliza Connector for AD con un conector AWS Directory Service AD, asegúrese de que la contraseña de la cuenta de servicio de AD Connector no esté caducada y sea válida. Para obtener información sobre las cuentas de servicio de AD Connector, consulte Introducción a AD Connector en la Guía de administración de AD Connector.
`INTERNAL_FAILURE`	El conector para AD sufrió un fallo interno.	Inténtelo de nuevo más tarde. Elimine el conector defectuoso y cree uno nuevo.

Estado del fallo	Descripción	Corrección
`INSUFFICIENT_FREE_ADDRESSES`	La subred de VPC debe tener al menos una dirección IP privada disponible.	Asegúrese de que haya una dirección IP privada disponible en la subred. Elimine el conector defectuoso y cree uno nuevo.
`INVALID_SUBNET_IP_PROTOCOL`	Connector for AD no puede crear el punto final en la VPC porque las subredes asociadas a su directorio no admiten el tipo de dirección IP especificado.	Asegúrese de que la VPC y las subredes que alojan el directorio sean compatibles con el tipo de dirección IP que haya elegido. Para obtener más información, consulte Tipos de direcciones IP. Elimine el conector defectuoso y cree uno nuevo con el tipo de dirección IP compatible.

Estado del fallo	Descripción	Corrección
`PRIVATECA_ACCESS_DENIED`	Connector for AD no puede acceder a su CA privada.	Revisa la página de requisitos previos y comprueba que tienes los permisos para crear un conector. Para obtener más información, consulte Paso 4: Crear una política de IAM. Si va a crear un conector a través AWS CLI de una API, consulte la página de requisitos previos y compruebe que ha compartido la CA privada con Connector para que la utilice AD. AWS Resource Access Manager Tras comprobar y corregir los permisos de IAM y el uso compartido de AWS RAM recursos, elimina el conector defectuoso y crea uno nuevo.
`PRIVATECA_RESOURCE_NOT_FOUND`	Connector for AD no encuentra la CA privada especificada.	Asegúrese de especificar el nombre de recurso (ARN) de CA privado correcto y, a continuación, elimine el conector defectuoso y cree uno nuevo con el ARN de CA privado que desee.

Estado del fallo	Descripción	Corrección
`SECURITY_GROUP_NOT_IN_VPC`	El grupo de seguridad no está en la VPC que aloja el directorio.	Utilice un grupo de seguridad que esté en la VPC que aloja el directorio. Para obtener más información, consulte Paso 7: Configurar grupos de seguridad. Elimine el conector defectuoso y cree uno nuevo con un grupo de seguridad que esté en la VPC.
`VPC_ACCESS_DENIED`	Connector for AD no puede acceder a la Amazon VPC que aloja su directorio.	Verifique sus permisos de IAM. Elimine el conector defectuoso y cree uno nuevo. Para ver un ejemplo de política de IAM que incluye permisos de acceso, consulte Paso 4: Crear una política de IAM
`VPC_ENDPOINT_LIMIT_EXCEEDED`	Connector for AD no puede crear un punto de conexión en su Amazon VPC. Has alcanzado el límite de puntos de conexión de VPC que puedes crear para tu cuenta.	Elimine los puntos de enlace de Amazon VPC o solicite un aumento del límite. Una vez que haya realizado uno de los dos pasos, elimine el conector defectuoso y cree uno nuevo. Para obtener información sobre las cuotas, consulte las cuotas de Amazon Virtual Private Cloud Service.

Estado del fallo	Descripción	Corrección
VPC_RESOURCE_NOT_FOUND	Connector for AD no encuentra la VPC especificada.	Asegúrese de haber especificado la VPC correcta y de que la VPC existe. A continuación, elimine el conector defectuoso y cree uno nuevo con el ID de VPC correcto.

Solucione el error de creación del SPN de AD en el conector

La creación del nombre principal del servicio (SPN) puede fallar por varios motivos. Si se produce un error al crear un SPN, recibirás el motivo del error en la respuesta de la API. Si utilizas la consola, el motivo del error se muestra en la página de detalles del conector, en el campo Detalles de estado adicionales del contenedor del nombre principal del servicio (SPN). En la siguiente tabla se describen los motivos del error y los pasos recomendados para su resolución.

Estado del fallo	Descripción	Corrección
DIRECTORY_ACCESS_DENIED	Connector for AD no puede acceder a su directorio.	Conceda a Connector for AD acceso a su directorio. Para ver un ejemplo de política de IAM que incluye permisos que otorgan acceso al directorio, consulte Paso 4: Crear una política de IAM.
DIRECTORY_NOT_REACHABLE	Connector para AD no puede acceder a su directorio.	Compruebe la red entre AWS y su directorio e intente crear un SPN de nuevo.
DIRECTORY_RESOURCE_NOT_FOUND	Connector for AD no encuentra el directorio especificado.	Asegúrese de especificar el identificador de directorio correcto y, a continuación,

Estado del fallo	Descripción	Corrección
		elimine el conector defectuoso y cree uno nuevo con el identificador de directorio deseado.
`INTERNAL_FAILURE`	El conector para AD ha sufrido un fallo interno.	Inténtelo de nuevo más tarde.
`SPN_EXISTS_ON_DIFFERENT_AD_OBJECT`	El nombre principal del servicio (SPN) existe en un objeto de Active Directory diferente.	Elimine el SPN del objeto de Active Directory e intente crearlo de nuevo.
`SPN_LIMIT_EXCEEDED`	Connector for AD no puede crear el SPN porque se ha alcanzado el límite por directorio. SPNs El número máximo de SPNs por directorio es 10.	Elimina uno o varios SPNs de tu cuenta e intenta crear el SPN de nuevo.

Soluciona problemas de actualización de plantillas de Connector for AD

Si ha realizado cambios en la plantilla o en la entrada de control de acceso de grupo, pero no ve los cambios, es posible que se deba al almacenamiento en caché de las políticas. AWS Private CA aplica la plantilla a la política cuando el cliente actualiza la caché de la política, es decir, cada ocho horas. Cuando el cliente actualiza la caché, consulta el conector para ver las plantillas disponibles. En el caso de la actualización de la inscripción automática, el cliente emite certificados que cumplen una o ambas de las siguientes condiciones:

- El certificado se encuentra dentro del período de renovación.
- El certificado no está presente en el dispositivo cliente.

Para la actualización manual, el cliente consultará el conector y usted deberá configurar la plantilla para que se emita.

Si está depurando, puede borrar manualmente la caché de políticas para ver inmediatamente los cambios en la plantilla. Para ello, ejecuta el siguiente comando de Powershell en tu cliente.

```
certutil -f -user -policyserver * -policycache delete
```

Autoridad de certificación privada de AWS Conector para SCEP

El conector para el protocolo simple de inscripción de certificados (SCEP) se conecta AWS Private Certificate Authority a sus dispositivos móviles y equipos de red compatibles con el SCEP. Con Connector for SCEP, puede utilizarlo AWS Private CA para emitir certificados e inscribir sus dispositivos SCEP. El conector para SCEP está disponible para su uso con los populares sistemas de administración de dispositivos móviles (MDM) y está diseñado para funcionar con clientes o terminales compatibles con el SCEP.

Temas

- Características
- ¿Cómo empezar a utilizar Connector for SCEP
- Servicios relacionados
- Conector de acceso para SCEP
- Precios
- Conector para conceptos de SCEP
- Conozca las consideraciones y limitaciones de Connector for SCEP
- Configure el conector para el SCEP
- Comience con Connector for SCEP
- Configura tu sistema MDM para Connector for SCEP
- Seguridad en el conector para SCEP
- Conector de monitor para SCEP
- Solucione problemas de AWS Private Certificate Authority SCEP con el conector

Características

Soporte para el protocolo SCEP: el SCEP es un protocolo ampliamente adoptado para obtener certificados de identidad digital de una autoridad de certificación (CA) y distribuirlos a dispositivos móviles y equipos de red. Puede utilizar Connector for SCEP como ayuda para inscribir sus terminales mediante el SCEP.

Inscripción de dispositivos móviles: puedes usar Connector for SCEP con los sistemas MDM más populares, como Microsoft Intune y Jamf Pro.

Emita certificados a gran escala: después de configurar sus dispositivos compatibles con el CEP para que soliciten certificados a través del terminal SCEP del conector, sus clientes podrán solicitarlos automáticamente. AWS Private CA

¿Cómo empezar a utilizar Connector for SCEP

Para empezar, inicie el asistente guiado desde la consola de administración de Connector for SCEP, que le ayudará a crear un conector y a designar la CA privada que se utilizará con el conector. Tras completar estos pasos, Connector for SCEP proporciona un punto final y otros parámetros de configuración que puede introducir en sus sistemas MDM o equipos de red. Tras configurar sus sistemas MDM o equipos de red, sus clientes solicitarán automáticamente los certificados. AWS Private CA Para obtener más información sobre cómo empezar a utilizar Connector for SCEP, consulte. Comience con Connector for SCEP

Servicios relacionados

El conector para SCEP está relacionado con los siguientes AWS servicios.

- AWS Private Certificate Authority- le AWS Private CA proporciona un servicio de CA privada de alta disponibilidad sin la inversión inicial ni los costos de mantenimiento continuos que implica operar su propia CA privada.
- AWS Private CA Conector para Active Directory: el conector para AD vincula su Active Directory (AD) a. AWS Private CA El conector gestiona el intercambio de certificados entre AWS Private CA los usuarios y las máquinas gestionados por su AD.

Conector de acceso para SCEP

Puede crear, acceder y administrar sus conectores Connector for SCEP mediante cualquiera de las siguientes interfaces:

- Consola de administración de AWS- Proporciona una interfaz web que puede utilizar para acceder a Connector for SCEP. Consulte Connector para la consola de administración del SCEP.

- AWS Command Line Interface- Proporciona comandos para un amplio conjunto de AWS servicios, incluido el conector para SCEP. AWS CLI Es compatible con Windows, macOS y Linux. Para obtener más información, consulte AWS Command Line Interface.
- AWS SDKs- Indique un idioma específico APIs y cuide muchos de los detalles de la conexión, como el cálculo de las firmas, la gestión de los reintentos de solicitud y la gestión de los errores. Para obtener más información, consulte AWS Command Line Interface.
- Conector para la API SCEP: proporciona acciones de API de bajo nivel a las que se puede recurrir mediante solicitudes HTTPS. El uso de la API Connector for SCEP es la forma más directa de acceder al servicio. Sin embargo, la API de Connector for SCEP requiere que la aplicación gestione detalles de bajo nivel, como la generación del hash para firmar la solicitud y la gestión de errores. Para obtener más información, consulte la referencia de la API Connector for SCEP.

Precios

El conector para SCEP se ofrece como una función sin Autoridad de certificación privada de AWS coste adicional. Solo paga por AWS Private Certificate Authority las operaciones y los certificados utilizados para crear y actualizar los conectores.

Para obtener la información Autoridad de certificación privada de AWS de precios más reciente, consulte AWS Private Certificate Authority Precios. También puede utilizar la calculadora de precios de AWS para estimar los costos.

Conector para conceptos de SCEP

El conector para SCEP es una función adicional de. AWS Private Certificate Authority

Los conceptos clave de Connector for SCEP son los siguientes:

Solicitud de firma de certificado (CSR)

> La información requerida proporcionada a una CA para que se emita un certificado digital. Esta información contiene una clave pública y una identidad.

Contraseña de desafío

> El protocolo SCEP utiliza contraseñas de desafío para autenticar una solicitud antes de emitir un certificado de una CA. El conector para el SCEP gestiona las contraseñas de desafío del SCEP

en función del tipo de conector. Para obtener más información, consulte Configura tu sistema MDM para Connector for SCEP.

Revocación del certificado

La revocación de un certificado es el proceso de revocar un certificado emitido antes de su fecha de caducidad. Puede revocar el certificado de CA privado asociado a un conector llamando a RevokeCertificatela API, el AWS SDK o. AWS Command Line Interface AWS CloudFormation

Conector para SCEP

Un conector para el SCEP se conecta AWS Private CA a sus dispositivos compatibles con el SCEP.

Administración de dispositivos móviles

La administración de dispositivos móviles (MDM) permite a los administradores de TI controlar, proteger y hacer cumplir las políticas en teléfonos inteligentes, tabletas y otros terminales o dispositivos. Muchos sistemas MDM ofrecen integraciones integradas para la inscripción de certificados basada en el CEP.

SCEP

El SCEP es un protocolo estandarizado (RFC 8894) para distribuir certificados automáticamente. El protocolo proporciona un punto final para que los dispositivos soliciten certificados a una CA. El SCEP utiliza contraseñas de desafío para autorizar la emisión de certificados a los dispositivos. El SCEP se suele aplicar a los sistemas de administración de dispositivos móviles (MDM) y a los equipos de red. Las soluciones MDM permiten a los administradores de TI controlar, proteger y hacer cumplir las políticas en teléfonos inteligentes, tabletas y otras entidades, como las estaciones de trabajo de Apple. La mayoría de las soluciones MDM son compatibles con SCEP, como Microsoft Intune, Apple MDM y Jamf Pro. La mayoría de los equipos de red, como los enrutadores, los balanceadores de carga, los concentradores Wi-Fi, los dispositivos VPN y los firewalls, utilizan el SCEP para la inscripción automática de certificados.

Perfil SCEP

Un perfil SCEP contiene parámetros de configuración que se utilizan para definir el perfil del certificado. Esto incluye el período de validez del certificado, el tamaño de la clave, el nombre de la configuración del SCEP, la contraseña de verificación, el número de intentos fallidos y el intervalo de reintentos, así como otra información relevante para la emisión de los certificados. Los sistemas MDM y las plataformas de administración de certificados suelen enviar el perfil SCEP al cliente, que solicitará un certificado de autenticación.

Conozca las consideraciones y limitaciones de Connector for SCEP

Tenga en cuenta las siguientes consideraciones y limitaciones al utilizar Connector for SCEP.

Consideraciones

Modos de funcionamiento de CA

Solo puede utilizar Connector for SCEP con un modo de funcionamiento privado CAs que utilice un modo de funcionamiento de uso general. Por defecto, Connector for SCEP emite certificados con un período de validez de un año. Una entidad emisora de certificados privada que utilice un modo de certificación de corta duración no admite la emisión de certificados con un período de validez superior a siete días. Para obtener información sobre los modos de funcionamiento, consulteAWS Private CA Comprenda los modos de CA.

Impugnar contraseñas

- Distribuya sus contraseñas confidenciales con mucho cuidado y compártalas solo con personas y clientes de gran confianza. Se puede utilizar una única contraseña de seguridad para emitir cualquier certificado, con cualquier asunto y SANs que suponga un riesgo para la seguridad.
- Si utiliza un conector de uso general, le recomendamos que cambie manualmente las contraseñas de desafío con frecuencia.

Cumple con el RFC 8894

El conector para SCEP se desvía del protocolo RFC 8894 al proporcionar puntos de enlace HTTPS en lugar de puntos de enlace HTTP.

CSRs

- Si una solicitud de firma de certificado (CSR) que se envía a Connector for SCEP no incluye la extensión de uso extendido de claves (EKU), estableceremos el valor de EKU en. `clientAuthentication` Para obtener más información, consulte 4.2.1.12. Uso extendido de claves en el RFC 5280.
- Admitimos `ValidityPeriod` y `ValidityPeriodUnits` personalizamos los atributos en. CSRs Si su CSR no incluye un certificado`ValidityPeriod`, emitimos un certificado que tiene un período de validez de un año. Tenga en cuenta que es posible que no pueda establecer estos atributos en su sistema MDM. Pero si puede configurarlos, los admitimos. Para obtener información sobre estos atributos, consulte SZEnrollment_NAME_VALUE_PAIR.

Uso compartido de terminales

Distribuya los puntos finales de un conector únicamente a partes de confianza. Trate los puntos finales como secretos, ya que cualquier persona que pueda encontrar su nombre de dominio y su ruta únicos y totalmente cualificados podrá recuperar su certificado de CA.

Limitaciones

Las siguientes limitaciones se aplican a Connector for SCEP.

Contraseñas de desafío dinámico

Solo puede crear contraseñas de desafío estáticas con conectores de uso general. Para utilizar contraseñas dinámicas con un conector de uso general, debe crear su propio mecanismo de rotación que utilice las contraseñas estáticas del conector. Los tipos de conectores Connector for SCEP para Microsoft Intune admiten contraseñas dinámicas, que se administran mediante Microsoft Intune.

HTTP

El conector para SCEP solo admite HTTPS y crea redireccionamientos para las llamadas HTTP. Si su sistema depende de HTTP, asegúrese de que puede admitir los redireccionamientos HTTP que proporciona Connector for SCEP.

Compartido: privado CAs

Solo puede utilizar Connector for SCEP con CAs la red privada de la que sea propietario.

Configure el conector para el SCEP

Los procedimientos de esta sección le ayudarán a empezar a utilizar Connector for SCEP. Se supone que ya ha creado una AWS cuenta. Tras completar los pasos de esta página, puede continuar con la creación de un conector para el SCEP.

Temas

- Paso 1: Crear una política AWS Identity and Access Management
- Paso 2: Crear una CA privada
- Paso 3: Cree un recurso compartido mediante AWS Resource Access Manager

Paso 1: Crear una política AWS Identity and Access Management

Para crear un conector para el SCEP, debe crear una política de IAM que permita a Connector for SCEP crear y gestionar los recursos que necesita el conector y emitir certificados en su nombre. Para obtener más información sobre IAM, consulte ¿Qué es IAM? en la Guía del usuario de IAM.

El siguiente ejemplo es una política gestionada por el cliente que puede utilizar para Connector for SCEP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "pca-connector-scep:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListCertificateAuthorities",
                "acm-pca:ListTags",
                "acm-pca:PutPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "acm-pca:IssueCertificate",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APICSRPassthrough/V*"
                },
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "pca-connector-scep.amazonaws.com"
```

```
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:CreateResourceShare",
                "ram:GetResourcePolicies",
                "ram:GetResourceShareAssociations",
                "ram:GetResourceShares",
                "ram:ListPrincipals",
                "ram:ListResources",
                "ram:ListResourceSharePermissions",
                "ram:ListResourceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

Paso 2: Crear una CA privada

Para utilizar Connector for SCEP, debe asociar una CA privada desde AWS Private Certificate Authority al conector. Le recomendamos que utilice una CA privada que sea únicamente para el conector, debido a las vulnerabilidades de seguridad inherentes que están presentes en el protocolo SCEP.

La CA privada debe cumplir los siguientes requisitos:

- Debe estar en un estado activo y utilizar el modo de funcionamiento de uso general.
- Debe ser propietario de la CA privada. No puedes usar una CA privada que se haya compartido contigo a través del uso compartido entre cuentas.

Tenga en cuenta las siguientes consideraciones al configurar su CA privada para utilizarla con Connector for SCEP:

- Restricciones de nombres DNS: considere utilizar restricciones de nombres DNS como forma de controlar qué dominios están permitidos o prohibidos en los certificados emitidos para sus dispositivos SCEP. Para obtener más información, consulte Cómo aplicar restricciones de nombres DNS en. AWS Private Certificate Authority

- Revocación: habilite el OCSP o CRLs en su CA privada para permitir la revocación. Para obtener más información, consulte Planifique el método de revocación AWS Private CA de su certificado.
- PII: le recomendamos que no añada información de identificación personal (PII) ni ningún otro tipo de información confidencial o delicada en sus certificados de CA. En caso de que se produzca una vulnerabilidad de seguridad, esto ayuda a limitar la exposición de la información confidencial.
- Guarde los certificados raíz en almacenes de confianza: almacene los certificados de CA raíz en los almacenes de confianza de sus dispositivos para poder comprobar los certificados y sus valores devueltos GetCertificateAuthorityCertificate. Para obtener información sobre los almacenes de confianza relacionados con ellos AWS Private CA, consulteCA raíz.

Para obtener información sobre cómo crear una CA privada, consulteCree una CA privada en AWS Private CA.

Paso 3: Cree un recurso compartido mediante AWS Resource Access Manager

Si utiliza Connector for SCEP mediante programación mediante el AWS SDK o la AWS Command Line Interface API de Connector for SCEP, debe compartir su CA privada con Connector for SCEP mediante el uso compartido del principal de servicio. AWS Resource Access Manager Esto le da a Connector for SCEP acceso compartido a su CA privada. Al crear un conector en la AWS consola, creamos automáticamente el recurso compartido por usted. Para obtener información sobre el uso compartido de recursos, consulte Crear un recurso compartido en la Guía del AWS RAM usuario.

Para crear un recurso compartido mediante el AWS CLI, puede utilizar el AWS RAM create-resource-share comando. El siguiente comando crea un recurso compartido. Especifique el ARN de la CA privada que desea compartir como valor. *resource-arns*

```
$  aws ram create-resource-share \
--region us-east-1 \
--name MyPcaConnectorScepResourceShare \
--permission-arns arn:aws:ram::aws:permission/
AWSRAMBlankEndEntityCertificateAPICSRPassthroughIssuanceCertificateAuthority \
--resource-arns arn:aws:acm-pca:Region:account:certificate-authority/CA_ID \
--principals pca-connector-scep.amazonaws.com \
--sources account
```

El director del servicio que llama `CreateConnector` tiene permisos de emisión de certificados en la CA privada. Para evitar que los directores de servicio que utilizan Connector for SCEP tengan acceso general a sus Autoridad de certificación privada de AWS recursos, restrinja sus permisos mediante. `CalledVia`

Comience con Connector for SCEP

Con AWS Private Certificate Authority Connector for SCEP, puede emitir certificados desde su entidad de certificación privada para dispositivos compatibles con el SCEP y sistemas de administración de dispositivos móviles (MDM). Al crear un conector, AWS Private Certificate Authority crea una URL del SCEP para que pueda solicitar certificados y, además, le proporciona información que puede utilizar para integrarla en sus sistemas MDM.

Para emitir certificados, debe crear una CA AWS Private Certificate Authority privada, crear un conector y, a continuación, configurar los sistemas y dispositivos MDM compatibles con el SCEP para solicitar certificados desde el conector.

Temas

- Antes de empezar
- Paso 1: Crear un conector
- Paso 2: Copie los detalles del conector en su sistema MDM

Antes de empezar

El siguiente tutorial le guiará por el proceso de creación de un conector para el SCEP.

Para seguir este tutorial, necesitará una CA privada y un dispositivo compatible con el SCEP. También debes cumplir primero con los requisitos previos que se indican en la sección. Configure el conector para el SCEP

El siguiente procedimiento le guía sobre cómo crear un conector mediante la AWS consola.

Tareas

- Paso 1: Crear un conector
- Paso 2: Copie los detalles del conector en su sistema MDM

Paso 1: Crear un conector

Creará un conector para uso general o un conector para SCEP para Microsoft Intune. Los conectores de uso general están diseñados para su uso con terminales compatibles con el SCEP, y usted administra las contraseñas de desafío del SCEP. El conector para SCEP para Microsoft Intune se usa con Microsoft Intune y las contraseñas de desafío se administran con Microsoft Intune.

General-purpose

Para crear un conector para uso general

Inicie sesión en su AWS cuenta y abra la consola Connector for SCEP en. **https://console.aws.amazon.com/pca-connector-scep/home**

1. Elija Crear conector.
2. En la página Crear conector, si lo desea, asigne un nombre descriptivo al conector en el campo de la etiqueta de nombre. El nombre aparecerá en la lista de conectores. Si lo desea, puede añadir más etiquetas al conector seleccionando Añadir más etiquetas. Una etiqueta es una etiqueta que se asigna a un AWS recurso. Cada etiqueta consta de una clave y un valor opcional. Puede usar etiquetas para buscar y filtrar sus recursos o realizar un seguimiento de sus AWS costos.
3. En Tipo de conector, selecciona Uso general.
4. En CA privada, elija la CA privada que se va a usar con este conector. O bien, cree una nueva seleccionando Crear CA privada. Debido a las vulnerabilidades inherentes al protocolo SCEP, recomendamos utilizar una CA privada dedicada a este conector. Si ha creado una nueva CA, cuando termine de crearla AWS Private CA, vuelva a la consola de Connector for SCEP y actualice la lista de entidades privadas. CAs Su nueva CA privada debería estar disponible para su selección.
5. En Contraseña de desafío, selecciona Generar contraseña de desafío automáticamente. Generaremos una contraseña de desafío estática para usted cuando creemos este conector.
6. En Conectividad, selecciona Público para crear un conector accesible a través de la Internet pública. O bien, seleccione Privado y especifique un punto de enlace de VPC para restringir el acceso a este conector solo a través de ese punto de enlace de VPC específico.
7. Seleccione Crear conector.

Microsoft Intune

Para crear un conector para SCEP para Microsoft Intune

Inicie sesión en su AWS cuenta y abra la consola de Connector for SCEP en. **https://console.aws.amazon.com/pca-connector-scep/home**

1. Elija Crear conector.
2. En la página Crear conector, si lo desea, asigne un nombre descriptivo al conector en el campo de la etiqueta de nombre. El nombre aparecerá en la lista de conectores. Si lo desea, puede añadir más etiquetas al conector seleccionando Añadir más etiquetas. Una etiqueta es una etiqueta que se asigna a un AWS recurso. Cada etiqueta consta de una clave y un valor opcional. Puede usar etiquetas para buscar y filtrar sus recursos o realizar un seguimiento de sus AWS costos.
3. En Tipo de conector, selecciona Microsoft Intune.
   a. Para el ID de la aplicación (cliente), introduzca el ID de la aplicación (cliente) que aparece en el registro de la aplicación Microsoft Entra ID. Para obtener información sobre el uso de Microsoft Intune con Connector for SCEP, consulte. Configura tu sistema MDM para Connector for SCEP
   b. Para el ID de directorio (inquilino) o el dominio principal, introduce el ID del directorio (inquilino) o el dominio principal desde el registro de la aplicación Microsoft Entra ID.
4. En CA privada, elige la CA privada que quieres usar con este conector. O bien, cree una nueva seleccionando Crear CA privada. Debido a las vulnerabilidades inherentes al protocolo SCEP, recomendamos utilizar una CA privada dedicada a este conector. Si ha creado una nueva CA, cuando termine de crearla AWS Private CA, vuelva a la consola de Connector for SCEP y actualice la lista de entidades privadas. CAs Su nueva CA privada debería estar disponible para su selección.
5. En Conectividad, elija Público para crear un conector accesible a través de la Internet pública. O bien, seleccione Privado y especifique un punto de enlace de VPC para restringir el acceso a este conector solo a través de ese punto de enlace de VPC específico.
6. Seleccione Crear conector.

Paso 2: Copie los detalles del conector en su sistema MDM

Tras crear el conector, tendrás que copiar los siguientes detalles del conector a tu sistema MDM. Para ver los detalles de un conector mediante la consola, seleccione el conector de la lista de la página Conectores para la consola SCEP.

- URL del SCEP: es el punto final del conector desde el que los clientes del SCEP solicitarán los certificados. Tenga cuidado de proporcionar este punto final únicamente a entidades de confianza.
- Contraseña de desafío (de uso general): en Contraseñas de desafío, seleccione la contraseña que generó automáticamente en el procedimiento anterior y, a continuación, seleccione Ver contraseña para ver la contraseña. Para crear una contraseña adicional, seleccione Crear contraseña. Tenga cuidado de distribuir las contraseñas cuidadosamente y solo a personas y clientes de gran confianza. Se puede utilizar una única contraseña de verificación para emitir cualquier certificado, con cualquier asunto y SANs, por lo tanto, debe manejarse con cuidado.
- Valores de Open ID (Microsoft Intune): si vas a realizar la integración con Microsoft Intune, debes copiar el emisor de Open ID, el asunto de Open ID y el público de Open ID en la credencial OpenID Connect (OIDC) del registro de la aplicación Microsoft Entra. Para obtener más información, consulte Configura tu sistema MDM para Connector for SCEP.

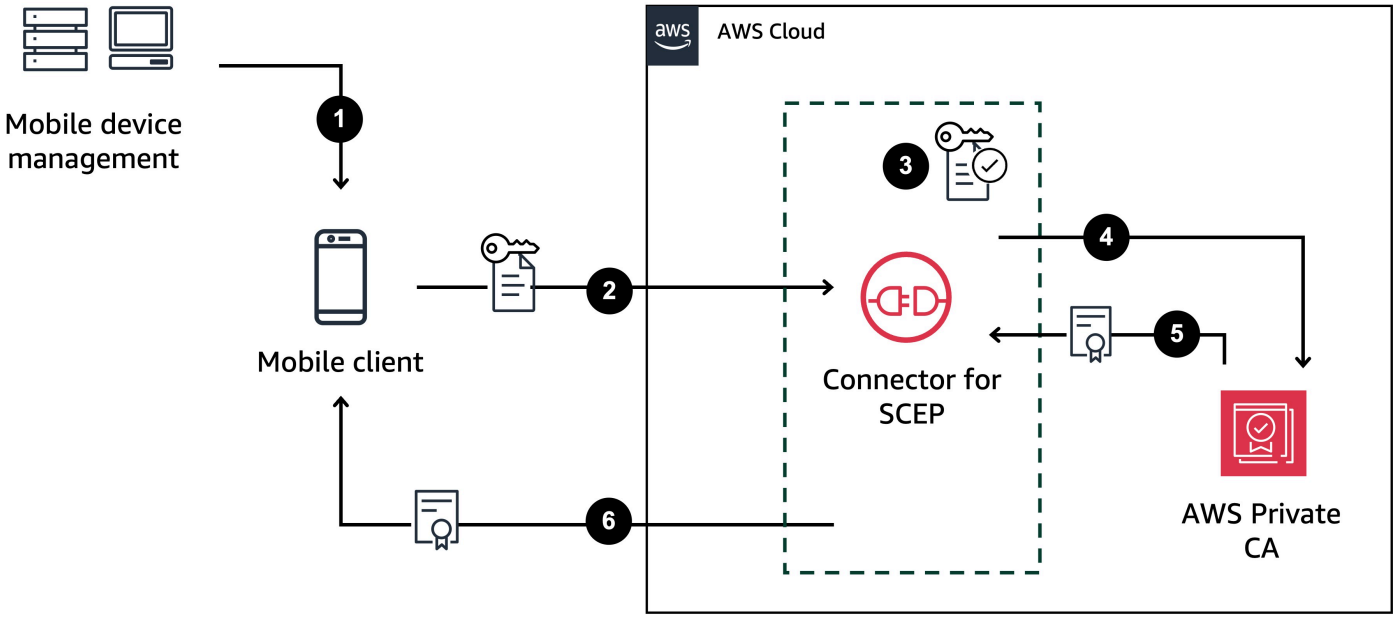
Configura tu sistema MDM para Connector for SCEP

El Protocolo simple de inscripción de certificados (SCEP) es un protocolo estándar que se utiliza para la inscripción y renovación de certificados. Connector for SCEP es un servidor SCEP basado en el RFC 8894 que emite certificados automáticamente a sus clientes del AWS Private Certificate Authority SCEP. Al crear un conector, Connector for SCEP proporciona un punto de conexión HTTPS desde el que los clientes del SCEP pueden solicitar certificados. Los clientes se autentican mediante una contraseña de seguridad que se incluye como parte de su solicitud de firma de certificados (CSR) al servicio. Puedes usar Connector for SCEP con los sistemas de administración de dispositivos móviles (MDM) más populares, como Microsoft Intune, Omnissa Workspace ONE y Jamf Pro, para inscribir dispositivos móviles. Está diseñado para funcionar con cualquier cliente o terminal compatible con SCEP.

El conector para SCEP ofrece dos tipos de conectores: de uso general y conector para SCEP para Microsoft Intune. En las siguientes secciones se describe cómo funcionan y cómo configurar el sistema MDM para utilizarlos.

Conector de uso general

Un conector de uso general está diseñado para funcionar con terminales de dispositivos móviles compatibles con SCEP, excepto Microsoft Intune, que tiene un conector dedicado. Con los conectores de uso general, como Jamf Pro u Omnissa Workspace ONE, puedes gestionar las contraseñas de desafío del SCEP. En el siguiente diagrama se utiliza un sistema de gestión de dispositivos móviles (MDM) como ejemplo, pero la misma funcionalidad se aplica a otros sistemas o dispositivos compatibles con el SCEP.

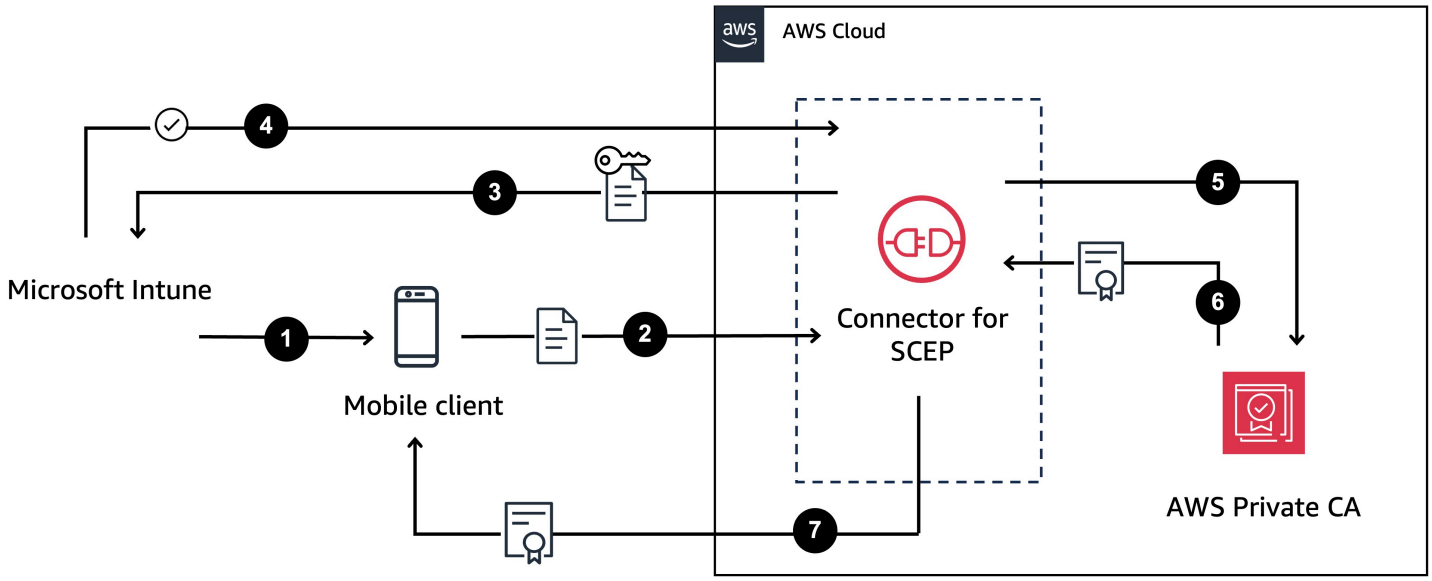


1. El sistema MDM (u otro dispositivo o sistema) envía un perfil SCEP al cliente móvil. Un perfil SCEP contiene parámetros de configuración que definen el perfil del certificado, como el período de validez del certificado, la contraseña de verificación y otra información relevante para la emisión de los certificados.
2. El cliente móvil solicita un certificado y también envía una solicitud de firma de certificado (CSR) que incluye una contraseña de verificación.
3. El conector para SCEP valida la contraseña de desafío. Si es válida, el servicio solicita un certificado AWS Private CA en nombre del cliente móvil.
4. AWS Private CA emite el certificado y lo envía a Connector for SCEP.
5. Connector para SCEP envía el certificado emitido al cliente móvil.

AWS Private CA Conector para SCEP para Microsoft Intune

AWS Private CA El conector para SCEP para Microsoft Intune está diseñado para usarse con Microsoft Intune. Con el tipo de conector Connector for SCEP para Microsoft Intune, utilizará Microsoft Intune para administrar sus contraseñas de desafío de SCEP. Para obtener más información sobre el uso de Connector for SCEP con Microsoft Intune, consulte. Configurar Microsoft Intune para el conector para SCEP

Para usar Connector for SCEP con Microsoft Intune, debe habilitar funcionalidades específicas mediante la API de Microsoft Intune y poseer una licencia de Microsoft Intune válida. También debe revisar las políticas de protección de aplicaciones de Microsoft Intune®.



1. Microsoft Intune envía un perfil SCEP al cliente móvil. El perfil contiene una contraseña de desafío cifrada que el cliente móvil coloca en la CSR.
2. El cliente móvil solicita un certificado y envía la CSR a Connector for SCEP.
3. El conector para SCEP envía la CSR a Microsoft Intune para su autorización.
4. Microsoft Intune descifra la contraseña de desafío en la CSR. Si es válido, Microsoft Intune envía la aprobación a Connector para que el SCEP emita el certificado al cliente móvil.
5. Connector for SCEP solicita un certificado AWS Private CA en nombre del cliente móvil.
6. AWS Private CA emite el certificado y lo envía a Connector for SCEP.
7. Connector para SCEP envía el certificado emitido al cliente móvil.

Temas

- Configura Jamf Pro para Connector para SCEP
- Configurar Microsoft Intune para el conector para SCEP
- Configure Omnissa Workspace ONE for Connector para SCEP

Configura Jamf Pro para Connector para SCEP

Puedes utilizarlo AWS Private CA como autoridad de certificación (CA) externa con el sistema de gestión de dispositivos móviles (MDM) de Jamf Pro. Esta guía proporciona instrucciones sobre cómo configurar Jamf Pro después de crear un conector de uso general.

Configura Jamf Pro para Connector para SCEP

Esta guía proporciona instrucciones sobre cómo configurar Jamf Pro para su uso con Connector for SCEP. Tras configurar correctamente Jamf Pro y Connector for SCEP, podrás emitir AWS Private CA certificados para tus dispositivos gestionados.

Requisitos de Jamf Pro

Tu implementación de Jamf Pro debe cumplir los siguientes requisitos.

- Debes activar la configuración Habilitar la autenticación basada en certificados en Jamf Pro. Puedes encontrar información sobre esta configuración en la página de configuración de seguridad de Jamf Pro, en la documentación de Jamf Pro.

Paso 1: (Opcional, recomendado) Obtén la huella digital de tu CA privada

Una huella digital es un identificador único de su CA privada que se puede utilizar para verificar la identidad de su CA al establecer relaciones de confianza con otros sistemas o aplicaciones. La incorporación de la huella digital de una autoridad de certificación (CA) permite a los dispositivos gestionados autenticar la CA a la que se están conectando y solicitar certificados únicamente a la CA prevista. Te recomendamos usar una huella digital de CA con Jamf Pro.

Para generar una huella digital para tu CA privada

1. Obtenga el certificado de CA privada desde cualquiera de las AWS Private CA consolas o mediante la GetCertificateAuthorityCertificate. Guárdelo como `ca.pem` archivo.
2. Instale las utilidades de línea de comandos de OpenSSL.
3. En OpenSSL, ejecute el siguiente comando para generar la huella digital:

```
openssl x509 -in ca.pem -sha256 -fingerprint
```

Paso 2: Configúralo AWS Private CA como una CA externa en Jamf Pro

Tras crear un conector para el SCEP, debes configurarlo AWS Private CA como una autoridad de certificación (CA) externa en Jamf Pro. Puedes configurarlo AWS Private CA como una CA externa y global. Como alternativa, puedes usar un perfil de configuración de Jamf Pro para emitir diferentes certificados AWS Private CA para diferentes casos de uso, como emitir certificados para un subconjunto de dispositivos de tu organización. Las directrices sobre la implementación de los perfiles de configuración de Jamf Pro van más allá del ámbito de este documento.

Para configurarlo AWS Private CA como una autoridad de certificación (CA) externa en Jamf Pro

1. En la consola de Jamf Pro, ve a la página de configuración de los certificados PKI desde Configuración > Global > Certificados PKI.
2. Selecciona la pestaña Plantilla de certificados de gestión.
3. Seleccione una CA externa.
4. Seleccione Editar.
5. (Opcional) Selecciona Activar Jamf Pro como proxy SCEP para los perfiles de configuración. Puedes utilizar los perfiles de configuración de Jamf Pro para emitir diferentes certificados adaptados a casos de uso específicos. Para obtener información sobre cómo utilizar los perfiles de configuración en Jamf Pro, consulta Cómo habilitar Jamf Pro como proxy SCEP para los perfiles de configuración en la documentación de Jamf Pro.
6. Selecciona Usar una CA externa habilitada para el SCEP para inscribir ordenadores y dispositivos móviles.
7. (Opcional) Selecciona Usar Jamf Pro como proxy SCEP para inscribir ordenadores y dispositivos móviles. Si se producen errores en la instalación del perfil, consulte. Soluciona errores de instalación de perfiles
8. Copia y pega la URL del SCEP de Connector for SCEP de los detalles del conector en el campo URL de Jamf Pro. Para ver los detalles de un conector, selecciónalo de la lista Conectores para SCEP. Como alternativa, puede obtener la URL llamando GetConnectory copiando el `Endpoint` valor de la respuesta.
9. (Opcional) Introduce el nombre de la instancia en el campo Nombre. Por ejemplo, puede asignarle un nombre AWS Private CA.

10. Selecciona Estático para el tipo de desafío.
11. Copia una contraseña de desafío de tu conector y pégala en el campo Desafío. Un conector puede tener varias contraseñas de desafío. Para ver las contraseñas de seguridad de su conector, vaya a la página de detalles del conector en la AWS consola y seleccione el botón Ver contraseña. También puedes obtener las contraseñas de seguridad de un conector llamando GetChallengePasswordy copiando un `Password` valor de la respuesta. Para obtener información sobre el uso de contraseñas de desafío, consulteConozca las consideraciones y limitaciones de Connector for SCEP.
12. Pegue la contraseña de desafío en el campo Verificar desafío.
13. Elige un tamaño de clave. Recomendamos un tamaño de clave de 2048 o superior.
14. (Opcional) Selecciona Usar como firma digital. Seleccione esta opción con fines de autenticación para conceder a los dispositivos un acceso seguro a recursos como Wi-Fi y VPN.
15. (Opcional) Seleccione Usar para el cifrado de claves.
16. (Opcional, recomendado) Introduzca una cadena hexadecimal en el campo Huella digital. Se recomienda añadir una huella digital de la CA para permitir que los dispositivos gestionados verifiquen la CA y que solo soliciten certificados a la CA. Para obtener instrucciones sobre cómo generar una huella digital para su CA privada, consultePaso 1: (Opcional, recomendado) Obtén la huella digital de tu CA privada.
17. Seleccione Guardar.

Paso 3: Configure un certificado de firma del perfil de configuración

Para utilizar Jamf Pro con Connector for SCEP, debes proporcionar los certificados de firma y de CA de la CA privada asociada a tu conector. Para ello, puedes subir a Jamf Pro un almacén de claves de certificados de firma de perfiles que contenga ambos certificados.

Estos son los pasos para crear un almacén de claves de certificados y subirlo a Jamf Pro:

- Genera una solicitud de firma de certificado (CSR) mediante tus procesos internos.
- Haga que la CA privada asociada a su conector firme la CSR.
- Cree un almacén de claves de certificados de firma de perfiles que contenga tanto la firma del perfil como los certificados de CA.
- Sube el almacén de claves de certificados a Jamf Pro.

Si sigues estos pasos, puedes asegurarte de que tus dispositivos pueden validar y autenticar el perfil de configuración firmado por tu entidad emisora de certificados privada, lo que te permitirá utilizar Connector for SCEP con Jamf Pro.

1. En el siguiente ejemplo se utiliza OpenSSL AWS Certificate Manager y, sin embargo, puede generar una solicitud de firma de certificado mediante el método que prefiera.

   AWS Certificate Manager console

   Para crear un certificado de firma de perfil mediante la consola ACM

   1. Utilice ACM para solicitar un certificado de PKI privado. Incluya lo siguiente:

      - Tipo: utilice el mismo tipo de CA privada que actúa como autoridad de certificación del SCEP para su sistema MDM.
      - En la sección de detalles de la autoridad de certificación, selecciona el menú de la autoridad de certificación y elige la CA privada que sirve de CA para Jamf Pro.
      - Nombre de dominio: proporciona un nombre de dominio para incluirlo en el certificado. Puede usar un nombre de dominio completo (FQDN), por ejemplo`www.example.com`, o un nombre de dominio simple o básico `example.com` (que excluye`www.`).

   2. Utilice ACM para exportar el certificado privado que creó en el paso anterior. Seleccione Exportar un archivo para el certificado, la cadena de certificados y la clave cifrada. Ten la contraseña a mano porque la necesitarás en el siguiente paso.

   3. En una terminal, ejecuta el siguiente comando en una carpeta que contenga los archivos exportados para escribir el paquete PKCS #12 en el `output.p12` archivo codificado con la contraseña que creaste en el paso anterior.

      ```
      openssl pkcs12 -export \
        -in "Exported Certificate.txt" \
        -certfile "Certificate Chain.txt" \
        -inkey "Exported Certificate Private Key.txt" \
        -name example \
        -out output.p12 \
        -passin pass:your-passphrase \
        -passout pass:your-passphrase
      ```

AWS Certificate Manager CLI

Para crear un certificado de firma de perfil mediante la CLI de ACM

- El siguiente comando muestra cómo crear un certificado en ACM y, a continuación, exportar los archivos como un paquete PKCS #12.

```
PCA=<Enter your Private CA ARN>

CERTIFICATE=$(aws acm request-certificate \
    --certificate-authority-arn $PCA \
    --domain-name <any valid domain name, such as test.name> \
    | jq -r '.CertificateArn')

while [[ $(aws acm describe-certificate \
  --certificate-arn $CERTIFICATE \
  | jq -r '.Certificate.Status') != "ISSUED" ]] do sleep 1; done

aws acm export-certificate \
  --certificate-arn $CERTIFICATE \
  --passphrase password | jq -r '.Certificate' > Certificate.pem
aws acm export-certificate \
  --certificate-arn $CERTIFICATE \
  --passphrase password | jq -r '.CertificateChain' > CertificateChain.pem
aws acm export-certificate \
  --certificate-arn $CERTIFICATE \
  --passphrase password | jq -r '.PrivateKey' > PrivateKey.pem

openssl pkcs12 -export \
  -in "Certificate.pem" \
  -certfile "CertificateChain.pem" \
  -inkey "PrivateKey.pem" \
  -name example \
  -out output.p12 \
  -passin pass:passphrase \
  -passout pass:passphrase
```

OpenSSL CLI

Para crear un certificado de firma de perfil mediante la CLI de OpenSSL

1. Con OpenSSL, genere una clave privada ejecutando el siguiente comando.

```
openssl genrsa -out local.key 2048
```

2. Genere una solicitud de firma de certificado (CSR):

```
openssl req -new -key local.key -sha512 -out local.csr -
subj "/CN=MySigningCertificate/O=MyOrganization" -addext
 keyUsage=critical,digitalSignature,nonRepudiation
```

3. Con el AWS CLI, emita el certificado de firma con la CSR que generaste en el paso anterior. Ejecute el siguiente comando y anote el ARN del certificado en la respuesta.

```
aws acm-pca issue-certificate --certificate-authority-arn <SAME CA AS
 USED ABOVE, SO IT'S TRUSTED> --csr fileb://local.csr --signing-algorithm
 SHA512WITHRSA --validity Value=365,Type=DAYS
```

4. Obtenga el certificado de firma ejecutando el siguiente comando. Especifique el ARN del certificado del paso anterior.

```
aws acm-pca get-certificate --certificate-authority-arn <SAME CA AS USED
 ABOVE, SO IT'S TRUSTED> --certificate-arn <ARN OF NEW CERTIFICATE> | jq -r
 '.Certificate' >local.crt
```

5. Obtenga el certificado de CA ejecutando el siguiente comando.

```
aws acm-pca get-certificate-authority-certificate --certificate-authority-
arn <SAME CA AS USED ABOVE, SO IT'S TRUSTED> | jq -r '.Certificate' > ca.crt
```

6. Con OpenSSL, genere el almacén de claves de certificados de firma en formato p12. Utilice los archivos CRT que generó en los pasos cuatro y cinco.

```
openssl pkcs12 -export -in local.crt -inkey local.key -certfile ca.crt -name
 "CA Chain" -out local.p12
```

7. Cuando se le pida, introduzca una contraseña de exportación. Esta contraseña es la contraseña del almacén de claves para proporcionársela a Jamf Pro.

2. En Jamf Pro, navega hasta la plantilla de certificado de gestión y ve al panel CA externa.
3. En la parte inferior del panel de CA externa, selecciona Cambiar firma y certificados de CA.
4. Siga las instrucciones que aparecen en pantalla para cargar los certificados de firma y de CA de la CA externa.

Paso 4: (opcional) Instalar el certificado durante la inscripción iniciada por el usuario

Para establecer la confianza entre tus dispositivos cliente y tu CA privada, debes asegurarte de que tus dispositivos confíen en los certificados emitidos por Jamf Pro. Puedes usar la configuración de inscripción iniciada por el usuario de Jamf Pro para instalar automáticamente tu certificado AWS Private CA de CA en los dispositivos cliente cuando estos lo soliciten durante el proceso de registro.

Soluciona errores de instalación de perfiles

Si se producen errores en la instalación de perfiles tras activar la opción Utilizar Jamf Pro como proxy SCEP para la inscripción de ordenadores y dispositivos móviles, consulta los registros de tu dispositivo e intenta lo siguiente.

Mensaje de error en el registro del dispositivo	Mitigación
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:15001>`	Si recibe este mensaje de error al intentar inscribirse, vuelva a intentarlo. Pueden pasar varios intentos antes de que la inscripción se realice correctamente.
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:14006>`	Es posible que la contraseña de verificac ión esté mal configurada. Comprueba que la contraseña de desafío de Jamf Pro coincide con la contraseña de desafío de tu conector.

Configurar Microsoft Intune para el conector para SCEP

Se puede utilizar AWS Private CA como entidad de certificación (CA) externa con el sistema de administración de dispositivos móviles (MDM) de Microsoft Intune. Esta guía proporciona instrucciones sobre cómo configurar Microsoft Intune después de crear un conector para SCEP para Microsoft Intune.

Requisitos previos

Antes de crear un conector para SCEP para Microsoft Intune, debe cumplir los siguientes requisitos previos.

- Cree un ID de Entra.
- Cree un inquilino de Microsoft Intune.
- Cree un registro de aplicación en su ID de Microsoft Entra. Consulte Actualizar los permisos solicitados de una aplicación en Microsoft Entra ID en la documentación de Microsoft Entra para obtener información sobre cómo administrar los permisos a nivel de aplicación para el registro de su aplicación. El registro de la aplicación debe tener los siguientes permisos:
  - En Intune, configura scep_challenge_provider.
  - Para Microsoft Graph, defina Application.Read-All y User.Read.
- Debe conceder el consentimiento de administrador a la aplicación al registrarse en la aplicación. Para obtener más información, consulte Otorgar el consentimiento de administrador de todo el inquilino a una aplicación en la documentación de Microsoft Entra.

> **Tip**
>
> Al crear el registro de la aplicación, anote el ID de la aplicación (cliente) y el ID del directorio (inquilino) o el dominio principal. Cuando cree su conector para SCEP para Microsoft Intune, introducirá estos valores. Para obtener información sobre cómo obtener estos valores, consulte Crear una aplicación y un director de servicio de Microsoft Entra que pueda acceder a los recursos en la documentación de Microsoft Entra.

Paso 1: Otorgue AWS Private CA permiso para usar su aplicación Microsoft Entra ID

Tras crear un conector para SCEP para Microsoft Intune, debe crear una credencial federada en el registro de aplicaciones de Microsoft para que Connector for SCEP pueda comunicarse con Microsoft Intune.

Para configurar AWS Private CA como una CA externa en Microsoft Intune

1. En la consola Microsoft Entra ID, navega hasta los registros de aplicaciones.
2. Elija la aplicación que creó para utilizarla con Connector for SCEP. El ID de aplicación (cliente) de la aplicación en la que haga clic debe coincidir con el ID que especificó al crear el conector.
3. Seleccione Certificados y secretos en el menú desplegable Administrado.
4. Seleccione la pestaña Credenciales federadas.
5. Seleccione Añadir una credencial.
6. En el menú desplegable del escenario de credenciales federadas, selecciona Otro emisor.
7. Copie y pegue el valor del emisor de OpenID de los detalles de Connector for SCEP para Microsoft Intune en el campo Emisor. Para ver los detalles de un conector, selecciónelo en la lista de conectores para SCEP de la consola. AWS Como alternativa, puede obtener la URL llamando GetConnectory copiando el `Issuer` valor de la respuesta.
8. En Tipo, selecciona Identificador de asunto explícito.
9. Copie y pegue el valor del asunto OpenID del conector en el campo Valor. Puede ver el valor del emisor de OpenID en la página de detalles del conector de la consola. AWS Como alternativa, puede obtener la URL llamando GetConnectory, a continuación, copiar el `Audience` valor de la respuesta.
10. (Opcional) Introduce el nombre de la instancia en el campo Nombre. Por ejemplo, puede asignarle un nombre AWS Private CA.
11. (Opcional) Introduzca una descripción en el campo Descripción.
12. Copie y pegue el valor de OpenID Audience de los detalles de Connector for SCEP for Microsoft Intune en el campo Audience. Para ver los detalles de un conector, selecciónelo en la lista de conectores para SCEP de la consola. AWS Como alternativa, puede obtener la URL llamando GetConnectory copiando el `Subject` valor de la respuesta.
13. Seleccione Añadir.

Paso 2: configurar un perfil de configuración de Microsoft Intune

Tras conceder AWS Private CA el permiso para llamar a Microsoft Intune, debe usar Microsoft Intune para crear un perfil de configuración de Microsoft Intune que indique a los dispositivos que se pongan en contacto con Connector for SCEP para emitir el certificado.

1. Cree un perfil de configuración de certificados de confianza. Debe cargar el certificado de CA raíz de la cadena que está utilizando con Connector for SCEP en Microsoft Intune para establecer la confianza. Para obtener información sobre cómo crear un perfil de configuración de certificados de confianza, consulte Perfiles de certificados raíz de confianza para Microsoft Intune en la documentación de Microsoft Intune.
2. Cree un perfil de configuración de certificados SCEP que dirija sus dispositivos al conector cuando necesiten un certificado nuevo. El tipo de perfil del perfil de configuración debe ser un certificado SCEP. Para el certificado raíz del perfil de configuración, asegúrese de utilizar el certificado de confianza que creó en el paso anterior.

   Para el servidor SCEP URLs, copie y pegue la URL del SCEP de los detalles del conector en el campo Servidor SCEP. URLs Para ver los detalles de un conector, selecciónelo en la lista Conectores para el SCEP. Como alternativa, puede obtener la URL llamando y ListConnectors, a continuación, copiar el `Endpoint` valor de la respuesta. Para obtener instrucciones sobre la creación de perfiles de configuración en Microsoft Intune, consulte Crear y asignar perfiles de certificados SCEP en Microsoft Intune en la documentación de Microsoft Intune.

   > **Note**
   >
   > Para los dispositivos que no son Mac OS e iOS, si no establece un período de validez en el perfil de configuración, Connector for SCEP emite un certificado con una validez de un año. Si no establece un valor de uso extendido de claves (EKU) en el perfil de configuración, Connector for SCEP emite un certificado con el EKU establecido con. `Client Authentication (Object Identifier: 1.3.6.1.5.5.7.3.2)` En el caso de los dispositivos macOS o iOS, Microsoft Intune no respeta `ExtendedKeyUsage` `Validity` los parámetros de los perfiles de configuración. Para estos dispositivos, Connector for SCEP emite un certificado con un período de validez de un año para estos dispositivos mediante la autenticación del cliente.

Paso 3: Compruebe la conexión con el conector para SCEP

Después de crear un perfil de configuración de Microsoft Intune que apunte al punto final de Connector for SCEP, confirme que un dispositivo inscrito puede solicitar un certificado. Para confirmarlo, asegúrese de que no haya ningún error en la asignación de políticas. Para confirmarlo, en el portal de Intune, vaya a Dispositivos > Administrar dispositivos > Configuración y compruebe que no haya nada en la lista Errores de asignación de políticas de configuración. Si lo hay, confirme la configuración con la información de los procedimientos anteriores. Si la configuración es correcta y aún hay errores, consulte Recopilar los datos disponibles desde un dispositivo móvil.

Para obtener información sobre la inscripción de dispositivos, consulta ¿Qué es la inscripción de dispositivos? en la documentación de Microsoft Intune.

Configure Omnissa Workspace ONE for Connector para SCEP

Puede utilizarlo AWS Private CA como entidad emisora de certificados (CA) externa con el sistema Omnissa Workspace ONE UEM (Unified Endpoint Management). Esta guía proporciona instrucciones sobre cómo configurar Omnissa Workspace ONE después de crear un conector SCEP en. AWS

Requisitos previos

Antes de crear un conector SCEP para Omnissa Workspace ONE, debe cumplir los siguientes requisitos previos:

- Cree una CA privada en la consola. AWS Para obtener más información, consulte Cree una CA privada en AWS Private CA.
- Cree un conector SCEP de uso general. Para obtener más información, consulte Crear un conector.
- Tenga una cuenta de administrador de entorno de Omnissa Workspace ONE activa con un identificador de grupo organizativo.
- Si va a inscribir un dispositivo Apple, configure el servicio de notificaciones push de Apple (APNs) para MDM. Para obtener más información, consulta APNs los certificados en la documentación de Omnissa.

Paso 1: Defina una autoridad de certificación y una plantilla en Omnissa Workspace ONE

Tras crear un conector CA y SCEP privado en la AWS consola, defina la autoridad de certificación y la plantilla en Omnissa Workspace ONE.

Añádala AWS Private CA como entidad emisora de certificados

1. En el menú Sistema, elija Enterprise Integration y, a continuación, elija Autoridades de certificación.
2. Seleccione + AGREGAR y proporcione la siguiente información:
   - Nombre: AWS-Private-CA.
   - Descripción: AWS Private CA para la emisión de certificados de dispositivos.
   - Tipo de autoridad: seleccione el SCEP genérico.
   - URL del SCEP: Introduzca la URL del SCEP desde. AWS Private CA
   - Tipo de desafío: seleccione STATIC.
   - Desafío estático: introduzca la contraseña del desafío estático del SCEP desde el conector para la configuración del SCEP en la AWS consola.
   - Introduzca los valores de tiempo de espera de reintento y máximo de reintentos.
3. Guarde la configuración.

Cree una plantilla de certificado

1. En el menú Sistema, elija Enterprise Integration, elija Autoridades de certificación y, a continuación, elija Plantillas.
2. Seleccione Añadir plantillas y proporcione la siguiente información:
   - Nombre de la plantilla: Device-Cert-Template.
   - Autoridad de certificación: elija AWS-Private-CA.
   - Nombre del asunto: se trata de un campo personalizable. Puede elegir valores variables de una lista de atributos. Por ejemplo, CN= {DeviceReportedName}, O= {DevicePlatform}, OU= {1} CustomAttribute
   - Longitud de la clave privada: 2048 bits.
   - Tipo de clave privada: seleccione la firma y el cifrado según sea necesario

- Renovación automática: Enabled/Disabled (según sus necesidades).

3. Guarde la plantilla.

Paso 2: Configurar un perfil de Omnissa Workspace ONE UEM

Cree un perfil en Omnissa Workspace ONE UEM que dirija los dispositivos a Connector for SCEP para emitir un certificado.

Cree un perfil de dispositivo SCEP para la distribución de certificados

1. En el menú Recursos, elija Perfiles y líneas base y, a continuación, elija Perfiles.
2. Seleccione Añadir y, a continuación, Añadir perfil
3. Selecciona la plataforma del dispositivo (Android, iOS, macOS, Windows).
4. Defina el tipo de administración y el contexto según corresponda.
5. Establezca el nombre: Device-Cert-Profile.
6. Desplázate hasta SCEP Payload.
7. Seleccione SCEP y, a continuación, elija +Añadir.
8. Utilice la siguiente configuración:
   - SCEP:
     - En Fuente de credenciales, seleccione Autoridad de certificación definida (predeterminada).
     - En Autoridad de certificación, seleccione -Private-CAAWS
     - Para la plantilla de certificado, seleccione la plantilla de certificado del dispositivo definida en el paso 1.
9. Elija Siguiente y, en la sección Asignación, seleccione el grupo inteligente correcto de la lista (grupo de asignación para el dispositivo).
10. Seleccione Tipo de asignación como Automático para activar la renovación automática.
11. Guarde y publique el perfil.

> **Note**
>
> Para obtener más información, consulte el SCEP en la documentación de Omnissa.

Paso 3: inscriba los dispositivos en Omnissa Workspace ONE

Cree o verifique un grupo inteligente

1. En Grupos y configuraciones, elija Grupos y, a continuación, elija Grupos de tareas.
2. Cree o edite el grupo inteligente de dispositivos POC:
   - Nombre: POC-Devices.
   - Tipo de dispositivo: selecciona Todas o una plataforma específica (Android o iOS, por ejemplo).
   - Criterios: uso UserGroup, plataforma y sistema operativo, OEM y modelo para especificar los criterios para agrupar los dispositivos de destino.
   - Propiedad: seleccione cualquiera para dispositivos personales o corporativos.
3. Guarde y verifique que los dispositivos de destino aparezcan en la pestaña Vista previa.

Inscripción manual de dispositivos

Android

- Descarga la aplicación Workspace ONE Intelligent Hub de Google Play.
- Abre la aplicación e introduce la URL de inscripción o escanea un código QR.
- Inicie sesión y siga las instrucciones para inscribirse como dispositivo administrado por MDM.

iOS/macOS

- En el dispositivo, abre Safari y navega hasta la URL de inscripción (https://<Workspace ONEUEMHostname >/enroll, por ejemplo).
- Inicie sesión con las credenciales de usuario.
- Descargue e instale la aplicación Workspace ONE Intelligent Hub desde la App Store.
- Sigue las instrucciones para instalar el perfil MDM en Configuración > General > Administración de dispositivos y VPN > Perfil > Instalar.

Windows

- Descargue Workspace ONE Intelligent Hub desde el servidor Workspace ONE o desde Microsoft Store.
- Inscríbase a través del Hub mediante la URL de inscripción y las credenciales.

Asigne los dispositivos inscritos al grupo inteligente de dispositivos POC en Dispositivos > Vista de lista > Más acciones > Asignar al grupo inteligente.

Para obtener más información, consulte la inscripción automática de dispositivos en la documentación de Omnissa.

Verifica la inscripción

1. En la consola Omnissa Workspace ONE UEM, vaya a Dispositivos y, a continuación, a Vista de lista.
2. Confirme que sus dispositivos inscritos aparezcan con el estado establecido en Inscrito.
3. Verifique que los dispositivos estén en el grupo inteligente de dispositivos POC en la pestaña Grupos de los detalles del dispositivo.

Paso 4: Emitir un certificado

Activa la emisión de un certificado

1. En la vista de lista de dispositivos, seleccione el dispositivo inscrito.
2. Pulse el botón de consulta para solicitar el registro.
3. Device-Cert-ProfileDeben emitir un certificado mediante. AWS Private CA

Verifique la instalación del certificado

Android

Seleccione Configuración, Seguridad, Credenciales de confianza y, por último, Usuario para comprobar el certificado.

iOS

Vaya a Configuración, seleccione General, Administración de dispositivos y VPN y, por último, Perfil de configuración. Compruebe que el certificado de AWS-Private-CA esté presente.

macOS

Abra Keychain Access y, a continuación, System Keychain y verifique el certificado.

Windows

Abra certmgr.msc, luego Personal y, por último, Certificados para verificar el certificado.

Resolución de problemas

Errores del SCEP (por ejemplo, «22013: el servidor SCEP devolvió una respuesta no válida»)

- Compruebe que la URL del SCEP y la contraseña de desafío estática en Workspace ONE coincidan. AWS Private CA
- <SCEP_URL>Pruebe la conectividad del punto final del SCEP: curl.
- Compruebe AWS CloudTrail los registros para AWS Private CA ver si hay errores (`IssueCertificate`fallos, por ejemplo).

APNs problemas (iOS/macOS)

- Asegúrese de que el APNs certificado sea válido y esté asignado al grupo organizativo correcto.
- Pruebe la APNs conectividad: telnet gateway.push.apple.com 2195.

Fallos en la instalación del perfil

- Confirme que los dispositivos estén en el grupo inteligente correcto (dispositivos, luego Vista de lista y, por último, grupos).
- Forzar la sincronización de un perfil: Más acciones, luego Enviar y, por último, Lista de perfiles.

Registros

- Android: usa los registros de Logcat o Workspace ONE.
- iOS/macOS: log show --predicate 'process == "mdmclient"' --last 1h (via Xcode/ AppleConfigurador).
- Windows: Visor de eventos, luego Registros de aplicaciones y servicios y luego Microsoft-Windows-. DeviceManagement
- Workspace ONE UEM: Monitor, luego Informes y análisis, Eventos y, por último, Eventos del dispositivo.

Para obtener información detallada sobre el conector para la supervisión del SCEP AWS, consulte Monitor Connector para el SCEP.

Consideraciones de seguridad

- Guarde el SCEP URLs y sus secretos de forma segura. Para obtener más información, consulte el AWS Secrets Manager servicio.
- Restrinja los criterios de los grupos inteligentes a los dispositivos de destino únicamente.

- Renueva periódicamente los certificados de notificaciones push de Apple (APNs) (válidos durante 1 año).
- Establezca períodos cortos de validez de los certificados para los proyectos de prueba de concepto a fin de minimizar el riesgo.
- En el caso de los dispositivos personales, asegúrese de que la limpieza elimine todos los perfiles y certificados.

Para obtener información sobre cómo configurar la integración de Omnissa Workspace ONE UEM y CA mediante un conector SCEP, consulte la documentación sobre el SCEP en Omnissa Workspace ONE.

Seguridad en el conector para SCEP

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de los centros de datos y las arquitecturas de red diseñados para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre AWS usted y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube.
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos vigentes.

Esta documentación le ayuda a entender cómo aplicar el modelo de responsabilidad compartida al utilizar Connector for SCEP. Los siguientes temas muestran cómo configurar Connector for SCEP para cumplir sus objetivos de seguridad y conformidad.

Temas

- Conector para puntos finales de VPC SCEP ()AWS PrivateLink

Conector para puntos finales de VPC SCEP ()AWS PrivateLink

Puede crear una conexión privada entre la VPC y Connector for SCEP configurando un punto final de la interfaz de la VPC. Los puntos finales de la interfaz funcionan con una tecnología que permite acceder de AWS PrivateLinkforma privada a las operaciones de la API de Connector for SCEP. AWS PrivateLink enruta todo el tráfico de red entre su VPC y Connector for SCEP a través de la red Amazon, lo que evita la exposición en Internet abierta. Cada punto de enlace de la VPC está representado por una o varias interfaces de red elástica con direcciones IP privadas en las subredes de la VPC.

El punto final de la interfaz de la VPC conecta la VPC directamente a Connector for SCEP sin necesidad de una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o una conexión. Direct Connect Las instancias de su VPC no necesitan direcciones IP públicas para comunicarse con la API de Connector for SCEP.

Para usar Connector for SCEP a través de la VPC, debe conectarse desde una instancia que esté dentro de la VPC. Como alternativa, puede conectar la red privada a la VPC mediante un AWS Virtual Private Network (Site-to-Site VPN) o. Direct Connect Para obtener más información Site-to-Site VPN, consulte Conexiones VPN en la Guía del usuario de Amazon VPC. Para obtener información acerca de Direct Connect, consulte Creación de una conexión en la Guía del usuario de Direct Connect .

El conector para SCEP no requiere el uso de AWS PrivateLink, pero lo recomendamos como capa de seguridad adicional. Para obtener más información sobre AWS PrivateLink los puntos de enlace de la VPC, consulte Acceder a los servicios mediante. AWS PrivateLink

Consideraciones sobre el conector para puntos finales de VPC SCEP

Antes de configurar los puntos finales de la interfaz de VPC para Connector for SCEP, tenga en cuenta las siguientes consideraciones:

- Es posible que Connector for SCEP no admita puntos de enlace de VPC en algunas zonas de disponibilidad. Cuando cree un punto de conexión de VPC, compruebe primero el soporte en la consola de administración. Las zonas de disponibilidad no compatibles aparecen marcadas como “Service not supported in this Availability Zone” (El servicio no es compatible en esta zona de disponibilidad).
- Los puntos de conexión de VPC no admiten las solicitudes entre regiones. Asegúrese de crear el punto final en la misma región en la que creó el conector.

- Los puntos de conexión de VPC solo admiten DNS proporcionadas por Amazon a través de Amazon Route 53. Si desea utilizar su propio DNS, puede utilizar el enrutamiento de DNS condicional. Para obtener más información, consulte Conjuntos de opciones de DHCP en la Guía del usuario de Amazon VPC.
- El grupo de seguridad asociado al punto de conexión de la VPC debe permitir las conexiones entrantes en el puerto 443 desde la subred privada de la VPC.

Creación del punto final de VPC para Connector for SCEP

Puede crear un punto final de VPC para el servicio Connector for SCEP mediante la consola de VPC en o la. https://console.aws.amazon.com/vpc/ AWS Command Line Interface Para obtener más información, consulte el procedimiento de creación de un punto final de interfaz en la Guía del usuario de Amazon VPC. Connector for SCEP permite realizar llamadas a todas sus operaciones de API dentro de su VPC.

Al crear el punto final, especifíquelo `com.amazonaws.region.pca-connector-scep` como nombre del servicio.

Si ha habilitado los nombres de host DNS privados para el punto final, el punto final predeterminado de Connector for SCEP pasa ahora a ser su punto final de VPC. Para obtener una lista completa de los puntos de conexión de servicio predeterminados, consulte Puntos de enlace de servicio y cuotas.

Si no ha habilitado nombres de host de DNS privados, Amazon VPC proporciona un nombre de punto de conexión de DNS que puede utilizar en el siguiente formato:

```
vpc-endpoint-id.pca-connector-scep.region.vpce.amazonaws.com
```

Para obtener más información, consulte los puntos de enlace de la VPC (AWS PrivateLink) en la Guía del usuario de Amazon VPC.

Cree una política de puntos finales de VPC para Connector for SCEP

Puede crear una política para los puntos de enlace de Amazon VPC para Connector for SCEP para especificar lo siguiente:

- La entidad principal que puede realizar acciones
- Las acciones que se pueden realizar

- Los recursos en los que se pueden llevar a cabo las acciones

Para obtener más información, consulte Control del acceso a los servicios con puntos de enlace de VPC en la Guía de Amazon VPC.

Ejemplo: política de puntos finales de VPC para acciones de Connector for SCEP

Cuando se conecta a un punto final, la siguiente política otorga acceso a todos los principales al conector que aparece en la lista para realizar acciones del SCEP en el recurso de conector especificado.

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "pca-connector-scep:GetConnector",
            "pca-connector-scep:ListConnectors"
         ],
         "Resource": "arn:aws:pca-connector-scep:region:account:connector/connector-id"
      }
   ]
}
```

Creación de un punto final de VPC para las operaciones de inscripción de Connector para SCEP

Connector for SCEP proporciona un servicio de punto final de VPC independiente para operaciones de inscripción como `GetCACaps` y. `PKIOperation`

Al crear el punto final de inscripción, especifíquelo `com.amazonaws.region.pca-connector-scep.enroll` como nombre del servicio.

Al crear un conector, si lo desea, puede especificar un `VpcEndpointId` para restringir el conector para que solo sea accesible a través de ese punto final de VPC específico.

Si no ha habilitado nombres de host de DNS privados, Amazon VPC proporciona un nombre de punto de conexión de DNS que puede utilizar en el siguiente formato:

```
vpc-endpoint-id.enroll.pca-connector-scep.region.vpce.amazonaws.com
```

> **Note**
>
> Para acceder al conector, debes usar la URL del punto de conexión incluida en los detalles del conector, no el nombre DNS del punto de conexión de la VPC directamente. Sin embargo, puede reemplazar la parte del nombre DNS de la URL del punto de conexión por cualquier nombre de DNS del punto de conexión de la VPC válido, como un nombre DNS específico de AZ.
> Por ejemplo, para usar un nombre DNS específico para la zona de disponibilidad, puedes sustituirlo
>
> ```
> https://vpc-endpoint-id.enroll.pca-connector-scep.region.vpce.amazonaws.com/account-id-connector-id/UUID
> ```
>
> por
>
> ```
> https://vpc-endpoint-id-availability-zone.enroll.pca-connector-scep.region.vpce.amazonaws.com/account-id-connector-id/UUID
> ```

Ejemplo: política de puntos finales de VPC para operaciones de inscripción de Connector para SCEP

Puede adjuntar una política de puntos finales de VPC para controlar el acceso a las operaciones de inscripción. Cuando se conecta a un punto final, la siguiente política otorga acceso a todos los directores a las operaciones `GetCACaps` y`PKIOperation`. El recurso de la estrofa es un conector.

Las operaciones de inscripción de Connector for SCEP no se autentican con SiGv4. Debido a esto, no están asociados a un principal de IAM y, en cambio, las políticas de puntos finales de la VPC los consideran anónimos. Por lo tanto, su política de puntos finales de VPC debe permitir que todos los principales realicen estas acciones.

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "pca-connector-scep:GetCACaps",
            "pca-connector-scep:GetCACert",
            "pca-connector-scep:PKIOperation"
```

```
         ],
         "Resource": [
            arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566
         ]
      }
   ]
}
```

Conector de monitor para SCEP

La supervisión es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de Connector for SCEP y del resto de sus soluciones. AWS AWS proporciona las siguientes herramientas de supervisión para vigilar Connector for SCEP, informar cuando algo va mal y tomar medidas automáticas cuando sea necesario:

- AWS CloudTrail captura las llamadas a la API y otros eventos relacionados que realiza la Cuenta de AWS o que se realizan en nombre de esta. Además, entrega los archivos de registro a un bucket de Amazon S3 especificado. Puede identificar qué usuarios y cuentas llamaron AWS APIs, la dirección IP de origen desde la que se realizaron las llamadas y cuándo se produjeron.

  Si supervisa CloudTrail los eventos de datos, los registros contienen la lista de todas las solicitudes recientes de los dispositivos cliente. Los eventos de datos incluyen información que identifica el dispositivo cliente, como la dirección IP, el tipo de operación realizada y el código de error y el mensaje detallado si la operación produce un `failed` estado. Para obtener más información, consulte la Guía del usuario de AWS CloudTrail.
- Amazon EventBridge es un servicio de bus de eventos sin servidor que facilita la conexión de sus aplicaciones con datos de diversas fuentes. EventBridge ofrece un flujo de datos en tiempo real desde sus propias aplicaciones, aplicaciones Software-as-a-Service (SaaS) y AWS servicios, y dirige esos datos a destinos como Lambda y Logs. CloudWatch Esto le permite monitorear los eventos que ocurren en los servicios y crear arquitecturas basadas en eventos. Para obtener más información, consulta la Guía del EventBridge usuario de Amazon.

Temas

- Automatice el conector para SCEP mediante EventBridge
- Conector de registro para llamadas a la API SCEP mediante AWS CloudTrail

Automatice el conector para SCEP mediante EventBridge

Puedes usar Amazon EventBridge para automatizar tus AWS servicios y responder automáticamente a eventos del sistema, como problemas de disponibilidad de aplicaciones o cambios en los recursos. Los eventos de AWS los servicios se entregan casi EventBridge en tiempo real. Puedes escribir reglas sencillas para indicar qué eventos te interesan y las acciones automatizadas que debes tomar cuando un evento coincide con una regla. EventBridge se publican al menos una vez. Para obtener más información, consulte Crear reglas que reaccionen a los eventos de EventBridge.

CloudWatch Los eventos se convierten en acciones mediante EventBridge. Con EventBridge, puedes usar eventos para activar objetivos. Para obtener más información, consulta ¿Qué es Amazon EventBridge?

Conector para tipos de eventos SCEP

La emisión del certificado se realizó correctamente

Connector for SCEP envía un `Certificate Issuance Succeeded` evento EventBridge cuando emitimos un certificado en respuesta a una `PkiOperationPost` solicitud.

A continuación se muestran algunos datos de ejemplo del evento.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Issuance Succeeded",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "success",
       "requestType": "PkiOperationPost",
       "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
```

```
   }
}
```

Falló la emisión del certificado

Connector for SCEP envía un `Certificate Issuance Failed` evento EventBridge cuando no podemos emitir un certificado en respuesta a una `PkiOperationPost` solicitud.

A continuación se muestran algunos datos de ejemplo del evento.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Issuance Failed",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "failure",
       "requestType": "PkiOperationPost",
       "reason": "The certificate authority is not active."
   }
}
```

La recuperación del certificado de la autoridad certificadora se realizó correctamente

El conector para SCEP envía un `Certificate Authority Certificate Retrieval Succeeded` evento EventBridge cuando recibimos una `GetCACert` solicitud y recuperamos correctamente el certificado de CA privado del conector.

A continuación, se muestran algunos datos de ejemplo del evento.

```
{
   "version": "0",
```

```
   "id": "event_ID",
   "detail-type": "Certificate Authority Certificate Retrieval Succeeded",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "success",
       "requestType": "GetCACert"
   }
}
```

Falló la recuperación del certificado de la autoridad certificadora

El conector para SCEP envía un `Certificate Authority Certificate Retrieval Failed` evento EventBridge cuando recibimos una `GetCACert` solicitud y no podemos recuperar el certificado de CA privado del conector. El evento incluye el motivo del error.

A continuación se muestran algunos datos de ejemplo del evento.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Authority Certificate Retrieval Failed",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "failure",
       "requestType": "GetCACert",
```

```
       "reason": "The certificate authority certificate validity must be at least one
 year from today."
    }
}
```

La recuperación del certificado de la autoridad certificadora se realizó correctamente

El conector para SCEP envía un `Certificate Authority Certificate Retrieval Succeeded` evento EventBridge cuando recibimos una `GetCACert` solicitud y recuperamos correctamente el certificado de CA privado del conector.

A continuación, se muestran algunos datos de ejemplo del evento.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Authority Certificate Retrieval Succeeded",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "success",
       "requestType": "GetCACert"
   }
}
```

La recuperación de las capacidades de la autoridad de certificación se realizó correctamente

Connector for SCEP envía un `Certificate Authority Capabilities Retrieval Succeeded` evento EventBridge cuando recibimos una `GetCACaps` solicitud de SCEP y recuperamos correctamente las capacidades de la CA.

A continuación se muestran algunos datos de ejemplo del evento.

Falló la recuperación de las capacidades de la autoridad de certificación

Connector for SCEP envía un `Certificate Authority Capabilities Retrieval Failed` evento EventBridge cuando recibimos una `GetCACaps` solicitud de SCEP y no podemos recuperar las capacidades de la CA. En este caso, incluimos el motivo del error.

A continuación se muestran algunos datos de ejemplo del evento.

```
{
 "resources":
     [
     "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
     "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector11223344-1234-1122-2233-112233445566"
     ],
 "detailType":"Certificate Authority Capabilities Retrieval Failed",
 "detail": {
     "result":"failure",
     "requestType":"GetCACaps",
     "reason":"The request was denied due to request throttling."
 },
 "source":"aws.pca-connector-scep","accountId":"111122223333"
 }
```

Se ha invocado una operación no compatible

Se ha invocado una operación no compatible

El conector para SCEP envía un `Unsupported Operation Invoked` evento a EventBridge si la operación enviada al punto final del conector no es compatible o se desconoce.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Unsupported Operation Invoked",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
```

```
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {}
}
```

Cree una regla EventBridge

En EventBridge, puede crear reglas que respondan a los eventos registrados por CloudTrail. Para crear una regla que incluya todos los eventos registrados por Connector for SCEP, defina `aws.pca-connector-scep` el origen en. Para obtener más información sobre las reglas, consulta Crear una regla en Amazon EventBridge.

Conector de registro para llamadas a la API SCEP mediante AWS CloudTrail

El conector para el protocolo simple de inscripción de certificados (SCEP) está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, función, cliente o servicio. AWS CloudTrail captura todas las llamadas a la API de Connector for SCEP como eventos. Las llamadas capturadas incluyen las llamadas desde la consola de Connector para el SCEP y las llamadas en código al Connector para las operaciones de la API del SCEP. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos de Connector for SCEP. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Connector para el SCEP, la dirección IP desde la que se realizó la solicitud, quién la hizo, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la Guía del AWS CloudTrail usuario.

Conector para obtener información sobre el SCEP en CloudTrail

CloudTrail está activado en su cuenta Cuenta de AWS al crear la cuenta. Cuando se produce una actividad en Connector for SCEP, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Puede ver, buscar y descargar los eventos recientes en su Cuenta de AWS. Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para obtener un registro continuo de los eventos que se han producido en su Cuenta de AWS territorio, incluidos los eventos de Connector for SCEP, cree un registro. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea

un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las Regiones de AWS. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para más información, consulte los siguientes temas:

- Introducción a la creación de registros de seguimiento
- CloudTrail servicios e integraciones compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones y recibir archivos de CloudTrail registro de varias cuentas

Todas las acciones de Connector for SCEP se registran CloudTrail y se documentan en la referencia de la API de Connector for SCEP. Por ejemplo, las llamadas `GetConnector` y las `CreateConnector` `CreateChallenge` acciones generan entradas en los archivos de CloudTrail registro.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de usuario root o AWS Identity and Access Management (IAM).
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.
- Si la solicitud la realizó un dispositivo cliente del SCEP.

Para obtener más información, consulte el elemento userIdentity de CloudTrail .

Conector para eventos de gestión del SCEP

Connector for SCEP se integra CloudTrail para registrar las acciones de la API realizadas por un usuario, un rol o un AWS servicio en Connector for SCEP. Puede usarlo CloudTrail para monitorear Connector en busca de solicitudes de API SCEP en tiempo real y almacenar registros en Amazon Simple Storage Service, Amazon CloudWatch Logs y Amazon CloudWatch Events. Connector for SCEP permite registrar las siguientes acciones como eventos en archivos de CloudTrail registro:

- CreateChallenge
- CreateConnector
- GetConnector
- GetChallengeMetadata
- GetChallengePassword
- DeleteConnector
- DeleteChallenge

Conector para eventos de datos SCEP en CloudTrail

Los eventos de datos proporcionan información sobre las operaciones de recursos que se realizan en o dentro de un recurso, por ejemplo, cuando el cliente envía un `GetCACaps` mensaje SCEP a un punto final del conector. Se denominan también operaciones del plano de datos. Los eventos de datos suelen ser actividades de gran volumen. De forma predeterminada, CloudTrail no registra ningún CloudTrail evento de datos y el historial de eventos no lo registra.

Se aplican cargos adicionales a los eventos de datos. Para obtener más información sobre CloudTrail los precios, consulta AWS CloudTrail Precios.

Puede registrar eventos de datos para el tipo de `AWS::PCAConnectorSCEP::Connector` recurso mediante la CloudTrail consola o las operaciones de la CloudTrail API. AWS CLI Para obtener más información sobre cómo registrar los eventos de datos, consulte Registro de eventos de datos con la Consola de administración de AWS y Registro de eventos de datos con la AWS Command Line Interface en la Guía del usuario de AWS CloudTrail .

En la siguiente tabla se muestra el tipo de recurso Connector for SCEP para el que puede registrar eventos de datos. La columna Tipo de evento de datos (consola) muestra el valor que se puede elegir en la lista de tipos de eventos de datos de la CloudTrail consola. La columna de valores resources.type muestra el `resources.type` valor que se debe especificar al configurar los selectores de eventos avanzados mediante o. AWS CLI CloudTrail APIs La CloudTrail columna Datos APIs registrados muestra las llamadas a la API registradas CloudTrail para el tipo de recurso.

Tipo de evento de datos (consola)	resources.type value	Datos APIs registrados en CloudTrail
Kinesis - S3	`AWS::PCAConnectorSCEP::Connector`	• `PKIOperationGet` - Se genera si se envía una

Tipo de evento de datos (consola)	resources.type value	Datos APIs registrados en CloudTrail
		solicitud HTTP GET SCEP que contiene un `PKCSReq` mensaje al punto final del plano de datos de un conector y el funcionam iento de ese mensaje está configurado en. `PKIOperat ion` • `PKIOperationPost` - Se genera si se envía una solicitud HTTP POST SCEP que contiene un `PKCSReq` mensaje al punto final del plano de datos de un conector y el funcionam iento de ese mensaje está configurado en. `PKIOperat ion` • `GetCACaps` - Se genera si se envía una solicitud SCEP que contiene un `GetCACaps` mensaje al punto final del plano de datos de un conector. • `GetCACert` - Se genera si se envía una solicitud de SCEP que contiene un `GetCACert` mensaje al punto final del plano de datos de un conector.

Puede configurar selectores de eventos avanzados para filtrar según los campos `eventName`, `readOnly` y `resources.ARN` y así registrar solo los eventos que son importantes para usted. El siguiente ejemplo es la vista JSON de una configuración de eventos de datos que registra eventos solo para una función específica. Para obtener más información acerca de estos campos, consulte AdvancedFieldSelector en la Referencia de la API de AWS CloudTrail .

```
[
  {
    "name": "connector-scep-events",
    "fieldSelectors": [
      {
        "field": "eventCategory",
        "equals": [
          "Data"
        ]
      },
      {
        "field": "resources.type",
        "equals": [
          "AWS::PCAConnectorSCEP::Connector"
        ]
      },
      {
        "field": "resources.ARN",
        "equals": [
          "arn:aws:pca-connector-scep:US West (N.
 California):111122223333:connector/11223344-1122-2233-3344-cae95a00d2a7"
        ]
      }
    ]
  }
]
```

Ejemplos de entradas

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

Ejemplo 1: evento de gestión, **`CreateConnector`**

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `CreateConnector` acción.


```
{
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
          "principalId": "AABB1122CCDD4455HHJJ1:11cc33nn2a97724dc48a89071111111111",
          "arn": "arn:aws:sts::111122223333:assumed-role/Admin",
          "accountId": "111122223333",
          "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "AABB1122CCDD4455HHJJ1",
                  "arn": "arn:aws:iam::111122223333:role/Admin",
                  "accountId": "111122223333",
                  "userName": "my-user-name"
              },
              "attributes": {
                  "creationDate": "2024-08-16T17:46:41Z",
                  "mfaAuthenticated": "false"
              }
          }
        },
        "eventTime": "2024-08-16T17:48:07Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "CreateConnector",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "requestParameters": {
          "ClientToken": "11223344-2222-3333-4444-666555444555",
          "CertificateAuthorityArn": "arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
        },
        "responseElements": {
          "ConnectorArn": "arn:aws:pca-connector-scep:us-east-1:111122223333:connector/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
        },
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
```

```
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
        }
```

Ejemplo 2: Evento de administración, **`CreateChallenge`**

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `CreateChallenge` acción.

```
{
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
          "principalId": "AABB1122CCDD4455HHJJ1:11cc33nn2a97724dc48a89071111111111",
          "arn": "arn:aws:sts::111122223333:assumed-role/Admin",
          "accountId": "111122223333",
          "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "AABB1122CCDD4455HHJJ1",
                  "arn": "arn:aws:iam::111122223333:role/Admin",
                  "accountId": "111122223333",
                  "userName": "user-name"
              },
              "attributes": {
                  "creationDate": "2024-08-16T17:46:41Z",
                  "mfaAuthenticated": "false"
              }
          }
        },
        "eventTime": "2024-08-16T17:47:52Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "CreateChallenge",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "requestParameters": {
          "ConnectorArn": "arn:aws:pca-connector-scep:us-east-1:111122223333:connector/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
```

```
          "ClientToken": "11223344-2222-3333-4444-666555444555"
        },
        "responseElements": {
          "Challenge": {
              "Arn": "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/9cac40bc-acba-412e-9a24-f255ef2fe79a/a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222",
              "ConnectorArn": "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
              "CreatedAt": 1723830472.942,
              "Password": "***",
              "UpdatedAt": 1723830472.942
          }
        },
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
        }
```

Ejemplo 3: Evento de administración, **`GetChallengePassword`**

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `GetChallengePassword` acción.

```
{
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
          "principalId": "AABB1122CCDD4455HHJJ1:11cc33nn2a97724dc48a89071111111111",
          "arn": "arn:aws:sts::111122223333:assumed-role/Admin",
          "accountId": "111122223333",
          "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "AABB1122CCDD4455HHJJ1",
                  "arn": "arn:aws:iam::111122223333:role/Admin",
                  "accountId": "905418114790",
                  "userName": "111122223333"
```

```
            },
            "attributes": {
                "creationDate": "2024-08-16T17:55:01Z",
                "mfaAuthenticated": "false"
            }
          },
          "invokedBy": "signin.amazonaws.com"
        },
        "eventTime": "2024-08-16T17:55:54Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "GetChallengePassword",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "requestParameters": {
          "ChallengeArn": "arn:aws:pca-connector-scep:us-east-1:111122223333:challenge/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
        },
        "responseElements": null,
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
        }
```

Ejemplo 4: Evento de datos, **`PkiOperationPost`**

El siguiente ejemplo muestra una entrada de CloudTrail registro que muestra una `PkiOperationPost` llamada fallida. El registro incluye un código de error y un mensaje de error con una explicación del error.

```
{
        "eventVersion": "1.10",
        "userIdentity": {
          "type": "FederatedUser",
          "principalId": "111122223333",
          "accountId": "111122223333"
        },
        "eventTime": "2024-08-16T17:40:09Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
```

```
        "eventName": "PkiOperationPost",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "errorCode": "BadRequestException",
        "errorMessage": "The certificate authority is not in a valid state for issuing
 certificates (Service: AcmPca, Status Code: 400, Request ID: a1b2c3d4-5678-90ab-cdef-
EXAMPLE55555)",
        "requestParameters": null,
        "responseElements": null,
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
        "resources": [
          {
              "accountId": "111122223333",
              "type": "AWS::PCAConnectorSCEP::Connector",
              "ARN": "arn:aws:pca-connector-scep:us-east-1:111122223333:connector/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
          }
        ],
        "eventType": "AwsApiCall",
        "managementEvent": false,
        "recipientAccountId": "905418114790",
        "eventCategory": "Data",
        "tlsDetails": {
          "clientProvidedHostHeader": "111122223333-a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333.enroll.pca-connector-scep.us-east-1.api.aws"
        }
        }
```


Solucione problemas de AWS Private Certificate Authority SCEP con el conector

Es posible que necesite solucionar problemas relacionados con la implementación de Connector for SCEP. En este capítulo se proporciona información detallada sobre los errores de HTTP y de cliente enviados por el servicio.

Temas

- Solucione los errores de HTTP de Connector for SCEP
- Solucione los errores del cliente SCEP en Connector

Solucione los errores de HTTP de Connector for SCEP

Cuando el cliente activa una acción de la API del plano de datos de Connector for SCEP y se produce un error, Connector for SCEP envía un código de respuesta HTTP al cliente solicitante con información sobre el error.

Además de las respuestas del servicio que se proporcionan directamente a sus clientes, puede utilizar las herramientas de supervisión que se describen en la Conector de monitor para SCEP sección para ver y depurar los errores que provocan un error de HTTP.

A continuación, se indican los mensajes de error que el servicio devuelve a los clientes del SCEP, las posibles causas y las medidas que puede adoptar para resolver los problemas.

Solicitud incorrecta de HTTP 400

Un código de respuesta HTTP 400 significa que Connector for SCEP no puede procesar la solicitud debido a un error aparente del cliente, como la falta o invalidez de datos en la solicitud. Si el error se debe a un error específico del protocolo SCEP, Connector for SCEP incluye la respuesta del SCEP en formato binario en el mensaje. El conector para el SCEP APIs puede devolver 400 respuestas por cualquiera de las siguientes razones.

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
LimitExceededException	Se ha superado el límite de emisión de la autoridad de certificación.	La autoridad de certificación (CA) privada asociada al conector ha superado su cuota en cuanto al número de certificados que puede emitir.	Un conector SCEP solo se puede conectar a una CA privada durante su vida útil. Si ha agotado los límites de su CA privada, cree un conector nuevo o solicite un aumento de cuota.	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
			Para obtener más información sobre las cuotas de CA privadas, consulte AWS Private Certificate Authority cuotas.	
ValidationException	La solicitud debe contener base64.	Connector for SCEP no puede procesar la solicitud HTTP GET porque el cuerpo no es válido en Base64.	Si es posible, configure sus clientes para que usen mensajes HTTP POST en lugar de mensajes HTTP GET. Si debe usar HTTP GET, los mensajes deben usar el formato Base64. Si sus clientes no cumplen estos requisitos, póngase en contacto con nosotros AWS Supportpara obtener ayuda.	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ValidationException	La autoridad de certificación no está activa.	La CA privada asociada al conector está inactiva.	Reactive la CA privada. Para obtener información, consulte Actualice una CA privada en AWS Private Certificate Authority.	No
ValidationException	La validez del certificado de la autoridad de certificación debe ser de al menos un año a partir de hoy.	La CA privada asociada al conector de uso general debe tener un período de validez de un año a partir de hoy.	Vuelva a emitir el certificado con un período de validez superior a un año a partir de hoy. Para obtener información sobre la administración de certificados, consulteAdministre el ciclo de vida de la CA privada.	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ValidationException	El certificado incluido en la solicitud ha caducado.	El certificado transitorio generado por el dispositivo cliente en cada transacción expiró al recibirlo el servicio.	Lo más probable es que los dispositivos cliente no tengan la configuración horaria correctamente configurada y que estén creando certificados con fechas atrasadas respecto a la hora real. Si no puedes resolver este problema, ponte en contacto con nosotros AWS Supportpara obtener ayuda.	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ValidationException	La solicitud contiene una sintaxis de mensaje criptográfico no válida.	El servicio no ha podido decodificar el mensaje de solicitud del SCEP.	Compruebe si los mensajes del SCEP se ajustan a la sintaxis de los mensajes criptográficos definida en la RFC 8894 del SCEP. Si no puede resolver este problema, póngase en contacto con nosotros para obtener ayuda. AWS Support	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ValidationException	El conector no está activo.	El estado del conector no está activo.	Puedes encontrar el estado de un conector en la consola o en el campo Estado de la API. El estado de un conector puede ser de creación, activo, borrado o fallido. Si el estado es de creación, prueba con la solicitud más tarde. Si el estado es erróneo, consulta el motivo del estado para solucionar el problema y, a continuación, crea un conector nuevo.	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ValidationException	Debe haber un certificado válido incluido en la solicitud.	El certificado transitorio incluido en el mensaje de solicitud del cliente faltaba o no era válido.	Los clientes compatibles con el CEP deben proporcionar un certificado autofirmado para autenticarse. Si su cliente no puede proporcionar el certificado autofirmado requerido, póngase en contacto con nosotros para obtener ayuda. AWS Support	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ValidationException	El URI de la solicitud no es válido.	Connector for SCEP no puede analizar la solicitud porque la ruta URI o la consulta de la solicitud no son válidas.	Los administradores deben verificar los ajustes de configuración de los dispositivos cliente, que normalmente se administran mediante un sistema de administración de dispositivos móviles (MDM). Para obtener más información, consulte Paso 2: Copie los detalles del conector en su sistema MDM.	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ValidationException	Se requiere exactamente un encabezado de host en la solicitud.	El cliente no proporcionó un encabezado de host HTTP válido en la solicitud, que es necesario para procesar la solicitud.	El encabezado del host HTTP es necesario para distinguir las solicitudes que llegan a distintos conectores. Si su cliente no puede proporcionar el encabezado de host HTTP requerido, póngase en contacto con nosotros AWS Supportpara obtener ayuda.	No
ValidationException	No se ha podido decodificar la solicitud. Envíe una solicitud de SCEP válida.	El servicio no pudo decodificar ni procesar la solicitud de sintaxis de mensajes criptográficos (CMS) que envió su cliente.	Si sus clientes tienen problemas con nuestra implementación del SCEP, anote el identificador de solicitud (`x-amzn-requestid`) que aparece en la respuesta y en el contacto. AWS Support	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ValidationException	No se pudo codificar la respuesta con valores derivados de la solicitud. Envíe una solicitud de SCEP válida.	El servicio no pudo codificar la respuesta del SCEP.	Este problema suele producirse cuando el servicio no puede utilizar el certificado del solicitante proporcionado para codificar correctamente el mensaje de respuesta del SCEP. Esto puede ocurrir, por ejemplo, si el certificado del solicitante tiene una clave del algoritmo de firma digital de curva elíptica (ECDSA), que Connector for SCEP no admite. Si encuentra este problema, primero configure su cliente MDM o SCEP para que utilice RSA. Si sigues sin poder resolver el problema, anota el identificador de solicitud (`x-amzn-requestid`) que aparece en la	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
			respuesta y ponte en contacto con nosotros AWS Supportpara solicitar ayuda.	

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ValidationException	Algoritmo no compatible: <OID>	La solicitud estaba firmada o cifrada mediante un algoritmo criptográfico no compatible.	Nuestro servicio no admite determinados algoritmos criptográficos obsoletos y débiles. Esta información se comunica a los clientes a través de la `GetCACaps` solicitud. Sin embargo, es posible que algunos clientes no utilicen este método para comprobar los algoritmos compatibles. Si sus clientes parecen ser incompatibles con los algoritmos criptográficos compatibles con nuestro servicio, póngase en contacto con nosotros AWS Supportpara obtener ayuda.	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ValidationException	PkiOperation MessageType no compatible.	El mensaje de solicitud contenía un tipo de `PkiOperation` mensaje no válido y el servicio no lo pudo procesar.	Nuestro servicio solo admite un subconjunto de los tipos de mensajes del protocolo SCEP definidos en el RFC 8894. En concreto, reconocemos y procesamos los siguientes tipos de mensajes: CertRep, PKCSReq GetCert, GetCRL y. CertPoll Comunicamos los tipos de mensajes admitidos a los clientes mediante el método GetCACaps . Lamentablemente, es posible que algunos clientes no utilicen este método y que no cumplan con las capacidades de nuestro servicio. Si sus clientes parecen ser incompatibles con los tipos	No

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
			de mensajes del SCEP admitidos por nuestro servicio, póngase en contacto con nosotros. AWS Support	
BadRequestException	La contraseña de desafío no es válida.	La contraseña de desafío proporcionada por el cliente no era válida para el punto final del servicio contactado y su conector asociado. La contraseña de desafío es una medida de seguridad obligatoria definida en el protocolo SCEP para garantizar que solo los clientes autorizados puedan acceder al servicio.	Asegúrese de que su cliente proporciona la contraseña de desafío correcta en su solicitud. Puedes consultarlos en los detalles del conector en la consola o a través de la GetChallengePasswordAPI. Para obtener más información, consulte Paso 2: Copie los detalles del conector en su sistema MDM.	Sí

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
BadRequestException	Se requiere exactamente una contraseña de desafío en la solicitud de firma del certificado.	El cliente proporcionó cero o varias contraseñas de desafío en su solicitud.	Asegúrese de que su cliente proporciona una contraseña de seguridad en su solicitud. Puedes encontrar las contraseñas de desafío en los detalles del conector en la consola o a través de la GetChallengePasswordAPI. Para obtener más información, consulte Paso 2: Copie los detalles del conector en su sistema MDM.	Sí
BadRequestException	El conector no tiene acceso a Azure.	Connector para Microsoft Intune autoriza las solicitudes de los clientes a través de Microsoft Intune. Esto requiere que concedas permiso a Connector for SCEP para acceder a tus recursos de Azure.	Configure los permisos que se detallan enPaso 1: Otorgue AWS Private CA permiso para usar su aplicación Microsoft Entra ID.	Sí

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
BadRequestException	La aplicación Azure no tiene acceso para funcionar<action>.	Connector para Microsoft Intune autoriza las solicitudes de los clientes a través de Microsoft Intune. Esto requiere que concedas permiso a Connector for SCEP para acceder a tus recursos de Azure.	Configure los permisos que se detallan enPaso 1: Otorgue AWS Private CA permiso para usar su aplicación Microsoft Entra ID.	Sí
BadRequestException	No se encontró la aplicación Azure.	Connector para Microsoft Intune autoriza las solicitudes de los clientes a través de Microsoft Intune. Este error indica que no tienes un registro de aplicación en tu ID de Microsoft Entra o que los detalles de Intune del conector están mal configurados.	Sigue las instrucciones del tema. Configurar Microsoft Intune para el conector para SCEP	Sí

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
BadRequestException	Falló la validación de la solicitud de firma de certificados de Intune. Motivo: <reason>	Connector para Microsoft Intune autoriza las solicitudes de los clientes a través de Microsoft Intune. Este mensaje de error indica que el proceso de validación de Intune ha fallado y se proporciona el código de error de Intune correspondiente.	Siga las instrucciones del tema. Configurar Microsoft Intune para el conector para SCEP Si el problema persiste, ponte en contacto con Microsoft Support.	Sí

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
BadRequestException	PkiOperation <message type>MessageType no compatible:.	El mensaje de solicitud contenía un tipo de mensaje no válido y el servicio no pudo procesarlo.	Nuestro servicio solo admite un subconjunto de los tipos de mensajes del protocolo SCEP definidos en el RFC 8894. En concreto, reconocemos y procesamos los siguientes tipos de mensajes: CertRep, PKCSReq GetCert, GetCRL y. CertPoll Comunicamos los tipos de mensajes admitidos a los clientes mediante el método GetCACaps . Lamentablemente, es posible que algunos clientes no utilicen este método y que no cumplan con las capacidades de nuestro servicio. Si sus clientes parecen ser incompatibles con los tipos de mensajes del	Sí

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
			SCEP admitidos por nuestro servicio, póngase en contacto con nosotros. AWS Support	
BadRequestException	No se admite el algoritmo o la longitud de la clave.	El servicio no admite la clave pública proporcionada incluida en la solicitud de firma del certificado.	Nuestro servicio solo admite claves RSA estándar de hasta 16.384 bits y claves ECDSA de hasta 521 bits. Si sus clientes requieren el uso de un algoritmo actualmente no compatible, póngase en contacto con nosotros para obtener ayuda. AWS Support	Sí

HTTP 401 No autorizado

Un código de estado de respuesta 401 sin autorización indica que la solicitud del cliente no se ha completado porque carece de credenciales de autenticación válidas para el recurso solicitado.

Cabecera de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
AccessDeniedException	El conector no tiene acceso a la autoridad de certificación.	El conector para el SCEP no tiene acceso a la CA privada asociada al conector.	Comparta su CA privada con el conector para el SCEP mediante. AWS Resource Access Manager	No
AccountDoesNotExistException	La AWS cuenta no existe.	El recurso Connector for SCEP ya no existe.	Se ha eliminado la cuenta propietaria del recurso de destino. Si lo hizo por error, póngase en contacto con usted AWS Supportdentro del período de 90 días posterior al cierre.	No

No se ha encontrado el HTTP 404

Un código de respuesta HTTP 404 normalmente significa que no se ha encontrado el recurso que estaba buscando.

Encabezado de respuesta (x-amzn-ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?

Encabezado de respuesta (x-amzn-ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ResourceNotFoundException	La autoridad de certificación no existe.	Se ha eliminado la CA privada asociada al conector.	Existe un período de gracia durante el cual se puede restaurar una autoridad de certificación (CA) privada si se ha eliminado por error. Para obtener más información, consulte Restaure una CA privada.	No
ResourceNotFoundException	<URL>No existe un conector con punto final.	El dispositivo cliente ha intentado conectarse a una URL que no pertenece a ningún conector existente.	Asegúrese de que el cliente proporciona el punto final correcto para el conector. Para ver el conector`Endpoint`, llama a la GetConnectorAPI o búscalo en la página de detalles del conector en la consola.	No

Conflicto de HTTP 409

Una respuesta a un conflicto de HTTP 409 indica que una CA privada asociada a un conector ha cambiado desde que se inició la solicitud.

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ConflictException	El conector ha cambiado desde que se inició la solicitud.	Se actualizó la CA privada asociada al conector, lo que provocó una rotación del certificado interno del conector utilizado para la comunicación con los dispositivos cliente a través del SCEP. Esta rotación de certificados puede provocar problemas temporales durante el período de actualización, a medida que se vaya implementando el nuevo certificado. Sin embargo, este error debe resolverse automáticamente de manera oportuna.	Vuelva a intentar realizar la solicitud en unos minutos. Si el problema no se resuelve, ponte en contacto con nosotros AWS Supportpara obtener ayuda.	No

HTTP 429: Demasiadas solicitudes

Connector for SCEP tiene cuotas a nivel de cuenta por región. Si supera el límite de solicitudes a un conector, se rechazarán y se generará un error HTTP 429. Si necesitas aumentar tu cuota, consulta AWS Private Certificate Authority puntos de conexión y cuotas.

Encabezado de respuesta (x-amzn-) ErrorType	Mensaje de error (x-amzn-) ErrorMessage	Causa raíz	Corrección	¿Incluye la respuesta del SCEP?
ThrottlingException	La solicitud fue denegada debido a una limitación de la solicitud.	Se han enviado demasiadas solicitudes a este conector, lo que ha provocado el rechazo de algunas solicitudes. Esta rotación de certificados puede provocar problemas temporales durante el período de actualización, a medida que se vaya implementando el nuevo certificado. Sin embargo, este error debe resolverse automáticamente de manera oportuna.	Si supera el límite de solicitudes a un conector, se rechazarán sus solicitudes. Si necesitas aumentar tu cuota, consulta Connector para ver los puntos finales y las cuotas del SCEP.	No

Solucione los errores del cliente SCEP en Connector

Utilice la siguiente guía para solucionar los errores del cliente relacionados con Connector for SCEP.

Ejemplo de mensaje	Causa raíz	Solución
	El conector está conectado a una CA privada que utiliza una clave	Considere la posibilidad de utilizar una CA privada cifrada con RSA en

Ejemplo de mensaje	Causa raíz	Solución
No se admiten las claves ECDSA	ECDSA en lugar de RSA. Si bien este servicio admite claves ECDSA, es posible que no todos los dispositivos cliente sean compatibles con este algoritmo.	lugar de una ECDSA. Si crea una CA privada que utilice RSA, también tendrá que crear un conector nuevo. Un conector solo se puede vincular a una CA privada durante su vida útil.
El certificado de cifrado o firma no está presente	Según el RFC 8894, un servicio SCEP devuelve los certificados de CA intermedios al cliente. El cliente utiliza estos certificados para realizar operaciones de cifrado y validación de firmas como parte del protocolo SCEP. Connector for SCEP utiliza el mismo certificado tanto para el cifrado como para la validación de la firma, lo cual es un enfoque común. Sin embargo, es posible que algunos clientes esperen tener dos certificados independientes en su lugar.	Si no puede utilizar clientes compatibles, póngase en contacto con nosotros AWS Supportpara obtener ayuda.

AWS Private CA cuotas de servicio

Autoridad de certificación privada de AWS asigna cuotas al número permitido de certificados y autoridades de certificación. Las tasas de solicitud de acciones de la API también están sujetas a cuotas. Autoridad de certificación privada de AWS las cuotas son específicas de una AWS cuenta y una región.

Autoridad de certificación privada de AWS limita las solicitudes de la API a diferentes velocidades según el funcionamiento de la API. La limitación significa que se Autoridad de certificación privada de AWS rechaza una solicitud que de otro modo sería válida porque la solicitud supera la cuota de la operación en cuanto al número de solicitudes por segundo. Cuando una solicitud se limita, devuelve un error. Autoridad de certificación privada de AWS ThrottlingException Autoridad de certificación privada de AWS no garantiza una tasa mínima de solicitudes para. APIs

Para ver qué cuotas se pueden ajustar, consulte la tabla de Autoridad de certificación privada de AWS cuotas del Referencia general de AWS.

Puede ver sus cuotas actuales y solicitar aumentos de cuota mediante AWS Service Quotas.

Para ver una up-to-date lista de tus Autoridad de certificación privada de AWS cuotas

1. Inicie sesión en su AWS cuenta.
2. Abra la consola de Service Quotas en https://console.aws.amazon.com/servicequotas/.
3. En la lista de servicios, elija AWS Certificate Manager Private Certificate Authority (ACM PCA). Cada cuota de la lista de Service Quotas muestra el valor de cuota aplicado actualmente, el valor de cuota predeterminado y si la cuota es ajustable o no. Elija el nombre de una cuota para obtener más información sobre ella.

Cómo solicitar un aumento de cuota

1. En la lista de Service Quotas, seleccione el botón de opción para obtener una cuota ajustable.
2. Elija el botón Solicitar aumento de cuota.
3. Complete y envíe el formulario Solicitar aumento de cuota.

Autoridad de certificación privada de AWS está integrado con AWS Certificate Manager. Puede usar la consola de ACM o la API de ACM para solicitar certificados privados de una CA privada existente.

AWS CLI Estos certificados de PKI privados, gestionados por ACM, están sujetos tanto a las cuotas de PCA como a las cuotas que ACM impone a los certificados públicos e importados. Para obtener más información sobre los requisitos de ACM, consulte Solicitar un certificado privado y cuotas en la Guía del AWS Certificate Manager usuario.

Historial de revisiones del documento

En la siguiente tabla se describen cambios importantes en esta documentación desde enero de 2018. Además de los cambios importantes que se indican a continuación, también actualizamos la documentación con frecuencia para mejorar las descripciones y los ejemplos y para dar cuenta de los comentarios que nos envía. Si desea recibir notificaciones sobre cambios importante, utilice el enlace de la esquina superior derecha para suscribirse a la fuente RSS.

Cambio	Descripción	Fecha
Actualización de la documentación	Se actualizó Secure Kubernetes AWS Private Certificate Authority con un nuevo procedimiento de introducción, ejemplos y temas de supervisión y solución de problemas.	1 de octubre de 2025
Compatibilidad con doble pila	AWS Private Certificate Authority admite doble pila.	23 de junio de 2025
La compatibilidad con dominios secundarios para Connector for AD ya está disponible de forma general	Ahora puedes configurar Connector para AD con tu dominio secundario.	2 de junio de 2025
Nueva política gestionada: `AWSPrivateCAConnectorForKubernetesPolicy`	Se introdujo una nueva política gestionada para su uso con AWS Private CA Connector for Kubernetes.	19 de mayo de 2025
Políticas actualizadas `AWSPrivateCAPrivilegedUser` y administradas `AWSPrivateCAUser`	`StringLike` Reemplazado por `ArnLike` in `AWSPrivateCAUser` y`AWSPrivateCAPrivilegedUser` . Se actualizó el ARN de la plantilla para incluir comodines	22 de enero de 2025

	`arn:aws:acm-pca:::template` en. `arn:aws:acm-pca:*:*:template`	
El conector para el SCEP ya está disponible de forma general	El conector para el SCEP ya está disponible de forma general.	16 de septiembre de 2024
Nuevo tema de solución de problemas	Se ha agregado un tema nuevo que le ayuda a solucionar problemas relacionados con la actualización de las plantillas de Connector para Active Directory.	31 de julio de 2024
Se agregó la forma de actualizar las plantillas de Connector para AD	Se agregó un procedimiento que describe cómo actualizar una plantilla de Connector for AD y cómo AWS Private CA se propagan esas actualizaciones.	31 de julio de 2024
El conector SCEP para Omnissa Workspace ONE ya está disponible de forma general	El conector SCEP para Omnissa Workspace ONE ya está disponible de forma general.	23 de julio de 2024
Se ha añadido una restricción para los informes de auditoría	AWS Private CA no admite el uso de Amazon S3 Object Lock con los buckets que se utilizan para los informes de auditoría.	3 de julio de 2024
Ahora es compatible con SM2 la región de China	AWS Private CA ahora es compatible con el algoritmo de SM2 firma, solo para la región de China.	27 de junio de 2024

AWS Private CA ahora es compatible con Connector for SCEP (versión preliminar)	Utilice Connector for SCEP para conectarse AWS Private CA a sus clientes y dispositivos compatibles con SCEP.	11 de junio de 2024
Nueva guía de solución de problemas de conectores	Se han añadido nuevas secciones sobre la solución de problemas relacionados con los errores de creación de conectores y SPN.	4 de abril de 2024
Se agregó la extensión CDP para Matter	Añade compatibilidad con la extensión de punto de distribución de listas de revocación de certificados (CDP) para Matter.	25 de enero de 2024
AWS Private CA Soporte de API para mDL	Se agregó compatibilidad con la API para crear certificados que se ajusten a la norma ISO/IEC para permisos de conducir móviles (mDL).	16 de enero de 2024
AWS Private CA Conector para Active Directory	Compatibilidad con la guía del usuario, la API y la CLI para Conector para AD. Para obtener más información, consulte la documentación para el Conector para AD.	24 de agosto de 2023

Cambiar los nombres de las políticas de seguridad para que coincidan con el nombre del nuevo servicio	Adopción de nuevos nombres para las políticas de IAM AWS administradas que especific an los permisos estándar para. Autoridad de certificación privada de AWS Para obtener más información, consulte Políticas administradas de AWS.	13 de febrero de 2023
Añadir un rastreador de cambios para las políticas AWS gestionadas	Se ha añadido documentación para realizar un seguimiento de los cambios en las políticas de IAM AWS gestionadas que especifican los permisos estándar. Autoridad de certificación privada de AWS Para obtener más información, consulte Actualizaciones para las políticas administradas de AWS por Autoridad de certifica ción privada de AWS.	11 de noviembre de 2022
Soporte de API y CLI para CAs emitir certificados de corta duración	Con la introducción de los modos de uso de las CA, se puede configurar una CA para que emita certificados de uso general o exclusivamente certificados de corta duración. Para obtener más informaci ón, consulte los Modos de entidades de certificación.	24 de octubre de 2022

Cambio de marca del servicio y actualización de la consola	El nombre del servicio pasa a ser AWS Private Certificate Authority (Autoridad de certificación privada de AWS). La Autoridad de certificación privada de AWS consola presenta mejoras de usabilidad, incluidos paneles de ayuda integrados que enlazan con la documentación completa.	27 de septiembre de 2022
Soporte de certificados compatible con Matter	Tres nuevas plantillas de certificados admiten certificados de entidad final y de CA compatibles con Matter. Para obtener más información, consulte Comprender las plantillas de certificación.	20 de julio de 2022
Compatibilidad con nuevas regiones	Se ha añadido un punto de conexión para Asia-Pacífico (Yakarta). Para obtener una lista completa de los puntos de AWS Private CA conexión, consulte Puntos de conexión y cuotas de la Autoridad de Certificación Privada de ACM.	4 de mayo de 2022
Compatibilidad con atributos personalizados y extensiones	Utilice el CustomAttributeobjeto para configurar los certificados CAs y personalizados y el CustomExtension objeto para configurar los certificados personalizados.	16 de marzo de 2022

Compatible con OSCP administrado	Consulte Configurar un método de revocación de certificados para ver las opciones de revocación, incluido el OCSP.	18 de agosto de 2021
Support para la función S3 Block Public Access para CRLs	Consulte la Habilitación de la característica Bloqueo de acceso público en S3.	27 de mayo de 2021
Ejemplos de implementación de Java nuevos y actualizados	Consulte Uso de la API de CA privada de ACM (ejemplos en Java).	9 de septiembre de 2020
Compatibilidad con nuevas regiones	Se han añadido puntos de conexión para África (Ciudad del Cabo) y Europa (Milán). Para obtener una lista completa de puntos de conexión de Autoridad de certificación privada de AWS , consulte Cuotas y puntos de conexión de entidad de certificación privada de AWS Certificate Manager.	27 de agosto de 2020
Compatible con el acceso de CA privado entre cuentas	AWS Certificate Manager se puede autorizar a los usuarios a emitir certificados utilizando certificados privados CAs que no sean de su propiedad. Para obtener más información, consulte Acceso multicuenta a Private CAs.	17 de agosto de 2020

Compatibilidad con puntos finales de VPC () PrivateLink	Se agregó compatibilidad con el uso de puntos de conexión de VPC (AWS PrivateLink) para mejorar la seguridad de la red. Para obtener más información, consulte ACM Private CA VPC Endpoints AWS PrivateLink().	26 de marzo de 2020
Se agregó la sección Seguridad dedicada	La documentación de seguridad de se AWS ha consolidado en una sección dedicada a la seguridad. Para obtener información sobre la seguridad, consulte Seguridad en una autoridad de certificación AWS Certificate Manager privada.	26 de marzo de 2020
ARN de plantilla agregada a los informes de auditoría.	Para obtener más información, consulte Creación de un informe de auditoría para su entidad de certificación privada.	6 de marzo de 2020
CloudFormation soporte	Support agregado para CloudFormation. Para obtener más información, consulte Referencia del tipo de recurso ACMPCA en la guía del usuario de CloudFormation .	22 de enero de 2020

CloudWatch Integración de eventos	Integración con CloudWatch eventos para eventos asíncronos, incluida la creación de entidades de certificación, la emisión de certificados y la creación de CRL. Para obtener más información, consulte Uso de eventos. CloudWatch	23 de diciembre de 2019
Puntos de conexión FIPS	Se agregaron puntos finales FIPS para AWS GovCloud (EE. UU. Este) y AWS GovCloud (EE. UU. Oeste). Para obtener una lista completa de puntos de Autoridad de certificación privada de AWS enlace, consulte Puntos de enlace y cuotas de una autoridad de certificación AWS Certificate Manager privada.	13 de diciembre de 2019
Permisos basados en etiquetas	Los permisos basados en etiquetas se admiten con las nuevas APIs `TagResource` , y`UntagResource` . `ListTagsForResource` Para obtener información general acerca de los controles basados en etiquetas, consulte Control del acceso para usuarios y roles de IAM mediante etiquetas de recursos de IAM.	5 de noviembre de 2019

Aplicación de restricciones del nombre	Se ha añadido compatibilidad con la aplicación de restricciones en el nombre del asunto en los certificados de CA importados. Para obtener más información, consulte Imponer restricciones de nombres en una CA privada.	28 de octubre de 2019
Nuevas plantillas de certificado	Se han agregado nuevas plantillas de certificados, incluidas plantillas para firmar código. AWS Signer Para obtener más información, consulte Uso de plantillas.	1 de octubre de 2019
Planificación de su CA	Se ha agregado una nueva sección sobre la planificación de su PKI mediante Autoridad de certificación privada de AWS. Para obtener más información, consulte Planificación de la implementación de entidad privada de ACM.	30 de septiembre de 2019
Se agregó compatibilidad con regiones	Se agregó soporte regional para la región de AWS Asia Pacífico (Hong Kong). Para obtener una lista completa de las regiones admitidas , consulte los puntos de conexión y cuotas de la entidad de certificación privada AWS Certificate Manager.	24 de julio de 2019

Se agregó una compatibilidad completa para jerarquías de CA privadas	Support for create and hosting root CAs elimina la necesidad de un padre externo.	20 de junio de 2019
Se agregó compatibilidad con regiones	Se agregó soporte regional para las regiones AWS GovCloud (EE. UU. oeste y EE. UU. Este). Para obtener una lista completa de las regiones admitidas, consulte Puntos de conexión y cuotas de la entidad de certificación privada del gestor de certificaciones de AWS.	8 de mayo de 2019
Se agregó compatibilidad con regiones	Se agregó el soporte regional para las regiones de AWS Asia Pacífico (Bombay y Seúl), EE.UU. Oeste (Norte de California) y UE (París y Estocolmo). Para obtener una lista completa de las regiones admitidas, consulte los puntos de conexión y cuotas de la entidad de certificación privada AWS Certificate Manager.	4 de abril de 2019
Prueba de flujo de trabajo de renovación de certificados	Los clientes ahora pueden probar manualmente la configuración de su flujo de trabajo de renovación administrado de ACM. Para obtener más información, consulte la sección Prueba de la configuración de renovación administrada de ACM.	14 de marzo de 2019

Se agregó compatibilidad con regiones	Se agregó soporte regional para la región de AWS la UE (Londres). Para obtener una lista completa de las regiones admitidas, consulte Puntos de conexión y cuotas de la entidad de certificación privada del gestor de certificaciones de AWS.	1 de agosto de 2018
Restaurar eliminado CAs	La restauración privada de CA permite a los clientes restaurar las autoridades de certificación (CAs) durante un máximo de 30 días después de su eliminación. Para obtener más información, consulte Restauración de una entidad de certificación privada.	20 de junio de 2018

Actualizaciones anteriores

En la siguiente tabla se describe el historial de publicación de la documentación AWS Private Certificate Authority anterior a junio de 2018.

Cambio	Descripción	Fecha
Nueva guía	Esta versión introduce AWS Private Certificate Authority.	04 de abril de 2018

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.