aws

Guía del usuario

AWS Private Certificate Authority

Version latest

AWS Private Certificate Authority: Guía del usuario

Table of Contents

¿Qué es Autoridad de certificación privada de AWS?

Autoridad de certificación privada de AWS permite la creación de jerarquías de entidades de certificación (CA) privadas, incluidas las entidades de certificación raíz y subordinadas, sin los costos de inversión y mantenimiento de operar una entidad de certificación en las instalaciones. Sus entidades de certificación privadas pueden emitir certificados X.509 de entidad final útiles en situaciones tales como:

- 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

Se puede acceder a las operaciones de Autoridad de certificación privada de AWS desde la AWS Management Console, usando la API de Autoridad de certificación privada de AWS o mediante la AWS CLI.

Temas

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

Hay dos servicios de AWS 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 desea implementar en otros servicios de AWS o administración y renovación automatizadas de 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 Autoridad de certificación privada de AWS, puede crear su propia jerarquía de entidad de certificación (CA) y emitir certificados con ella para autenticar usuarios internos, equipos, aplicaciones, servicios, servidores y otros dispositivos y para firmar código informático. 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, inicia sesión en la consola https://console.aws.amazon.com/acm-pca/.
2. AWS Certificate Manager (ACM): este servicio administra certificados para clientes empresariales que necesitan una presencia web públicamente segura 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 utiliza Autoridad de certificación privada de AWS para crear una CA, ACM puede administrar la emisión de certificados desde esa entidad de certificación privada y automatizar las renovaciones de certificados.

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

Regiones

Al igual que la mayoría de los recursos de AWS, las entidades de certificación (CA) privadas son recursos regionales. Para poder utilizar entidades de certificación privadas en varias regiones, debe

crearlas en esas regiones. Las entidades de certificación privadas no se pueden copiar de una región a otra. 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**
>
> En estos momentos, ACM está disponible en algunas regiones en las que Autoridad de certificación privada de AWS no lo está.

Servicios integrados con AWS Private Certificate Authority

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

También puede integrar las CA privadas en Amazon Elastic Kubernetes Service para emitir certificados dentro de un clúster de Kubernetes. Para obtener más información, consulte Protección de Kubernetes con Autoridad de certificación privada de AWS.

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

Si utiliza la API de Autoridad de certificación privada de AWS o AWS CLI para emitir un certificado o para exportar un certificado privado desde ACM, podrá instalar el certificado donde quiera.

Algoritmos criptográficos compatibles

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
RSA_2048	SHA256WITHECDSA
RSA_4096	SHA384WITHECDSA
EC_prime256v1	SHA512WITHECDSA
EC_secp384r1	SHA256WITHRSA SHA384WITHRSA
	SHA512WITHRSA

Esta lista se aplica únicamente a los certificados emitidos directamente por 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 Solicitud de un certificado privado en la Guía del usuario de AWS Certificate Manager.

> **Note**
>
> La familia de algoritmos de firma especificada (RSA o ECDSA) debe coincidir con la familia de algoritmos de la clave privada de la entidad de certificación.

Cuotas

Autoridad de certificación privada de AWS asigna cuotas al número permitido de certificados y entidades de certificación. Las tasas de solicitud de acciones de la API también están sujetas a cuotas. Las cuotas de Autoridad de certificación privada de AWS son específicas de una cuenta de AWS y una región.

Autoridad de certificación privada de AWS limita las solicitudes de API a distintas tasas en función de la operación de la API. La limitación controlada significa que Autoridad de certificación privada de AWS rechaza una solicitud válida porque supera la cuota de la operación del número de solicitudes por segundo. Cuando se limita una solicitud, Autoridad de certificación privada de AWS devuelve un

error. ThrottlingException Autoridad de certificación privada de AWSno garantiza una tasa mínima de solicitudes para las API.

Para ver qué cuotas se pueden ajustar, consulta 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 la cuenta maestra de AWS.
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.

Para 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 en AWS Certificate Manager. Puede utilizar la consola de ACM, la AWS CLI o la API de ACM para solicitar certificados privados a una entidad de certificación (CA) privada existente. 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 Quotas en la Guía del usuario de AWS Certificate Manager.

Conformidad de RFC

Autoridad de certificación privada de AWS no impone determinadas restricciones definidas en 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 aplica restricciones básicas y longitud de ruta en los certificados de entidad de certificación 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 marca `critical`, debe ser establecido `CA=true`. Autoridad de certificación privada de AWS aplica restricciones básicas al devolver un error 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 ruta (pathLenConstraint) determina cuántas CA subordinadas pueden existir aguas abajo del certificado de CA importado. Autoridad de certificación privada de AWSimpone 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.

No se aplica

- Restricciones de política. Estas restricciones limitan la capacidad de una entidad de certificación para emitir certificados de entidad de certificación subordinada.
- 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 entidad de certificación deben contener una extensión AKI que coincida con el SKI del certificado de entidad de certificación. 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 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

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 cobro incluye los certificados exportados desde Autoridad de certificación privada de AWS y los certificados creados a través de la API de o la CLI de Autoridad de certificación privada de AWS. 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 información acerca de los precios de Autoridad de certificación privada de AWS, consulte los Precios de AWS Private Certificate Authority. También puede utilizar la calculadora de precios de AWS para estimar los costos.

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. Usted también es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos 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

El acceso a Autoridad de certificación privada de AWS requiere credenciales que AWS puede utilizar para autenticar sus solicitudes. Los siguientes temas contienen información detallada sobre cómo utilizar AWS Identity and Access Management (IAM) para ayudar a proteger las entidades de certificación (CA) privadas al controlar quién puede obtener acceso 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 una CA.
  > - 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 administrados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones de Creación de un rol para un proveedor de identidades 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.

AWSPrivateCAFullAccess

Otorga un control administrativo sin restricciones.

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

AWSPrivateCAReadOnly

Otorga acceso limitado a las operaciones de la API de solo lectura.

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":[
         "acm-pca:DescribeCertificateAuthority",
         "acm-pca:DescribeCertificateAuthorityAuditReport",
         "acm-pca:ListCertificateAuthorities",
         "acm-pca:GetCertificateAuthorityCsr",
         "acm-pca:GetCertificateAuthorityCertificate",
```

```
         "acm-pca:GetCertificate",
         "acm-pca:GetPolicy",
         "acm-pca:ListPermissions",
         "acm-pca:ListTags"
      ],
      "Resource":"*"
   }
}
```

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.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:IssueCertificate"
         ],
         "Resource":"arn:aws:acm-pca:*:*:certificate-authority/*",
         "Condition":{
            "StringLike":{
               "acm-pca:TemplateArn":[
                  "arn:aws:acm-pca:::template/*CACertificate*/V*"
               ]
            }
         }
      },
      {
         "Effect":"Deny",
         "Action":[
            "acm-pca:IssueCertificate"
         ],
         "Resource":"arn:aws:acm-pca:*:*:certificate-authority/*",
         "Condition":{
            "StringNotLike":{
               "acm-pca:TemplateArn":[
                  "arn:aws:acm-pca:::template/*CACertificate*/V*"
               ]
```

```
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:RevokeCertificate",
            "acm-pca:GetCertificate",
            "acm-pca:ListPermissions"
         ],
         "Resource":"arn:aws:acm-pca:*:*:certificate-authority/*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:ListCertificateAuthorities"
         ],
         "Resource":"*"
      }
   ]
}
```

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.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:IssueCertificate"
         ],
         "Resource":"arn:aws:acm-pca:*:*:certificate-authority/*",
         "Condition":{
            "StringLike":{
               "acm-pca:TemplateArn":[
                  "arn:aws:acm-pca:::template/EndEntityCertificate/V*"
               ]
            }
```

```
         }
      },
      {
         "Effect":"Deny",
         "Action":[
            "acm-pca:IssueCertificate"
         ],
         "Resource":"arn:aws:acm-pca:*:*:certificate-authority/*",
         "Condition":{
            "StringNotLike":{
               "acm-pca:TemplateArn":[
                  "arn:aws:acm-pca:::template/EndEntityCertificate/V*"
               ]
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:RevokeCertificate",
            "acm-pca:GetCertificate",
            "acm-pca:ListPermissions"
         ],
         "Resource":"arn:aws:acm-pca:*:*:certificate-authority/*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:ListCertificateAuthorities"
         ],
         "Resource":"*"
      }
   ]
}
```

AWSPrivateCAAuditor

Conceda acceso a las operaciones de la API de solo lectura y permiso para generar un informe de auditoría de CA.

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

```
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:CreateCertificateAuthorityAuditReport",
            "acm-pca:DescribeCertificateAuthority",
            "acm-pca:DescribeCertificateAuthorityAuditReport",
            "acm-pca:GetCertificateAuthorityCsr",
            "acm-pca:GetCertificateAuthorityCertificate",
            "acm-pca:GetCertificate",
            "acm-pca:GetPolicy",
            "acm-pca:ListPermissions",
            "acm-pca:ListTags"
         ],
         "Resource":"arn:aws:acm-pca:*:*:certificate-authority/*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:ListCertificateAuthorities"
         ],
         "Resource":"*"
      }
   ]
}
```

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 Autoridad de certificación privada de AWS, suscríbase a la fuente RSS de la Historial de revisiones del documento página.

Cambios en las políticas administradas

Cambio	Descripción	Fecha
Nuevos nombres de las políticas: • `AWSPrivateCAFullAccess`	Los prefijos de los nombres de las políticas se cambiaron de `AWSCertificateMana`	13 de febrero de 2023

Cambio	Descripción	Fecha
• `AWSPrivateCAReadOnly` • `AWSPrivateCAPrivilegedUser` • `AWSPrivateCAAuditor` • `AWSPrivateCAUser`	`gerPrivateCA` a `AWSPrivateCA` . 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 únicamente las acciones que usted especifique. Autoridad de certificación privada de AWS

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 Autoridad de certificación privada de AWS las operaciones 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.

```
{
```

```
   "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. Seleccione 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 AWS Management Console, 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

- Mostrar las CA privadas
- 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
- Acceso de administrador a todos los recursos de AWS

Mostrar las CA privadas

La siguiente política permite a un usuario mostrar todas las CA privadas de una cuenta.

```
{
   "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.

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:GetCertificateAuthorityCertificate",
      "Resource":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   }
}
```

Importar el certificado de una CA privada

La siguiente política permite al usuario importar el certificado de una CA privada.

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:ImportCertificateAuthorityCertificate",
      "Resource":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   }
}
```

Eliminar una CA privada

La siguiente política permite al usuario eliminar el certificado de una CA privada.

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:DeleteCertificateAuthority",
      "Resource":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   }
}
```

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.

```
{
   "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.

```
{
   "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 a las CA con el par clave-valor Environment=. PreProd También requiere que las nuevas CA incluyan esta etiqueta.

```
{
   "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.

```
{
   "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.

```
{
```

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

Acceso de administrador a todos los recursos de AWS

La siguiente política permite a un usuario realizar cualquier acción en cualquier AWS recurso.

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

Prácticas recomendadas de seguridad para el acceso entre cuentas a CA privadas

Un Autoridad de certificación privada de AWS administrador puede compartir una CA con los responsables (usuarios, funciones, etc.) de otra AWS cuenta. 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 al respecto AWS Organizations, consulte la Guía AWS Organizations del usuario.
- 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 sección de Controlar el acceso a una CA privada de esta guía proporciona flujos de trabajo para conceder acceso a las CA tanto en escenarios de una sola cuenta como de cuentas múltiples.

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 administradas 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 las API PutPolicyde la PCA o los AWS CLI comandos correspondientes put-policy DeletePolicy, get-policy y delete-policy para aplicar y administrar políticas basadas en recursos. GetPolicy

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 AWS gestionadas basadas en recursos 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_API Passthrough/v1
- BlankEndEntityCertificate_API SRPassthrough/v1
- Certificado CA subordinado_ 0/V1 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 pueden configurar los campos de ID de los principales (el número de AWS cuenta o el ID de la AWS organización) y los ARN de CA.

1. Archivo: policy1.json — Compartir el acceso a una CA con un usuario de una cuenta diferente

   Sustituya *555555555555* por el ID de AWS cuenta que 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.

```
{
  "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-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
```

```
      },
      {
         "Sid":"ExampleStatementID2",
         "Effect":"Allow",
         "Principal":{
            "AWS":"555555555555"
         },
         "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"
            }
         }
      }
   ]
}
```

2. Archivo: policy2.json — Compartir el acceso a una CA a través de AWS Organizations

   Sustituya *o-a1b2c3d4z5* 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.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ExampleStatementID3",
         "Effect":"Allow",
         "Principal":"*",
         "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",
               "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-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
         "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. Usted es responsable de mantener

el control sobre el contenido alojado en esta infraestructura. Usted también es 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:

- Utilice autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos. AWS Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con. AWS CloudTrail
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utilice servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-2 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) 140-2.

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 Autoridad de certificación privada de AWS o Servicios de AWS utiliza la consola, la API o los SDK. AWS CLI AWS Cualquier dato que ingrese en etiquetas o campos de formato libre utilizados para nombres se puede 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 Autoridad de certificación privada de AWS claves privadas

Las claves privadas de las CA privadas se almacenan en módulos de seguridad de hardware (HSM) AWS gestionados. Los HSM cumplen con los requisitos de seguridad FIPS PUB 140-2 de nivel 3 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 en 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:

- Para las organizaciones que deben cifrar sus buckets de Amazon S3, en los siguientes temas se describe cómo configurar el cifrado para dar cabida Autoridad de certificación privada de AWS a los activos:
  - Cifrado de los informes de auditoría

  - Cifrado de las CRL
- Guías de inicio rápido sobre sobre seguridad y conformidad: estas guías de implementación analizan las consideraciones arquitectónicas y proporcionan los pasos para implementar entornos básicos centrados en la seguridad y el cumplimiento. AWS
- Documento técnico sobre cómo diseñar una arquitectura basada en la seguridad y el cumplimiento de la HIPAA: este documento técnico 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 recursos con reglas en la Guía para desarrolladores de AWS Config : el servicio AWS Config evalúa en qué medida las configuraciones de sus recursos cumplen las prácticas internas, las directrices del sector y las normativas.
- AWS Security Hub— Este AWS servicio proporciona una visión integral del estado de su seguridad AWS que le ayuda a comprobar el cumplimiento de los estándares y las mejores prácticas del sector de la seguridad.

Uso de 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 `bucket-name`. `CA_ID` es el identificador único de una CA emisora. `UUID` es el identificador único de un informe de auditoría.

```
bucket-name/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.

Preparación de un bucket de Amazon S3 para informes de auditoría

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?

Su bucket de S3 debe estar protegido por una política de permisos asociada. Los usuarios autorizados y los directores del 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 cuenta de AWS específica como al ARN de una CA privada. Para obtener más información, consulte Agregar una política de bucket mediante la consola de Amazon S3.

> **Note**
>
> Durante el procedimiento de consola para crear un informe de auditoría, puede optar por dejar que se Autoridad de certificación privada de AWS cree un nuevo depósito y aplicar una política de permisos predeterminada. La política predeterminada no aplica ninguna restricción de `SourceArn` a la CA y, por lo tanto, es más permisiva que la política recomendada. Si decide elegir la predeterminada, siempre podrá modificarla más adelante.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"acm-pca.amazonaws.com"
         },
         "Action":[
            "s3:PutObject",
```

```
            "s3:PutObjectAcl",
            "s3:GetBucketAcl",
            "s3:GetBucketLocation"
         ],
         "Resource":[
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
         ],
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"account",
               "aws:SourceArn":"arn:partition:acm-pca:region:account:certificate-
authority/CA_ID"
            }
         }
      }
   ]
}
```

Creación de 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. Inicie sesión en su AWS cuenta y abra la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/home.
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. 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 Preparación de un bucket de Amazon S3 para informes de auditoría. Para obtener más información sobre cómo adjuntar una política, consulte Agregar una política de bucket mediante la consola de Amazon S3

5. En Formato de salida, selecciona JSON para la notación de JavaScript objetos o CSV para los 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 Preparación de un bucket de Amazon S3 para 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 bucket_name \
--audit-report-response-format JSON
```

Recuperar 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. Seleccione 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. Seleccione 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.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puede utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

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 de conexión de la VPC de la interfaz 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. AWS Direct Connect Las instancias de tu VPC no necesitan direcciones IP públicas para comunicarse con la Autoridad de certificación privada de AWS API.

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 (AWS VPN) o. AWS Direct Connect Para obtener más información AWS VPN, consulte Conexiones VPN en la Guía del usuario de Amazon VPC. Para obtener más información AWS Direct Connect, consulte Creación de una conexión en la Guía del AWS Direct Connect usuario.

Autoridad de certificación privada de AWS no requiere el uso de AWS PrivateLink, pero se recomienda 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.
- AWS Certificate Manager no admite puntos finales de VPC.
- Los puntos de conexión de FIPS (y sus regiones) no admiten puntos de enlace 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

- US East (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 (Bombay)
- Asia-Pacífico (Osaka)
- Asia-Pacífico (Seúl)
- Asia-Pacífico (Singapur)
- Asia-Pacífico (Sídney)
- Asia-Pacífico (Tokio)
- Canadá (centro)
- Europa (Fráncfort)
- Europa (Irlanda)
- Europa (Londres)
- Europa (París)
- Europa (Estocolmo)
- Europa (Milán)
- Israel (Tel Aviv)
- Medio Oriente (Baréin)

- América del Sur (São Paulo)

Creación de puntos de conexión 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 https://console.aws.amazon.com/vpc/ o el. 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**
>
> La *región* de valores representa el identificador de AWS región de una región compatible Autoridad de certificación privada de AWS, como `us-east-2` la región EE.UU. Este (Ohio). Para obtener una lista Autoridad de certificación privada de AWS, consulte AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.

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.

Creación de 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 Controlar el acceso a servicios con puntos de conexión de VPC en la Guía del usuario 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 acciones`IssueCertificate`,,`DescribeCertificateAuthority`, `GetCertificate` `GetCertificateAuthorityCertificate`, `ListPermissions`, 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"
            ]
         }
      ]
   }
```

Registrar y monitorear para AWS Private Certificate Authority

El monitoreo es una parte importante del mantenimiento de la confiabilidad, la disponibilidad y el rendimiento de AWS Private Certificate Authority sus AWS soluciones. Debe recopilar los datos de supervisión de todas las partes de la AWS solución para poder depurar más fácilmente un error multipunto en caso de que se produzca.

En los siguientes temas se describen las herramientas de AWS supervisión de la nube disponibles para su uso con. Autoridad de certificación privada de AWS

Temas

- Métricas compatibles CloudWatch
- Uso de CloudWatch eventos
- Usando CloudTrail

Métricas compatibles CloudWatch

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	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

Uso de 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 los ARN de la entidad de certificación y del 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 los ARN de la entidad de certificación y del 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"
   }
}
```

Usando CloudTrail

Se puede utilizar AWS CloudTrailpara grabar las llamadas a la API realizadas por AWS Private Certificate Authority. Para obtener más información, consulte los siguientes temas.

Temas

- Creación de una política
- Recuperar una política
- Eliminación de una política
- Creación de una entidad de certificación
- GenerateCRL
- GenerateOCSPResponse
- Creación de un informe de auditoría
- Eliminación de una entidad de certificación
- Restauración de una entidad de certificación

- Descripción de una entidad de certificación
- Recuperar el certificado de una entidad de certificación
- Recuperar la solicitud de firma de una entidad de certificación
- Recuperación de un certificado
- Importación del certificado de una entidad de certificación
- Emitir un certificado
- Enumerar las entidades de certificación
- Enumeración de etiquetas
- Revocación de un certificado
- Uso de etiquetas en las entidades de certificación privadas
- Eliminación de etiquetas en una entidad de certificación privada
- Actualización de una entidad de certificación

Creación de una política

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la PutPolicyoperación.


```
{
   "eventVersion":"1.08",
   "userIdentity":{
      },
      "invokedBy":"agent"
   },
   "eventTime":"2021-02-26T21:25:36Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"PutPolicy",
   "awsRegion":"region",
   "sourceIPAddress":"xx.xx.xx.xx",
   "userAgent":"agent",
   "requestParameters":{
      "resourceArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "policy":"{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"01234567-89ab-cdef-0123-456789abcdef4-external-principals\",\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":\"account\"},\"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\"}}},{\"Sid\":\"01234567-89ab-cdef-0123-456789abcdef-external-principals
\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"account\"},\"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\"}]}"
   },
   "responseElements":null,
   "requestID":"01234567-89ab-cdef-0123-456789abcdef",
   "eventID":"01234567-89ab-cdef-0123-456789abcdef",
   "readOnly":false,
   "eventType":"AwsApiCall",
   "managementEvent":true,
   "eventCategory":"Management",
   "recipientAccountId":"account"
}
```


Recuperar una política

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la GetPolicyoperación.


```
{
   "eventVersion":"1.08",
   "userIdentity":{
      "type":"AssumedRole",
      "principalId":"account",
      "arn":"arn:aws:sts::account:assumed-role/role",
      "accountId":"account",
      "accessKeyId":"key_ID",
      "sessionContext":{
         "sessionIssuer":{
            "type":"Role",
            "principalId":"account",
            "arn":"arn:aws:iam::account:role/role",
            "accountId":"account",
            "userName":"name"
         },
         "webIdFederationData":{

         },
         "attributes":{
            "mfaAuthenticated":"false",
            "creationDate":"2021-02-26T20:49:51Z"
```

```
         }
      }
   },
   "eventTime":"2021-02-26T21:19:14Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"GetPolicy",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "errorCode":"ResourceNotFoundException",
   "errorMessage":"Could not find policy for resource arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566.",
   "requestParameters":{
      "resourceArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "readOnly":true,
   "eventType":"AwsApiCall",
   "managementEvent":true,
   "eventCategory":"Management",
   "recipientAccountId":"account"
}
```

Eliminación de una política

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la DeletePolicyoperación.

```
{
   "eventVersion":"1.08",
   "userIdentity":{
      "type":"AssumedRole",
      "principalId":"account",
      "arn":"arn:aws:sts::account:assumed-role/role",
      "accountId":"account",
      "accessKeyId":"key_ID",
      "sessionContext":{
         "sessionIssuer":{
            "type":"Role",
            "principalId":"account",
            "arn":"arn:aws:iam::account:role/role",
```

```
            "accountId":"account",
            "userName":"name"
         },
         "webIdFederationData":{

         },
         "attributes":{
            "mfaAuthenticated":"false",
            "creationDate":"2021-02-26T21:23:17Z"
         }
      }
   },
   "eventTime":"2021-02-26T21:23:31Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"DeletePolicy",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "resourceArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "readOnly":false,
   "eventType":"AwsApiCall",
   "managementEvent":true,
   "eventCategory":"Management",
   "recipientAccountId":"account"
}
```

Creación de una entidad de certificación

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la CreateCertificateAuthorityoperación.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
```

```
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T21:22:33Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"CreateCertificateAuthority",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityConfiguration":{
         "keyType":"RSA2048",
         "signingAlgorithm":"SHA256WITHRSA",
         "subject":{
            "country":"US",
            "organization":"Example Company",
            "organizationalUnit":"Corp",
            "state":"WA",
            "commonName":"www.example.com",
            "locality":"Seattle"
         }
      },
      "revocationConfiguration":{
         "crlConfiguration":{
            "enabled":true,
            "expirationInDays":3650,
            "customCname":"your-custom-name",
            "s3BucketName":"your-bucket-name"
         }
      },
      "certificateAuthorityType":"SUBORDINATE",
      "idempotencyToken":"98256344"
   },
   "responseElements":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   },
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

GenerateCRL

En el siguiente CloudTrail ejemplo, se muestra el registro de un evento GenerateCRL.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "account",
        "invokedBy": "acm-pca.amazonaws.com"
    },
    "eventTime": "2021-02-09T17:37:45Z",
    "eventSource": "acm-pca.amazonaws.com",
    "eventName": "GenerateCRL",
    "awsRegion": "region",
    "sourceIPAddress": "acm-pca.amazonaws.com",
    "userAgent": "acm-pca.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "01234567-89ab-cdef-0123-456789abcdef",
    "readOnly": false,
    "resources": [
        {
            "type": "AWS::ACMPCA::CertificateAuthority",
            "ARN": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
        }
    ],
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "account"
}
```

GenerateOCSPResponse

En el siguiente CloudTrail ejemplo, se muestra el registro de un evento GenerateOCSPResponse.

```
{
   "eventVersion":"1.08",
   "userIdentity":{
      "accountId":"account",
      "invokedBy":"acm-pca.amazonaws.com"
   },
```

```
   "eventTime":"2021-02-08T23:52:29Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"GenerateOCSPResponse",
   "awsRegion":"region",
   "sourceIPAddress":"acm-pca.amazonaws.com",
   "userAgent":"acm-pca.amazonaws.com",
   "eventID":"01234567-89ab-cdef-0123-456789abcdef",
   "readOnly":false,
   "resources":[
      {
         "type":"AWS::ACMPCA::Certificate",
         "ARN":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
      }
   ]
}
```

Creación de un informe de auditoría

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la CreateCertificateAuthorityAuditReportoperación.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T21:56:00Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"CreateCertificateAuthorityAuditReport",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "s3BucketName":"bucket_name",
      "auditReportResponseFormat":"JSON"
   },
```

```
   "responseElements":{
      "auditReportId":"report_ID",
      "s3Key":"audit-report/CA_ID/audit_report_ID.json"
   },
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Eliminación de una entidad de certificación

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la DeleteCertificateAuthorityoperación. En este ejemplo, la entidad de certificación no se puede eliminar, ya que no tiene el estado `ACTIVE`.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T22:01:11Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"DeleteCertificateAuthority",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "errorCode":"InvalidStateException",
   "errorMessage":"The certificate authority is not in a valid state for deletion.",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
```

```
}
```

Restauración de una entidad de certificación

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la RestoreCertificateAuthorityoperación. En este ejemplo, la entidad de certificación no se puede restaurar, ya que no tiene el estado `DELETED`.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T22:01:11Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"RestoreCertificateAuthority",
   "awsRegion":"region",
   "sourceIPAddress":"xIP_address",
   "userAgent":"agent",
   "errorCode":"InvalidStateException",
   "errorMessage":"The certificate authority is not in a valid state for restoration.",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Descripción de una entidad de certificación

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la DescribeCertificateAuthorityoperación.

```
{
```

```
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T21:58:18Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"DescribeCertificateAuthority",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```


Recuperar el certificado de una entidad de certificación

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la GetCertificateAuthorityCertificateoperación.


```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T22:03:52Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"GetCertificateAuthorityCertificate",
   "awsRegion":"region",
```

```
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Recuperar la solicitud de firma de una entidad de certificación

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la GetCertificateAuthorityCsroperación.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T21:40:33Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"GetCertificateAuthorityCsr",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Recuperación de un certificado

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la GetCertificateoperación.


```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T22:22:54Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"GetCertificate",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```


Importación del certificado de una entidad de certificación

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la ImportCertificateAuthorityCertificateoperación.


```
{
   "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"
}
```

Emitir un certificado

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la IssueCertificateoperación.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T22:18:43Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"IssueCertificate",
   "awsRegion":"region",
   "sourceIPAddress":"xIP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "csr":{
         "hb":[
            45,
            45,
            ...10
         ],
         "offset":0,
         "isReadOnly":false,
         "bigEndian":true,
         "nativeByteOrder":false,
         "mark":-1,
         "position":1090,
         "limit":1090,
```

```
         "capacity":1090,
         "address":0
      },
      "signingAlgorithm":"SHA256WITHRSA",
      "validity":{
         "value":365,
         "type":"DAYS"
      },
      "idempotencyToken":"1234"
   },
   "responseElements":{
      "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
   },
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Enumerar las entidades de certificación

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la ListCertificateAuthoritiesoperación.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T22:09:43Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"ListCertificateAuthorities",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "maxResults":10
   },
```

```
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Enumeración de etiquetas

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la ListTagsoperación.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-02-02T00:21:56Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"ListTags",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   },
   "responseElements":{
      "tags":[
         {
            "key":"Admin",
            "value":"Alice"
         },
         {
            "key":"User",
            "value":"Bob"
         }
      ]
   },
   "requestID":"request_ID",
```

```
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Revocación de un certificado

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la RevokeCertificateoperación.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accessKeyId": "key_ID"
   },
   "eventTime": "2018-01-26T22:35:03Z",
   "eventSource": "acm-pca.amazonaws.com",
   "eventName": "RevokeCertificate",
   "awsRegion": "region",
   "sourceIPAddress": "IP_address",
   "userAgent": "agent",
   "requestParameters": {
      "certificateAuthorityArn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "certificateSerial": "67:07:44:76:83:a9:b7:f4:05:56:27:ff:d5:5c:eb:cc",
      "revocationReason": "KEY_COMPROMISE"
   },
   "responseElements": null,
   "requestID": "request_ID",
   "eventID": "event_ID",
   "eventType": "AwsApiCall",
   "recipientAccountId": "account"
}
```

Uso de etiquetas en las entidades de certificación privadas

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la TagCertificateAuthorityoperación.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-02-02T00:18:48Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"TagCertificateAuthority",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "tags":[
         {
            "key":"Admin",
            "value":"Alice"
         }
      ]
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Eliminación de etiquetas en una entidad de certificación privada

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la UntagCertificateAuthorityoperación.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
```

```
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-02-02T00:21:50Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"UntagCertificateAuthority",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "tags":[
         {
            "key":"Admin",
            "value":"Alice"
         }
      ]
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Actualización de una entidad de certificación

El siguiente CloudTrail ejemplo muestra los resultados de una llamada a la UpdateCertificateAuthorityoperación.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T22:08:59Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"UpdateCertificateAuthority",
```

```
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",

      "revocationConfiguration":{
         "crlConfiguration":{
            "enabled":true,
            "expirationInDays":3650,
            "customCname":"your-custom-name",
            "s3BucketName":"your-bucket-name"
         }
      },
      "status":"DISABLED"
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Planificación de la Autoridad de certificación privada de AWS implementación

Autoridad de certificación privada de AWS le proporciona un control total y basado en la nube sobre la PKI (infraestructura de clave pública) privada de su organización, desde una autoridad de certificación (CA) raíz, pasando por las CA subordinadas, 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

Configurar su AWS cuenta y el AWS CLI

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.

> **Note**
>
> Autoridad de certificación privada de AWS no está disponible en la capa AWS gratuita.

Temas

- Inscríbase en una Cuenta de AWS
- Cómo crear un usuario 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 suscribirse a una Cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en una Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea una. 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 administrativo y utilice únicamente el usuario raíz para ejecutar tareas que requieren acceso de usuario raíz.

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. Puede ver la actividad de la cuenta y administrar la cuenta en cualquier momento entrando en https://aws.amazon.com/ y seleccionando Mi cuenta.

Cómo crear un usuario administrativo

Después de registrarte en un usuario Cuenta de AWS, protege Usuario raíz de la cuenta de AWS AWS IAM Identity Center, habilita y crea un usuario administrativo para que no utilices el usuario root en las tareas diarias.

Proteja su Usuario raíz de la cuenta de AWS

1. Inicie sesión AWS Management Consolecomo 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.

Crear un usuario administrativo

1. Activar IAM Identity Center

   Consulte las instrucciones en Enabling AWS IAM Identity Center en la Guía del usuario de AWS IAM Identity Center .
2. En el Centro de identidades de IAM, conceda acceso administrativo a un usuario administrativo.

   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 administrativo

- Para iniciar sesión con el usuario de IAM Identity Center, utilice 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.

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.

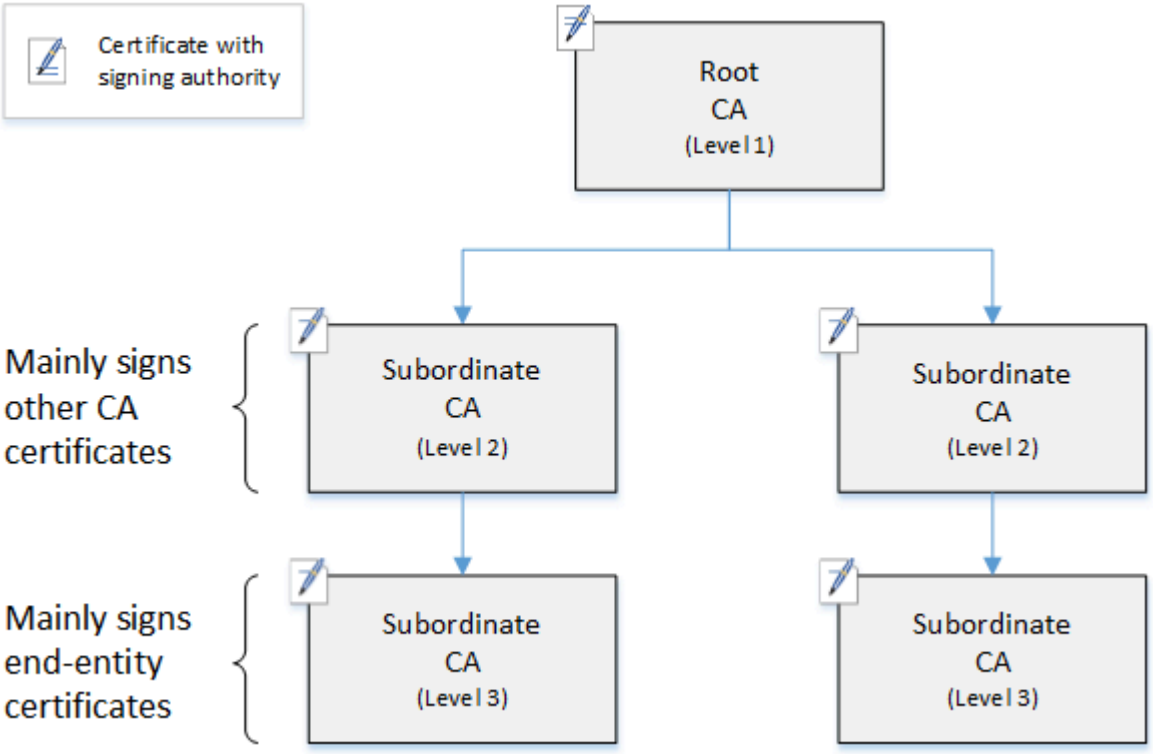
Diseño de una jerarquía de entidad de certificación

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 entidad de certificación raíz puede tener hasta cuatro niveles de entidades de certificación subordinadas 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
- Uso de entidades de certificación con confianza limitada y revocable para 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 pen-and-paper icono). Esto significa que, como entidades de certificación, pueden firmar otros certificados subordinados a ellas. 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. Estas entidades de certificación, a su vez, firman certificados para las entidades de certificación del nivel 3 que utilizan los administradores de la PKI (infraestructura de clave pública) que administran certificados de entidad final.

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 de certificación raíz ancla la confianza para todas las entidades de certificación subordinadas y los certificados de entidad final 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. Las entidades de certificación subordinadas de nivel raíz y superior se utilizan con poca frecuencia (normalmente para firmar otros certificados de entidad de certificación). 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)
>
> Las entidades de certificación raíz y otras entidades de certificación de alto nivel requieren procesos operativos altamente seguros y protocolos de control de acceso.

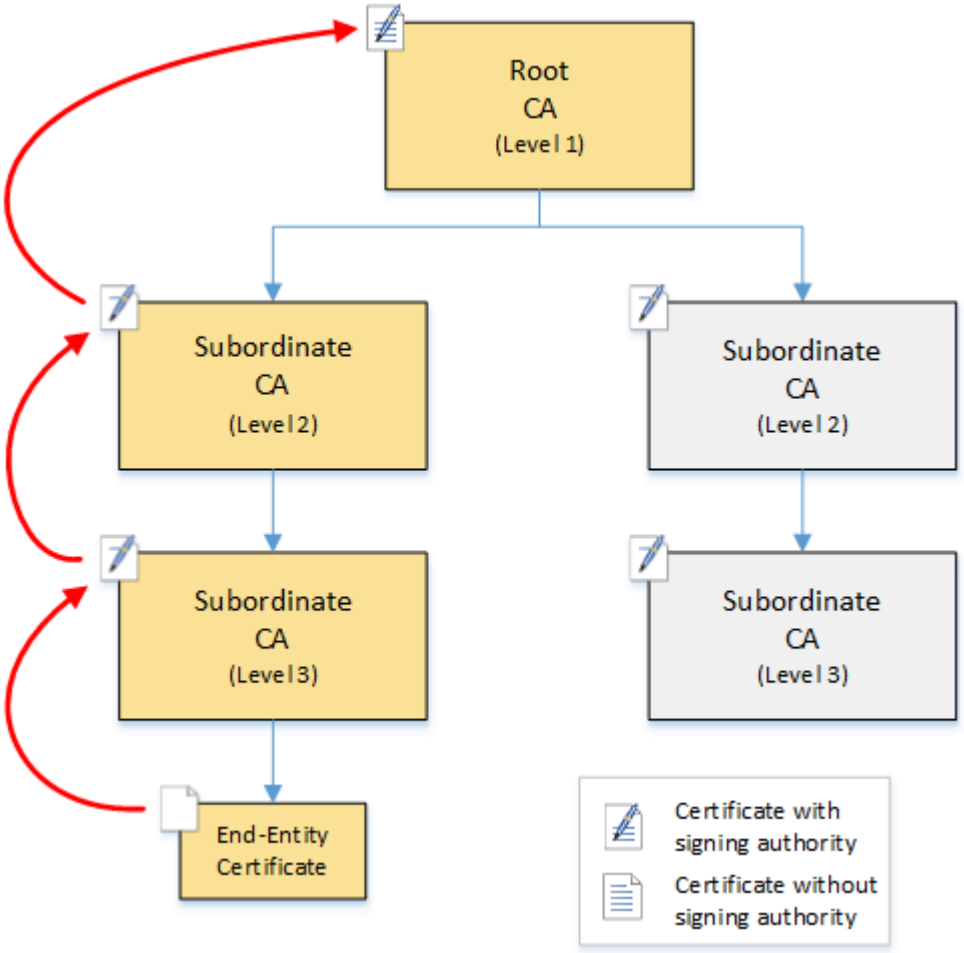
Temas

Validación de certificados de entidad final

Los certificados de entidad final obtienen su confianza de una ruta de certificación que conduce de vuelta a través de las entidades de certificación subordinadas a una entidad de certificación 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 entidades de certificación de confianza 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 de las entidades de certificación subordinadas 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.

Planificación de la estructura de una jerarquía de entidad de certificación

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 CA subordinadas. Una CA subordinada puede tener tantas CA subordinadas secundarias como permita su configuración interna. Autoridad de certificación privada de AWS Las jerarquías administradas 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 las entidades de certificación subordinadas
- Una estructura jerárquica que protege las entidades de certificación de nivel superior 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 entidad de certificación media sólo se utiliza para firmar entidades de certificación subordinadas que llevan a cabo la emisión de certificados de 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

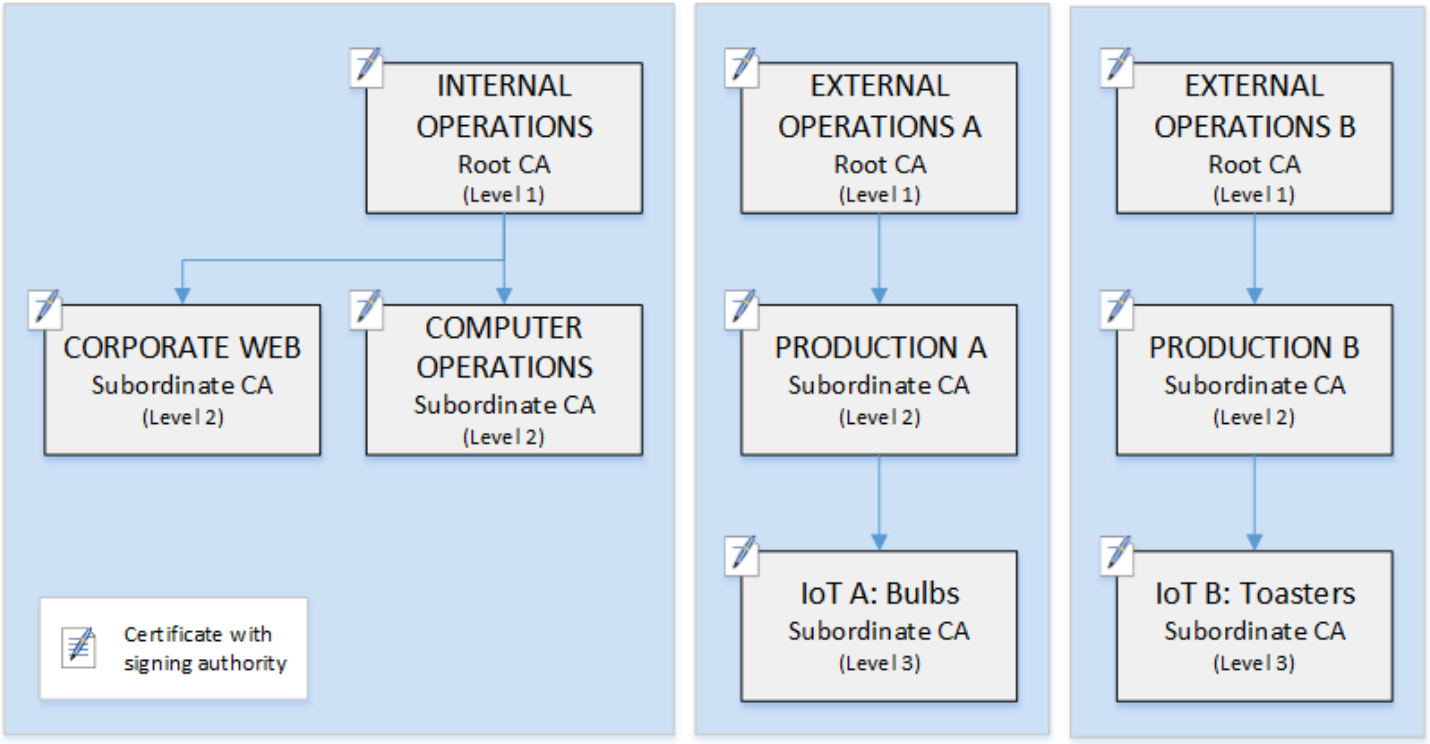
  Esta estructura se utiliza habitualmente para el desarrollo y las pruebas cuando no se requiere una cadena de confianza completa. Su uso en producción es atípico. Además, infringe la práctica recomendada de mantener políticas de seguridad separadas para la entidad de certificación raíz y las entidades de certificación que emiten certificados de entidad final.

  Sin embargo, si ya está emitiendo certificados directamente desde una CA raíz, puede migrar 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 entidades de certificación raíz separadas y profundidad adicional en el lado de operaciones externas es una decisión de diseño que satisface 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. Y con dos niveles de entidad de certificación subordinada, las entidades de certificación raíz tienen un alto nivel de aislamiento de las entidades de certificación de nivel 3 que son responsables de firmar en bloque los certificados de miles o millones de artículos manufacturados.

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.

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:

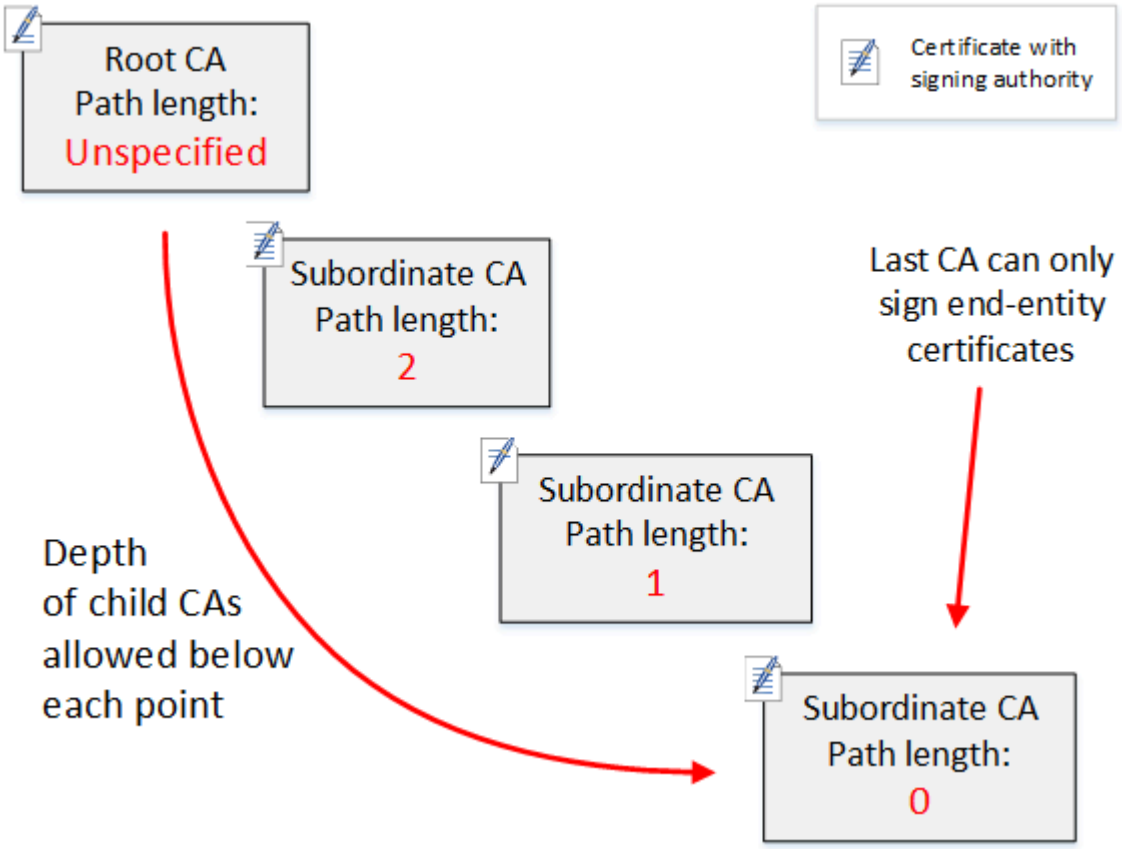
- `cA`: si el certificado define una CA. Si este valor es falso (el valor predeterminado), el certificado es un certificado de entidad final.
- `pathLenConstraint`: el número máximo de CA subordinadas de nivel inferior que pueden existir en una cadena de confianza válida. El certificado de la entidad 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.

Las entidades de certificación subordinadas tienen valores `pathLenConstraint` iguales o superiores a cero, en función de la ubicación en la ubicación de la jerarquía y de las características deseadas. Por ejemplo, en una jerarquía con tres entidades de certificación, no se especifica ninguna restricción de ruta para la entidad de certificación raíz. La primera entidad de certificación subordinada tiene una longitud de ruta de 1 y, por lo tanto, puede firmar entidades de certificación secundarias. Cada una de estas entidades de certificación secundarias debe tener necesariamente un valor `pathLenConstraint` de cero. Esto significa que pueden firmar certificados de entidad final pero no pueden emitir certificados de entidad de certificación adicionales. Limitar la potencia para crear nuevas entidades de certificación es un control de seguridad importante.

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). Pero la primera entidad de certificación subordinada tiene un valor `pathLenConstraint` de 2, lo que limita a sus entidades de certificación secundarias a profundizar 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.

Administrar 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:

- RootCACertificate/V1
- Certificado CA subordinado_ 0/V1 PathLen
- Certificado CA subordinado_ 1/V1 PathLen

- Certificado CA subordinado_ 2/V1 PathLen
- Certificado CA subordinado_ 3/V1 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 Descripción de las plantillas de certificados.

Automatización de la configuración de jerarquía de entidad de certificación 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 AWS CloudFormation .

Administración del ciclo de vida de entidad de certificación privada

Los certificados de entidad de certificación tienen una duración determinada o un período de validez. Cuando caduca un certificado de entidad de certificación, todos los certificados emitidos directa o indirectamente por las entidades de certificación subordinadas debajo de él en la jerarquía de entidades de certificación se vuelven no válidos. Puede evitar la caducidad del certificado de entidad de certificación planificando con antelación.

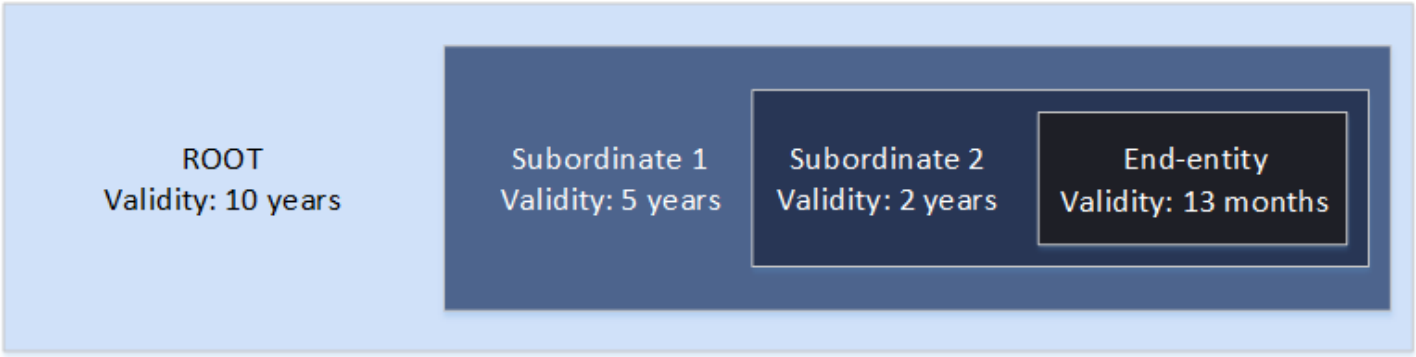
Selección de períodos de validez

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 ayudar a configurar los períodos de validez de los certificados sujetos 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. Dicho de otro modo, las entidades de certificación secundarias y los certificados de entidad final no pueden sobrevivir a sus 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 de mayor duración; los certificados de entidad final son relativamente efímeros; y las entidades de certificación subordinadas varían 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 CA emisora.

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.

Administración de sucesión de entidad de certificación

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.

Reemplazar una entidad de certificación 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 está desactivada, la

entidad de certificación antigua admite la revocación de certificados antiguos emitidos desde la entidad de certificación y, si está configurada, continúa validando las certificaciones mediante OCSP o las listas de revocación de certificados (CRL). 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 entidad de certificación antigua tiene entidades de certificación subordinadas, también debe reemplazarlas, ya que las entidades de certificación subordinadas caducan al mismo tiempo o antes de su entidad de certificación principal. Comience reemplazando la entidad de certificación más alta de la jerarquía que debe reemplazarse. A continuación, cree nuevas entidades de certificación subordinadas de reemplazo en cada nivel inferior posterior.

AWS recomienda incluir un identificador de generación de CA en los nombres de las CA, 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 simple convención de nomenclatura puede contribuir a evitar confusiones cuando ambas entidades de certificación no caducan.

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 la clave: las entidad de certificación que expidan más certificados deben rotarse 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.

Reemplazar 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 Creación e instalación del certificado para una 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.

Revocación de una entidad de certificación

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.

Configuración de un método de revocación de certificados

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, como 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 (CRL) o ambos.

> **Note**
>
> Si crea su CA sin configurar la revocación, siempre podrá configurarla más adelante. Para obtener más información, consulte Actualización de su entidad de certificación privada.

- 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 la revocación de los certificados sin necesidad de que los clientes operen la infraestructura ellos mismos. Los clientes pueden habilitar OCSP en las CA nuevas o existentes con una sola operación mediante la Autoridad de certificación privada de AWS consola, la API, la CLI o mediante AWS CloudFormation. Mientras que las CRL se almacenan y procesan en el punto de conexión y pueden quedar obsoletas, los requisitos de almacenamiento y procesamiento del OCSP se gestionan de forma sincrónica en el backend del sistema de respuesta.

  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. Las CA creadas en la consola de Autoridad de certificación privada de AWS utilizan el algoritmo de firma SHA256WITHRSA de forma predeterminada. Puede encontrar otros algoritmos compatibles en la documentación de la CertificateAuthorityConfigurationAPI.
  - Las plantillas de certificados APIPassthrough y CSRPassthrough no funcionarán con la extensión AIA si el sistema de respuesta de 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.
- Lista de revocación de certificados (CRL)

  La CRL contiene una lista de certificados revocados. Al configurar una CA para generar CRL, Autoridad de certificación privada de AWS incluye la extensión CRL Distribution Points en cada nuevo certificado emitido. Esta extensión proporciona la URL de la CRL. Las extensiones permiten a los clientes, como los navegadores web, consultar las CRL para determinar si se puede confiar en un certificado de entidad final o CA subordinada.

Como un cliente debe descargar las CRL y procesarlas localmente, su uso consume más memoria que el OCSP. Las CRL pueden consumir menos ancho de banda de la red porque la lista de CRL se descarga y se almacena en caché, en comparación con el OCSP, que comprueba el estado de revocación para cada nuevo intento de conexión.

> **Note**
>
> Tanto el OCSP como las CRL 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 las CRL, 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 vuelve a intentarlo cada 15 minutos.

Requisitos generales para las configuraciones de revocación

Los siguientes requisitos se aplican a todas las configuraciones de revocación.

- Una configuración que deshabilite las CRL o el OCSP debe contener solo el parámetro `Enabled=False` y fallará si se incluyen otros parámetros, como `CustomCname` o `ExpirationInDays`.
- 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 contiene un parámetro de nombre canónico (CNAME) personalizado para CRL o OCSP debe cumplir con las restricciones RFC7230 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

- Planificación de una lista de revocación de certificados (CRL)

- Configuración de una URL personalizada para OCSP de Autoridad de certificación privada de AWS

Planificación de una lista de revocación de certificados (CRL)

Antes de poder configurar una CRL como parte del proceso de creación de una CA, 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 Opciones de revocación de certificados y Configuración de una URL personalizada para OCSP de Autoridad de certificación privada de AWS.

Temas

- Estructura de CRL
- Políticas de acceso para las CRL en Amazon S3
- Habilitar el acceso público en bloque (BPA) de S3 con CloudFront
- Cifrado de las CRL
- Determinar el URI del punto de distribución CRL (CDP)

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
```

Las CRL 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 las CRL en 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 las entidades principales de servicio necesitan permiso de `Put` para permitir que Autoridad de certificación privada de AWS coloque objetos en el bucket y permiso de `Get` para recuperarlos. Durante el procedimiento de consola para crear una CA, puede optar por dejar que se Autoridad de certificación privada de AWS cree un nuevo depósito y aplicar una política de permisos predeterminada.

> **Note**
>
> La configuración de la política de IAM depende de la persona Regiones de AWS 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. 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.
> Al configurar las CRL 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`

PCA	S3	Entidad principal de servicio
Habilitado	Habilitado	`acm-pca.amazonaws.com`
Deshabilitad	Habilitado	`acm-pca.Region.amazonaws.com`
Habilitado	Deshabilitad	No compatible

La política predeterminada no aplica ninguna restricción de `SourceArn` a la CA. Le recomendamos que aplique manualmente la política menos permisiva que se muestra a continuación, que restringe el acceso tanto a una AWS cuenta específica como a una CA privada específica. Para obtener más información, consulte Agregar una política de bucket mediante la consola de Amazon S3.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"acm-pca.amazonaws.com"
         },
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:GetBucketAcl",
            "s3:GetBucketLocation"
         ],
         "Resource":[
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
         ],
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"account",
               "aws:SourceArn":"arn:partition:acm-pca:region:account:certificate-
authority/CA_ID"
```

```
            }
         }
      }
   ]
}
```

Si decide permitir la política predeterminada, siempre podrá modificarla más adelante.

Habilitar el acceso público en bloque (BPA) de S3 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 recuperan las CRL a través de la Internet pública (es decir, sin haber iniciado sesión en una cuenta), es necesaria una planificación adicional. AWS En esta sección se describe cómo configurar una solución de PKI privada mediante Amazon CloudFront, una red de entrega de contenido (CDN), para ofrecer CRL sin necesidad de acceso de 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.

Configuración de Amazon S3 con BPA

En S3, cree un nuevo bucket para su CRL, como de costumbre, y luego active el BPA en él.

Para configurar un bucket de Amazon S3 que bloquee el acceso público a su CRL

1. Cree un nuevo bucket de S3 mediante el procedimiento que se describe en Creación de un bucket. Durante el procedimiento, seleccione la opción Bloquear todo el acceso público.

   Para obtener más información, consulte Bloqueo del acceso público al almacenamiento de Amazon S3.
2. Cuando se haya creado el depósito, elija su nombre en la lista, vaya a la pestaña Permisos, elija Editar en la sección Propiedad del objeto y seleccione el Propietario del bucket preferido.
3. También en la pestaña Permisos, añada una política de IAM al bucket, tal y como se describe en Políticas de acceso para las CRL en Amazon S3 .

Configure el BPA CloudFront

Cree una CloudFront distribución que tenga acceso a su bucket privado de S3 y que pueda enviar las CRL 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 Procedimiento para crear una CA (CLI) .

  Al realizar el procedimiento, el archivo de revocación `revoke_config.txt` debe incluir las siguientes líneas para especificar un objeto 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.

Cifrado de las CRL

Si lo desea, puede configurar el cifrado en el bucket de Amazon S3 que contiene sus CRL. Autoridad de certificación privada de AWS admite dos modos de cifrado para los activos en Amazon 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. Seleccione 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. Seleccione 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
```

Determinar el URI del punto de distribución CRL (CDP)

Si usa el bucket S3 como el CDP de su CA, el URI del CDP puede tener uno de los siguientes formatos.

- `http://DOC-EXAMPLE-BUCKET.s3.region-code.amazonaws.com/crl/CA-ID.crl`
- `http://s3.region-code.amazonaws.com/DOC-EXAMPLE-BUCKET/crl/CA-ID.crl`

Si ha configurado su CA con un CNAME personalizado, el URI del CDP incluirá el CNAME, por ejemplo, `http://alternative.example.com/crl/CA-ID.crl`

Configuración de una URL personalizada para OCSP de Autoridad de certificación privada de AWS

> **Note**
>
> Este tema está dirigido a los clientes que desean personalizar la URL pública del punto de conexión del sistema de respuesta de 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 de Configurar la revocación.

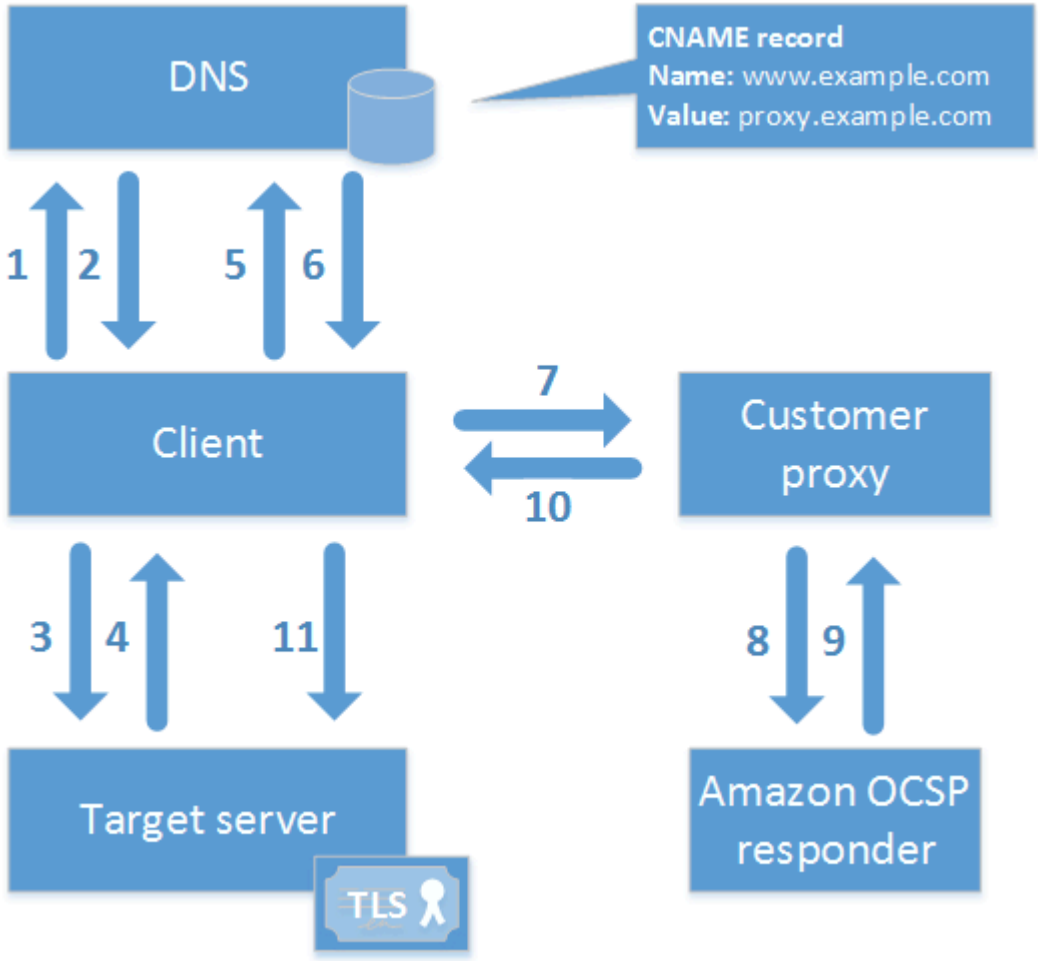
De forma predeterminada, al habilitar el OCSP para Autoridad de certificación privada de AWS, cada certificado que emita contiene la URL del respondedor de OCSP. AWS Esto permite a los clientes que soliciten una conexión criptográficamente segura enviar consultas de validación de OCSP directamente a 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 Procedimiento para crear una CA (CLI) .
- 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, consulte Ejemplo 2: crear una CA con OCSP activado y un CNAME personalizado y habilitado en Procedimiento para crear una CA (CLI) .
- 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.
>   - Configure ocsp.acm-pca.*<region>*.amazonaws.com como origen.
>   - 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 CloudFront distribución.

Modos de entidades de certificación

Autoridad de certificación privada de AWS admite la creación de una CA en cualquiera de los dos modos. Los modos GENERAL_PURPOSE y SHORT_LIVED_CERTIFICATE 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.

GENERAL_PURPOSE (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.

SHORT_LIVED_CERTIFICATE

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.

Las CA con el modo SHORT_LIVED_CERTIFICATE cuestan menos que las CA de uso general. Para obtener más información, consulte Precios de AWS Private Certificate Authority.

Para crear una CA que emita certificados de corta duración, defina el `UsageMode` parámetro en SHORT_LIVED_CERTIFICATE mediante el AWS CLIprocedimiento de creación de 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

Planificar la resiliencia

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 DR al planificar su jerarquía de CA. Autoridad de certificación privada de AWS está disponible en varias regiones, lo que le permite crear CA redundantes 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 CA raíz en dos AWS regiones diferentes para garantizar 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 entidades de certificación raíz redundantes aumenta la complejidad operativa: tendrá que distribuir ambos certificados de entidad de certificación raíz a los almacenes de confianza de los navegadores y los sistemas operativos de su entorno.
2. También puede crear CA subordinadas redundantes para implementarlas en cada una de sus regiones de AWS y encadenarlas a la misma CA raíz única en una sola región de 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.

Prácticas recomendadas de Autoridad de certificación privada de AWS

Las prácticas recomendadas son recomendaciones que pueden ayudarlo a utilizar Autoridad de certificación privada de AWS de forma más eficaz. Las siguientes prácticas recomendadas se basan en experiencias reales de clientes de AWS Certificate Manager y Autoridad de certificación privada de AWS actuales.

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

AWS recomienda documentar todas las políticas y prácticas para operar la entidad de certificación Esto podría incluir:

- Razonamiento de sus decisiones sobre la estructura de la entidad de certificación
- Un diagrama que muestra sus entidades de certificación y sus relaciones
- 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

En general, una entidad de certificación raíz sólo debe utilizarse para emitir certificados para las entidades de certificación intermedias. Esto permite que la entidad de certificación raíz se almacene

fuera de peligro mientras que las entidades de certificación intermedias realizan la tarea diaria de emitir certificados de entidad final.

Sin embargo, si la práctica actual de la organización consiste en emitir certificados de entidad final directamente desde una entidad de certificación raíz, Autoridad de certificación privada de AWS puede admitir este flujo de trabajo mientras mejora la seguridad y los controles operativos. 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 la suya a la CA raíz Cuenta de AWS

La creación de una CA raíz y una CA subordinada en dos cuentas AWS diferentes es una práctica recomendada. 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. El beneficio de este enfoque es que puede separar el control de sus entidades de certificación por cuenta. La desventaja es que no puede utilizar el asistente de AWS Management Console para simplificar el proceso de firma del certificado de CA de una CA subordinada desde su CA raíz.

> **Important**
>
> Recomendamos encarecidamente el uso de la autenticación multifactor (MFA) en cualquier momento que acceda a 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 la CA y el emisor del certificado residen en el mismo Cuenta de AWS lugar, 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. Por lo general, los clientes se niegan a aceptar los certificados revocados. Autoridad de certificación privada de AWS ofrece dos opciones estándares para la revocación gestionada: el Protocolo de estado de certificados en línea (OCSP) y las listas de revocación de certificados (CRL). Para obtener más información, consulte Configuración de un método de revocación de certificados.

Activar 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 incluirá las llamadas a la API realizadas desde la AWS Management Console, los SDK de AWS, la AWS Command Line Interface y los servicios de AWS de nivel superior. 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. Puedes CloudTrail integrarlo en las aplicaciones mediante la API, automatizar la creación de rutas para tu 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. Vaya a Usando CloudTrail para ver ejemplos de registros de seguimiento de las operaciones de Autoridad de certificación privada de AWS.

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 una entidad de certificación no utilizada

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 su entidad de certificación privada.

> **Note**
>
> AWS le factura por una entidad de certificación hasta que se haya eliminado.

Bloquear el acceso público de sus CRL

Autoridad de certificación privada de AWS recomienda utilizar la característica Bloqueo de acceso público de Amazon S3 (BPA) en los buckets que contienen las CRL. 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 Habilitar el acceso público en bloque (BPA) de S3 con CloudFront.

Prácticas recomendadas de aplicaciones de Amazon EKS

Cuando utilice Autoridad de certificación privada de AWS para aprovisionar certificados X.509 a Amazon EKS, siga las recomendaciones para proteger los entornos con varios inquilinos que figuran en las Guías de prácticas recomendadas de Amazon EKS. Para obtener información general sobre la integración Autoridad de certificación privada de AWS con Kubernetes, consulte Protección de Kubernetes con Autoridad de certificación privada de AWS.

Administración de CA privada

Con Autoridad de certificación privada de AWSél, puede crear una jerarquía completamente AWS alojada de autoridades de certificación (CA) 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 (CRL) o ambos. Autoridad de certificación privada de AWS almacena y administra los certificados de CA, las CRL y las respuestas de OCSP, y las claves privadas de las 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 AWS Management Console, 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 Uso de la API de Autoridad de certificación privada de AWS (ejemplos de Java).

Temas

- Restauración de una CA privada

Creación de una entidad de certificación (CA) privada

Puede utilizar estos procedimientos para crear entidades de certificación (CA) raíz y entidades de certificación subordinadas, lo que da como resultado una jerarquía auditable de relaciones de confianza que coincida con las necesidades de la organización. Puede crear una CA mediante la AWS Management Console parte de la PCA de AWS CLI, o AWS CloudFormation.

Para obtener información sobre la actualización de la configuración de una CA que ya ha creado, consulte Actualización de su entidad de certificación privada.

Para obtener información sobre el uso de una entidad de certificación para firmar certificados de entidad final para los usuarios, dispositivos y aplicaciones, consulte Emisión de certificados de entidad final privadas.

> **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

- Procedimiento para crear una CA (consola)
- Procedimiento para crear una CA (CLI)
- Se utiliza para crear una CA AWS CloudFormation

Procedimiento para crear una CA (consola)

Utilice los siguientes pasos para crear una CA privada mediante la AWS Management Console.

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 abre la consola en una región en la que no tiene CA privadas, aparecerá la página de introducción. Elija Crear una CA privada.
- Si abre la consola en una región en la que ya ha creado una CA, se abre la página Autoridades de certificación privadas con una lista de sus CA. Seleccione Crear CA.

Opciones de modo

En la sección Opciones de modo de la consola, elija el modo de caducidad de los certificados que emite la CA.

- Uso general: emite certificados que se pueden configurar con cualquier fecha de caducidad. Esta es la opción predeterminada.
- 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.

Opciones de tipo de CA

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. Sirve como la autoridad de firma definitiva para otras entidades de certificación y certificados de 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. Normalmente, las CA subordinadas se utilizan para crear otras CA subordinadas o para emitir certificados de entidad final a usuarios, computadoras y aplicaciones.

> **Note**
>
> Autoridad de certificación privada de AWS proporciona un proceso de firma automatizado cuando la CA principal de su CA subordinada también está alojada Autoridad de certificación privada de AWS en. 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 Instalación de un certificado de entidad de certificación subordinada firmado por una entidad de certificación principal externa.

Opciones de nombre distintivos del asunto

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 plantilla APIPassthrough 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.

Opciones de algoritmos de clave

En Opciones de algoritmos clave, elija el algoritmo clave y el tamaño en bits de la clave. El valor predeterminado es un algoritmo RSA con una longitud clave de 2048 bits. Puede elegir entre los siguientes algoritmos:

- RSA 2048

- RSA 4096
- ECDSA P256
- ECDSA P384

Opciones de revocación de certificados

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 Configuración de un método de revocación de certificados 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 Actualización de su entidad de certificación privada.

Para configurar una CRL

1. En Opciones de revocación de certificados, elija Activar la distribución de CRL.
2. Si desea crear un bucket de Amazon S3 para las entradas de CRL, seleccione Create a new S3 bucket (Crear un nuevo bucket de S3) y escriba un nombre de bucket que sea único. (No es necesario incluir la ruta de acceso al bucket). De lo contrario, en S3 bucket URI (URI del bucket de S3), seleccione un bucket existente de la lista.

   Al crear un bucket nuevo a través de la consola, Autoridad de certificación privada de AWS intenta adjuntar la política de acceso requerida al bucket e inhabilitar la configuración de bloqueo de acceso público (BPA) predeterminada de S3. Si, por el contrario, especifica un bucket existente, debe asegurar que el BPA esté desactivado para la cuenta y para el bucket. De lo contrario, se produce un error en la operación para crear la CA. Si la CA se ha creado

correctamente, debe adjuntar manualmente una política antes de poder empezar a generar las CRL. Utilice uno de los patrones de políticas descritos en Políticas de acceso para las CRL en Amazon S3 . Para obtener más información, consulte Agregar una política de bucket mediante la consola de Amazon S3.

> **Important**
>
> Se produce un error al intentar crear una CA mediante la Autoridad de certificación privada de AWS consola si se cumplen todas las condiciones siguientes:
>
> - Está configurando una CRL.
> - Solicita Autoridad de certificación privada de AWS crear un bucket de S3 automáticamente.
> - Está aplicando la configuración de BPA en S3.
>
> En esta situación, la consola crea un bucket, pero intenta hacerlo accesible al público y no lo consigue. Compruebe la configuración de Amazon S3 si esto ocurre, desactive el BPA según sea necesario y, a continuación, repita el procedimiento para crear una CA. Para obtener más información, consulte Bloqueo del acceso público al almacenamiento de Amazon S3.

3. Expanda Configuración de CRL para obtener opciones de configuración adicionales.

   - 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.
   - Escriba la Validez en días que la CRL estará en vigor. El valor predeterminado es 7 días. En el caso de las CRL en línea, es habitual establecer un periodo de validez de entre dos y siete días. Autoridad de certificación privada de AWS intenta regenerar la CRL en el punto medio del periodo especificado.

4. Amplíe los ajustes de S3 para configurar de forma opcional el control de versiones de los buckets y el registro de acceso a los buckets.

Para configurar OCSP

1. En Opciones de revocación de certificados, seleccione 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 que no sea de Amazon.

   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 AWS OCSP. 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 Configuración de una URL personalizada para OCSP de Autoridad de certificación privada de AWS.

   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://”.

Agregue etiquetas

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 a las CA tras su creación, consulte. Administrar las etiquetas de 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.

Opciones de permisos CA

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.

Precios

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.

Crear CA

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:
>
> - Si está instalando un certificado de CA raíz
> - Instalación de un certificado de CA subordinado hospedado por Autoridad de certificación privada de AWS
> - Instalación de un certificado de entidad de certificación subordinada firmado por una entidad de certificación principal externa

Procedimiento para crear una CA (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 utilizar el OCSP o una 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 las CRL en 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
- 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 Habilitar el acceso público en bloque (BPA) de S3 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 sobre la extensión CDP, consulte. Determinar el URI del punto de distribución CRL (CDP) CustomCname No se puede establecer A si OmitExtension es «verdadero».

> **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).

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 Configuración de una URL personalizada para OCSP de Autoridad de certificación privada de AWS.

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":"DOC-EXAMPLE-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": "DOC-EXAMPLE-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":"DOC-EXAMPLE-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": "DOC-EXAMPLE-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 almacén Enterprise NTauth de Microsoft Active Directory (AD), donde puede 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 de certificación (CA) de terceros al almacén Enterprise 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 el almacén de NTauth. 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: Cree una CA de Matter con una CRL adjunta y la extensión CDP omitida 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 las CRL 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":"DOC-EXAMPLE-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": "DOC-EXAMPLE-BUCKET",
   "CrlDistributionPointExtensionConfiguration":{
  "OmitExtension":true
   }
   },
   ...
}
...
```

Se utiliza para crear una CA AWS CloudFormation

Para obtener información sobre cómo crear una CA privada mediante AWS CloudFormation, consulte la referencia de tipos de Autoridad de certificación privada de AWS recursos en la Guía del AWS CloudFormation usuario.

Creación e instalación del certificado para una 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 de la consola Entidades de certificación privadas.

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 de certificación principal con el algoritmo de firma RSA puede emitir certificados con los siguientes algoritmos:
  - SHA256 RSA
  - SHA384 RSA
  - SHA512 RSA
- En el pasado Región de AWS, una entidad emisora de certificados principal con el algoritmo de firma EDCSA podía emitir certificados con los siguientes algoritmos:
  - SHA256 ECDSA
  - SHA384 ECDSA
  - SHA512 ECDSA

  Los sistemas heredados incluyen Regiones de AWS :

Nombre de la región	Ubicación geográfica
eu-north-1	Europa (Estocolmo)
me-south-1	Medio Oriente (Baréin)
ap-south-1	Asia-Pacífico (Bombay)

Nombre de la región	Ubicación geográfica
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)
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)

Nombre de la región	Ubicación geográfica
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.

Si está instalando un certificado de CA raíz

Puede instalar un certificado de CA raíz desde o desde. AWS Management Console AWS CLI

Para crear e instalar un certificado para su entidad de certificación raíz privada (consola)

1. (Opcional) Si aún no se encuentra en la página de detalles de la CA, abra la consola de Autoridad de certificación privada de AWS en https://console.aws.amazon.com/acm-pca/home. 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 información[Algoritmos de firma compatibles](), consulte[Algoritmos criptográficos compatibles](), 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": "DOC-EXAMPLE-BUCKET1"
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

Instalación de un certificado de CA subordinado hospedado por Autoridad de certificación privada de AWS

Puede usarlo AWS Management Console 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 se encuentra en la página de detalles de la CA, abra la consola de Autoridad de certificación privada de AWS en https://console.aws.amazon.com/acm-pca/home. 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 las CA 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 entidad de certificación subordinada puede emitir certificados para crear entidades de certificación adicionales subordinadas 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 Descripción de las plantillas de certificados.

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.

Instalación de un certificado de entidad de certificación subordinada firmado por una entidad de certificación principal externa

Tras crear una CA privada subordinada, tal como se describe en Procedimiento para crear una CA (consola) o Procedimiento para crear una CA (CLI) , 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 Certificados de CA privados firmados externamente .

Controlar el acceso a una 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 de ACM, la RequestCertificateAPI de 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

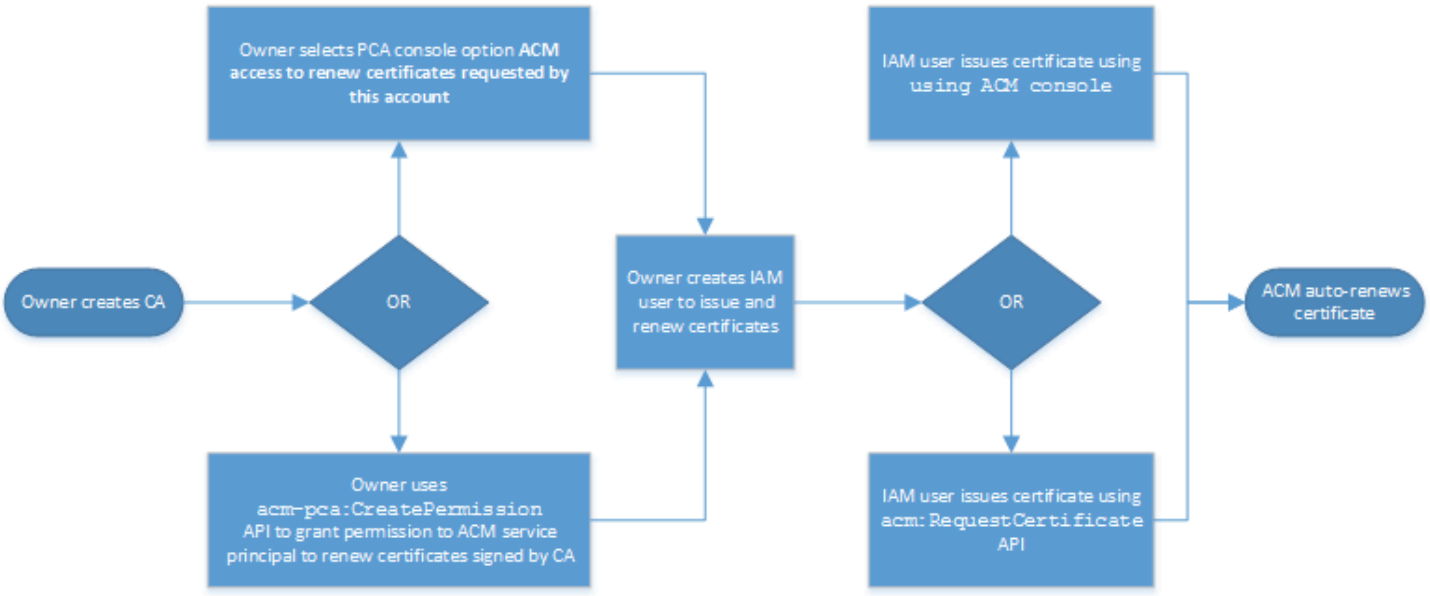
Caso 1 de cuenta única: emisión de un certificado no administrado

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 en https://console.aws.amazon.com/acm-pca/home.
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 Guardar.

Para gestionar 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.

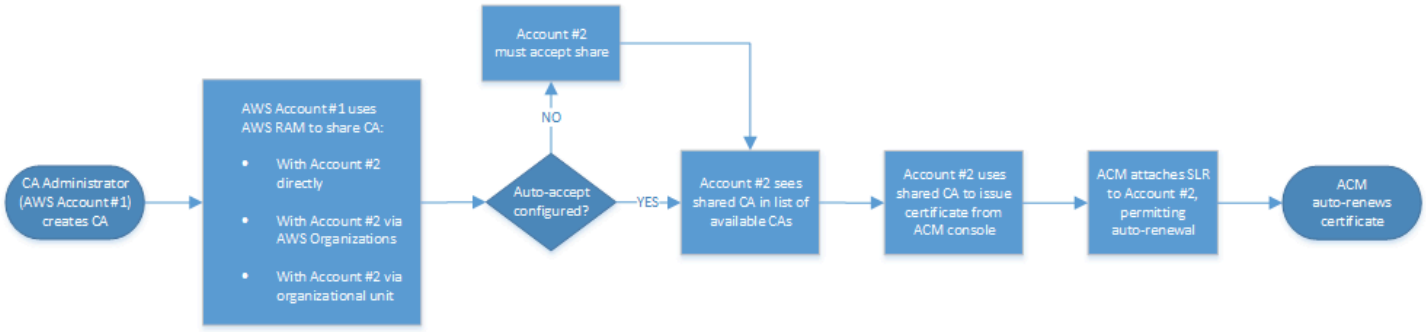
  > **Note**
  >
  > Para abrir fácilmente la consola de RAM, seleccione una CA y, a continuación, seleccione Acciones y Administrar recursos compartidos.

- De forma programática, mediante las API de la PCA, y. PutPolicyGetPolicyDeletePolicy
- 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.



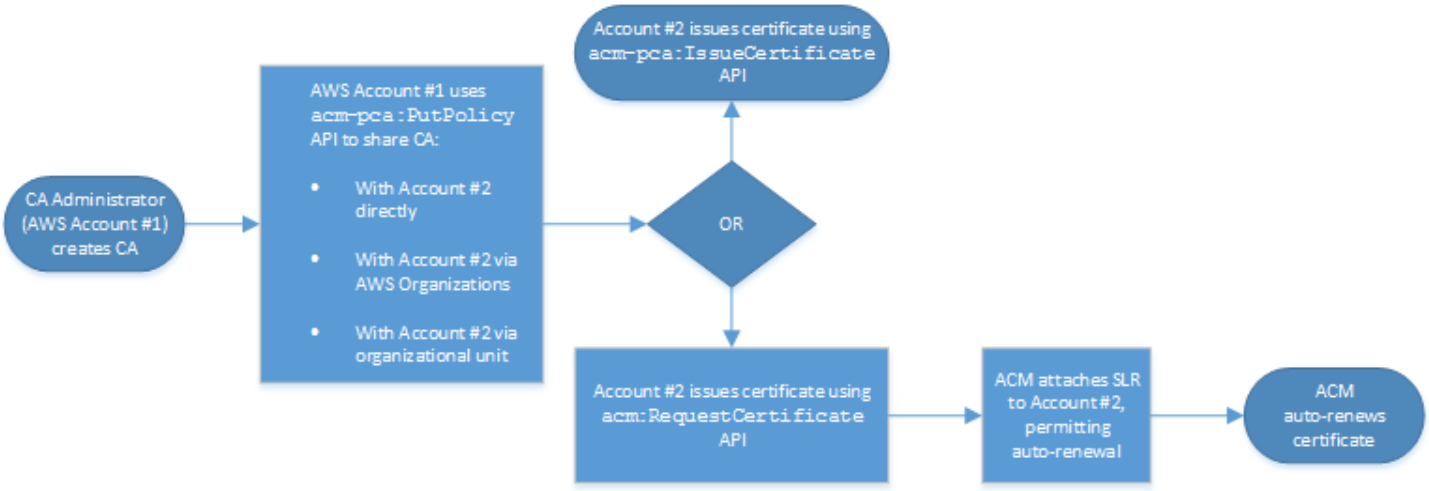
Una vez que RAM comparte un recurso AWS Organizations, el destinatario principal debe aceptar el recurso para que surta efecto. El destinatario puede configurarse AWS Organizations para aceptar automáticamente las acciones ofrecidas.

> **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.

Mostrar las CA privadas

Puede usar la Autoridad de certificación privada de AWS consola o AWS CLI para enumerar las CA privadas que sean de su propiedad o a las que tenga acceso.

Para enumerar las CA disponibles mediante la consola

1. Inicie sesión en su AWS cuenta y abra la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/home.
2. Revise la información de la lista de Entidades de certificación privadas. Puede explorar diferentes páginas de CA utilizando los números de página de la parte 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 RSA_2048, RSA_4096, EC_prime256v1 y EC_secp384r1.
- 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 SHA256WITHECDSA,

SHA384WITHECDSA, SHA512WITHECDSA, SHA256WITHRSA, SHA384WITHRSA y SHA512WITHRSA.

> **Note**
>
> Puede personalizar las columnas que desee mostrar, así como otros ajustes, seleccionando el icono de ajustes situado en la esquina superior derecha de la consola.

Para ver una lista de las CA disponibles, utilice la AWS CLI

Utilice el list-certificate-authoritiescomando para enumerar las CA disponibles, 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 utilizar 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 sobre la actualización de un secreto, consulte Actualización de su entidad de certificación privada.

Para ver los detalles de la CA en la consola

1. Inicie sesión en su AWS cuenta y abra la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/home.
2. Revise la lista de Entidades de certificación privadas. Puede explorar diferentes páginas de CA utilizando los números de página de la parte 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 Opciones de nombre distintivos del asunto. 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 RSA 2048, RSA 4096, ECDSA P2567 y ECDSA P384.
   - 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 SHA256 ECDSA, SHA384 ECDSA, SHA512 ECDSA, SHA256 RSA, SHA384 RSA y SHA512 RSA

- Estándar de seguridad del almacenamiento de claves: nivel conforme a las normas federales de procesamiento de información. Los valores posibles son el FIPS 140-2 de nivel 3 o superior y el FIPS 140-2 de nivel 3 o superior. Este parámetro varía según la región AWS .

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).

Administrar las etiquetas de 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 utilizar la Autoridad de certificación privada de AWS consola, AWS Command Line Interface (AWS CLI) o la API de la PCA para añadir, ver o eliminar etiquetas de las CA privadas.

Puede agregar o quitar etiquetas personalizadas para su CA privada en cualquier momento. Por ejemplo, podría etiquetar entidades de certificación privadas con pares clave-valor de `Environment=Prod` o `Environment=Beta` para identificar el entorno para el que está pensada 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 para instancias de Linux.

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 AWS Management Console.

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. Inicie sesión en su AWS cuenta y abra la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/home.
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 Guardar.

Para etiquetar una entidad de certificación privada (AWS CLI)

Utilice el tag-certificate-authoritycomando para añadir etiquetas a su 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
```

Actualización de su entidad de certificación privada

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 las CA, junto con ejemplos de actualizaciones de la consola y la CLI a las CA.

Actualización del estado de la entidad de certificación

El estado de una CA gestionada por una entidad emisora es el Autoridad de certificación privada de AWS resultado de 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 CRL	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. •						No

Status	Emitir certificados	Valide los certificados con OCSP	Genera CRL	Genere auditorí s	Puede actualizar el certificado de CA	Los certificados se pueden revocar	Se le factura por la CA
	Si llama a la API `RestoreCertificateAuthority` en una entidad de certificación con el estado `DELETED` y un 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 su entidad de certificación privada.						

* Para completar la activación, debe generar una CSR, obtener un certificado de CA firmado por una CA e importar el certificado a 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 Creación e instalación del certificado para una 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.

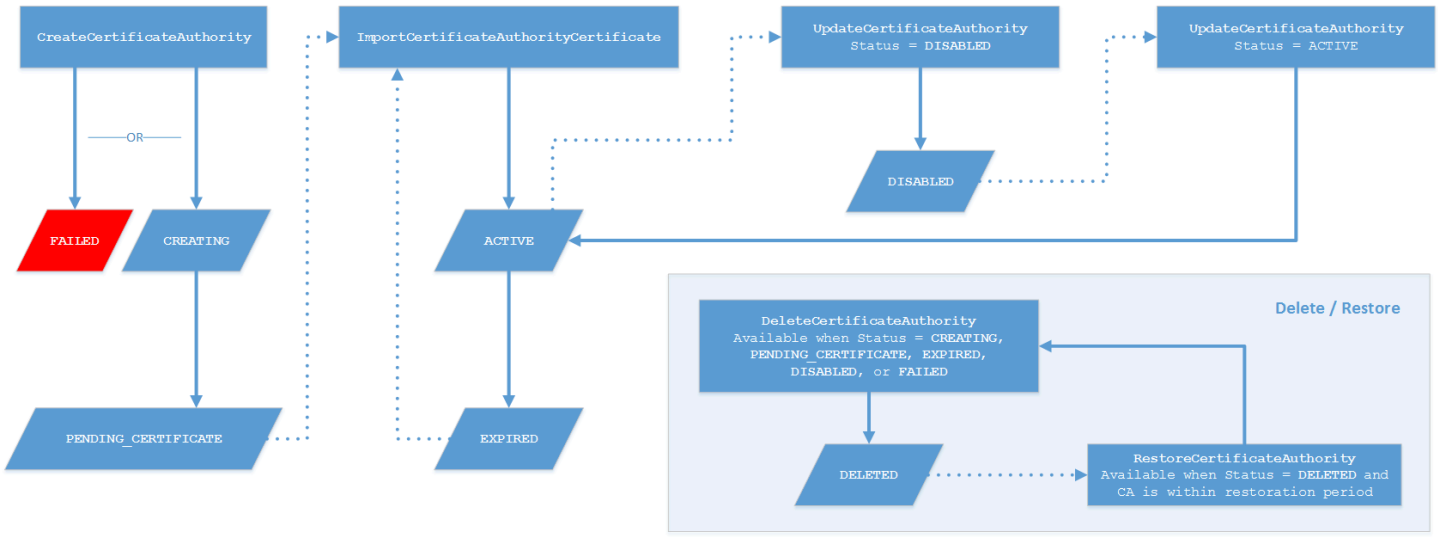
Consideraciones adicionales sobre los certificados de CA caducados:

- Los certificados de CA no se renuevan automáticamente de forma predeterminada. Para obtener información sobre la automatización de 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.

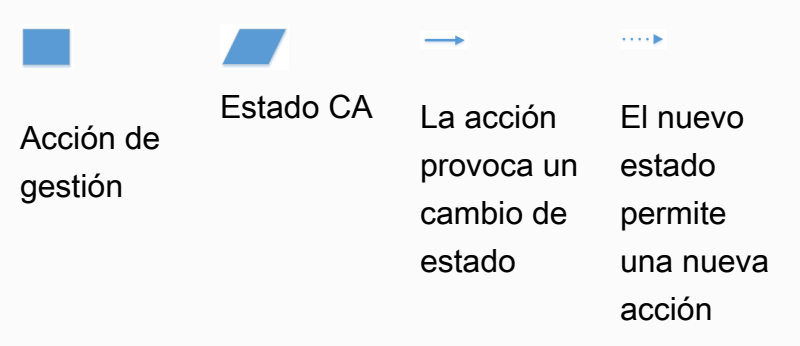
- Las API The `ListCertificateAuthorities` y `DescribeCertificateAuthority` devuelven un estado `EXPIRED` si el certificado de entidad de certificación ha caducado, con independencia de si el estado de la entidad de certificación está establecido en `ACTIVE` o `DISABLED`. Sin embargo, si la entidad de 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.

Estado de la CA y 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.

Actualización de una CA (consola)

Los siguientes procedimientos muestran cómo actualizar configuraciones de CA existentes utilizando AWS Management Console.

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. Inicie sesión en su AWS cuenta y abra la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/home
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:

- 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 empieza 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. Inicie sesión en su AWS cuenta y abra la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/home.
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 Configuración de un método de revocación de certificados 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 crea un depósito nuevo, Autoridad de certificación privada de AWS crea y adjunta la política de acceso requerida. Si decide utilizar un bucket existente, debe adjuntarle una política de acceso para poder empezar a generar las CRL. Utilice uno de los patrones de políticas descritos en Políticas de acceso para las CRL en 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.

- 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.
- Escriba el número de días que la CRL estará en vigor. El valor predeterminado es 7 días. En el caso de las CRL en línea, 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.

   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 en 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 Configuración de una URL personalizada para OCSP de Autoridad de certificación privada de AWS.

   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": "DOC-EXAMPLE-BUCKET1"
   },
            "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": "DOC-EXAMPLE-BUCKET1"
            },
```

```
            "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": "bucket-name"
   }
}
```

> **Note**
>
> Al actualizar una CA de atestación de dispositivos Matter para habilitar las CRL, debe configurarla para que omita la extensión CDP de los certificados emitidos a fin de 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": "bucket-name"
>       "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_numbner",
        "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": "DOC-EXAMPLE-BUCKET1",
            },
            "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 su entidad de certificación privada

Puede eliminar una CA privada de la AWS Management Console 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 Restauración de 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 eliminar una entidad de certificación privada (consola)

1. Inicie sesión en su AWS cuenta y abra la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/home.
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, llame 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
```

Restauración de 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 su entidad de certificación 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.

Restauración de una CA privada (consola)

Puede utilizar el AWS Management Console para restaurar una CA privada.

Para restaurar una CA privada (consola)

1. Inicie sesión en su AWS cuenta y abra la Autoridad de certificación privada de AWS consola en https://console.aws.amazon.com/acm-pca/home.
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.

Restauración de una CA privada (AWS CLI)

Utilice el restore-certificate-authoritycomando para restaurar una CA privada eliminada que se encuentre en ese `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
```

Administración de certificados

Una vez que haya creado y activado una entidad de certificación (CA) privada y haya configurado el acceso a ella, usted o sus usuarios autorizados podrán realizar las tareas descritas en esta sección. Si aún no ha configurado políticas (IAM) AWS Identity and Access Management 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 Controlar el acceso a una CA privada.

Temas

Emisión de certificados de entidad final privadas

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.

Capability	ACM	Autoridad de certificación privada de AWS
Emitir un certificado de entidad final	✓ (utilizando RequestCertificate o la consola)	✓ (usando IssueCertificate)
Asociación con equilibradores de carga y servicios conectados a Internet AWS	✓	No compatible

Capability	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, aplica 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 Descripción de las plantillas de certificados.

Si bien los certificados ACM están diseñados en torno a la confianza pública, Autoridad de certificación privada de AWS satisfacen las necesidades de su PKI privada. En consecuencia, puede configurar los certificados mediante la API Autoridad de certificación privada de AWS 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 mediante Autoridad de certificación privada de AWS, puede importarlo a ACM y utilizarlo con un servicio AWS compatible.

> **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.

Emitir 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 API de Autoridad de certificación privada de AWS o AWS CLI para emitir un certificado privado, el certificado no se administra, lo que significa que no puede usar la consola 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 los CSR. Esto garantiza que Autoridad de certificación privada de AWS analice correctamente los datos.

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 comando issue-certificate. 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 en Emitir un certificado estándar (AWS CLI), se pasan dos argumentos adicionales al comando issue-certificate: el ARN de una plantilla APIPassthrough y un archivo de configuración JSON que especifica los atributos personalizados y sus identificadores de objeto (OID). No se puede usar `StandardAttributes` junto con `CustomAttributes`. Sin embargo, se pueden pasar los OID estándar como parte de `CustomAttributes`. Los OID de nombres de asunto predeterminados se muestran en la siguiente tabla (información de la RFC 4519 y de la base de datos de referencia global sobre los 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
title		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 comando issue-certificate: el ARN de una plantilla APIPassthrough y un archivo de configuración JSON que especifica las extensiones personalizadas, y una CSR como la que se muestra en Emitir 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
```

Recuperar un certificado privado

Puede usar la API de Autoridad de certificación privada de AWS y la AWS CLI para emitir un certificado privado. Si lo hace, puede usar la AWS CLI o la API de Autoridad de certificación privada de AWS 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

Utilice el comando get-certificate AWS CLI para recuperar un certificado de entidad final privado. También puede utilizar la operación 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 Uso de 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 utilizar la Autoridad de certificación privada de AWS y la API de la AWS CLI para recuperar el certificado de la entidad de certificación (CA) para la entidad de certificación 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----
```

Listado de 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 Uso de

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"
}
```

Exportación de 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 utilizar 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 Exportar un certificado privado en la Guía del usuario de AWS Certificate Manager.

Como ventaja adicional, AWS Certificate Manager ofrece la renovación administrada de los certificados privados que se emitieron mediante la consola de ACM, la acción `RequestCertificate` de la API de ACM o el comando request-certificate de la sección ACM de la AWS CLI. Para obtener más información sobre las renovaciones, consulte Renovación de certificados en una PKI privada.

Revocación de 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 RevokeCertificateAPI. 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 administrados que permiten comprobar el estado de las revocaciones: el Protocolo de estado de los certificados en línea (OCSP) y las listas de revocación de certificados (CRL). 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 las CRL 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 administrada mediante Autoridad de certificación privada de AWS, consulte Configuración de un método de revocación de certificados.

Los certificados revocados siempre se registran en el informe de auditoría de Autoridad de certificación privada de AWS.

> **Note**
>
> Los emisores de certificados multicuenta necesitan permisos adicionales para revocar los certificados que emiten; de lo contrario, el propietario de la CA debe realizar la revocación.

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

Use 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 las CRL, que los clientes pueden almacenar en caché durante días, los clientes no suelen almacenar en caché las respuestas del OCSP.

Certificados revocados en una CRL

Las CRL se actualizan aproximadamente 30 minutos después de que un certificado se revoque. Si, por algún motivo, se produce un error en la actualización de la CRL, la entidad de certificación privada de Autoridad de certificación privada de AWS realizará más 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 configuración de CRL, consulte Planificación de una lista de revocación de certificados (CRL).

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 Uso de 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"
   }
]
```


Automatización de la exportación de un certificado renovado

Al usar Autoridad de certificación privada de AWS para crear una CA, puede importarla a AWS Certificate Manager y dejar que ACM administre 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.

Descripción de las plantillas de certificados

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**
>
> Los usuarios de AWS Certificate Manager (ACM) con acceso compartido entre cuentas a una CA privada pueden emitir certificados administrados 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_API Passthrough/V1
> - BlankEndEntityCertificate_API SRPassthrough/v1
> - Certificado CA subordinado_ 0/V1 PathLen
>
> Para obtener más información, consulte Políticas basadas en recursos.

Temas

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.

- Plantillas CSRPassthrough

  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 Orden de operaciones de la plantilla.

- Plantillas APIPassthrough

  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 Orden de operaciones de la plantilla.

- Plantillas APICSRPassthrough

  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 Orden de operaciones de la plantilla.

En la siguiente tabla se indican los tipos de plantilla admitidos por Autoridad de certificación privada de AWS con enlaces a sus definiciones.

> **Note**
>
> Para obtener información sobre los ARN de plantilla en las regiones, consulte la Guía del usuario. GovCloud AWS Private Certificate AuthorityAWS GovCloud (US)

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
OCSP /V1 SigningCertificate	`arn:aws:acm-pca:::template/OCSPSigningCertificate/V1`	Firma de OCSP

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
RootCACertificate/V1	arn:aws:acm-pca:::template/RootCACertificate/V1	CA
Certificado CA subordinado_ 0/V1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1	CA
Certificado CA subordinado_ 1/V1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1/V1	CA
Certificado CA subordinado_ 2/V1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2/V1	CA
Certificado CA subordinado_ 3/V1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3/V1	CA

Plantillas CSRPassthrough

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
BlankEndEntityCertificate_CSRPassThrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CSRPassthrough/V1	Entidad final
BlankEndEntityCertificate_ _CSRPassthrough/v1 CriticalBasicConstraints	arn:aws:acm-pca:::template/BlankEndEntityCertificate_C	Entidad final

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
	riticalBasicConstraints_CSRPassthrough/V1	
BlankSubordinateCertificado CA_PathLen 0_CSRPassThrough/v1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
BlankSubordinateCertificado CA_1_CSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
BlankSubordinateCertificado CA_2_CSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen 3_CSRPassThrough/v1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA
CodeSigningCertificate_CSRPassThrough/v1	arn:aws:acm-pca:::template/CodeSigningCertificate_CSRPassthrough/V1	Firma de código

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
EndEntityCertificate_CSRPassthrough/v1	`arn:aws:acm-pca:::template/EndEntityCertificate_CSRPassthrough/V1`	Entidad final
EndEntityClientAuthCertificate_CSRPassthrough/v1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_CSRPassthrough/V1`	Entidad final
EndEntityServerAuthCertificate_CSRPassthrough/v1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_CSRPassthrough/V1`	Entidad final
OCSP _CSRPassThrough/v1 SigningCertificate	`arn:aws:acm-pca:::template/OCSPSigningCertificate_CSRPassthrough/V1`	Firma de OCSP
Certificado CA subordinado_ PathLen 0_CSRPassthrough/v1	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_CSRPassthrough/V1`	CA
Certificado CA subordinado_ 1_CSR Passthrough/v1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_CSRPassthrough/V1`	CA

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
Certificado CA subordinado_2_CSR Passthrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
Certificado CA subordinado_3_CSR Passthrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA

Plantillas APIPassthrough

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
BlankEndEntityCertificate_APIPassThrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_APIPassthrough/V1	Entidad final
BlankEndEntityCertificate__APIPassthrough/v1 CriticalBasicConstraints	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough/V1	Entidad final
CodeSigningCertificate_APIPassthrough/v1	arn:aws:acm-pca:::template/CodeSigningCertificate_APIPassthrough/V1	Firma de código
EndEntityCertificate_APIPassthrough/v1	arn:aws:acm-pca:::template/EndEntity	Entidad final

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
	`Certificate_APIPas sthrough/V1`	
EndEntityClientAuthCertific ate_APIPassthrough/v1	`arn:aws:acm-pca::: template/EndEntity ClientAuthCertific ate_APIPassthrough/V1`	Entidad final
EndEntityServerAuthCertific ate_APIPassthrough/v1	`arn:aws:acm-pca::: template/EndEntity ServerAuthCertific ate_APIPassthrough/V1`	Entidad final
OCSP _APIPassthrough/v1 SigningCertificate	`arn:aws:acm-pca::: template/OCSPSigni ngCertificate_APIP assthrough/V1`	Firma de OCSP
RootCACertificate_APIPassth rough/V1	`arn:aws:acm-pca::: template/RootCACer tificate_APIPassth rough/V1`	CA
BlankRootCACertificate_API Passthrough/v1	`arn:aws:acm-pca::: template/BlankRoot CACertificate_APIP assthrough/V1`	CA
BlankRootCACertificate_ 0_APIPassthrough/v1 PathLen	`arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len0_APIPassthrough/V1`	CA

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
BlankRootCACertificate_1_APIPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen1_APIPassthrough/V1	CA
BlankRootCACertificate_2_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen2_APIPassthrough/V1	CA
BlankRootCACertificate_3_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen3_APIPassthrough/V1	CA
Certificado CA subordinado_0_API Passthrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APIPassthrough/V1	CA
BlankSubordinateCACertificate_0_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1	CA
Certificado CA subordinado_1_API Passthrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APIPassthrough/V1	CA

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
BlankSubordinateCACertificate_1_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
Certificado CA subordinado_2_API Passthrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
BlankSubordinateCACertificate_2_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
Certificado CA subordinado_3_API Passthrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APIPassthrough/V1	CA
BlankSubordinateCACertificate_3_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1	CA

Plantillas APICSRPassthrough

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
BlankEndEntityCertificate_APISRPassThrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_APICSRPassthrough/V1	Entidad final
BlankEndEntityCertificate_CriticalBasicConstraints_APICSRPassThrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APICSRPassthrough/V1	Entidad final
CodeSigningCertificate_APICSRPassThrough/v1	arn:aws:acm-pca:::template/CodeSigningCertificate_APICSRPassthrough/V1	Firma de código
EndEntityCertificate_APICSRPassThrough/v1	arn:aws:acm-pca:::template/EndEntityCertificate_APICSRPassthrough/V1	Entidad final
EndEntityClientAuthCertificate_APICSRPassThrough/v1	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_APICSRPassthrough/V1	Entidad final
EndEntityServerAuthCertificate_APICSRPassThrough/v1	arn:aws:acm-pca:::template/EndEntityServerAuthCertific	Entidad final

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
	ate_APICSRPassthrough/V1	
OCSP _APICSRPassThrough/v1 SigningCertificate	arn:aws:acm-pca:::template/OCSPSigningCertificate_APICSRPassthrough/V1	Firma de OCSP
Certificado CA PathLen subordinado_ 0_APICSRPassThrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
BlankSubordinateCertificado CA_ 0_APICSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
Certificado CA subordinado_ 1_API PathLen CSR PassThrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA
BlankSubordinateCertificado CA_ 1_API CSR PassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA

Nombre de la plantilla	ARN de plantilla	Tipo de certificado
Certificado CA subordinado_ PathLen 2_API SRPassThrough/ 3_API PassThrough V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APICSRPassthrough/V1`	CA
BlankSubordinateCACertificate_ 2_APICSRPassThrough/v1 PathLen	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1`	CA
Certificado CA subordinado_ 3_API PathLen CSR PassThrough/v1	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA
BlankSubordinateCertificadoCA_ 3_API CSR PassThrough/v1 PathLen	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	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. Los valores de acceso directo de CSR solo se respetan cuando se utiliza una plantilla de acceso directo CSRPassthrough o de acceso directo de 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 si se copiara 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.

Definiciones de plantilla

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 sencillo que Autoridad de certificación privada de AWS 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 la configuración de extensiones KU y EKU, donde KU puede ser cualquiera de los nueve valores admitidos (DigitalSignature, NonRepudiation, KeyEncipherment, DataEncipherment, KeyAgreement keyCertSign, CRLSign, EnCipherOnly y DecipherOnly) y EKU puede ser cualquiera de los valores admitidos (ServeAuth, ClientAuth, codesign, EmailProtection, timestown camping y OCSPSigning) además de extensiones personalizadas.

BlankEndEntityCertificate_API Passthrough/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.

BlankEndEntityCertificateDefinición de _APICSRPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankEndEntityCertificate_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	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_ _Definición de APICSRPassThrough/v1 CriticalBasicConstraints

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _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: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 _Definición de apiPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _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: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 _definición de CSRPassthrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _CSRPassthrough/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: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.

BlankEndEntityCertificateDefinición de _CSRPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankEndEntityCertificate_CSRPassthrough/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]
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.

BlankSubordinateDefinición de CACertificate_ 0_CSRPassThrough/v1 PathLen

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 0_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_ 0_Definición de APICSRPassThrough/v1 PathLen

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 0_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.

BlankSubordinateDefinición de CACertificate_ PathLen 0_APIPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 0_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]

BlankSubordinateDefinición de CACertificate_ PathLen 1_APIPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 1_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.

BlankSubordinateDefinición de CACertificate_ PathLen 1_CSRPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 1_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_ 1_Definición de APICSRPassThrough/v1 PathLen

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 1_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.

BlankSubordinateDefinición de CACertificate_ PathLen 2_APIPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 2_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.

BlankSubordinateDefinición de CACertificate_ PathLen 2_CSRPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 2_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_ 2_Definición de PathLen APICSRPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 2_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.

BlankSubordinateDefinición de CACertificate_ PathLen 3_API PassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 3_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.

BlankSubordinateDefinición de CACertificate_ PathLen 3_CSRPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 3_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_ 3_Definición de PathLen APICSRPassThrough/v1

Para obtener información general sobre las plantillas en blanco, consulte BlankEndEntityCertificateDefinición de _apiPassThrough/v1.

BlankSubordinateCACertificate_ PathLen 3_API CSR PassThrough

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 desde Autoridad de certificación privada de AWS con cualquier solución de firma de código basada en una infraestructura de entidad de certificación privada. Por ejemplo, los clientes que utilizan la Firma de código para AWS IoT pueden generar un certificado de firma de código con Autoridad de certificación privada de AWS e importarlo a AWS Certificate Manager. Para obtener más información, consulte ¿Para qué sirve la firma de AWS IoT código? 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 claves	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.

CodeSigningCertificateDefinición de _apicsrPassThrough/v1

Esta plantilla amplía CodeSigningCertificate /V1 para admitir los valores de transferencia de API y CSR.

CodeSigningCertificate_API CSR Passthrough/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 claves	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.

CodeSigningCertificateDefinición de _apiPassThrough/v1

Esta plantilla es idéntica a la plantilla `CodeSigningCertificate` con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales de la solicitud de la API al certificado si las extensiones no se especifican 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 claves	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.

CodeSigningCertificateDefinición de _csrPassthrough/v1

Esta plantilla es idéntica a la plantilla `CodeSigningCertificate` con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no se especifican en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

CodeSigningCertificate_CSRPassthrough/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]
Identificador de clave de asunto	[Derivado de CSR]
Uso de claves	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 claves	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.

EndEntityCertificateDefinición de _apicsrPassThrough/v1

Esta plantilla amplía EndEntityCertificate /V1 para admitir los valores de transferencia de API y CSR.

EndEntityCertificate_API CSR Passthrough/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 claves	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.

EndEntityCertificateDefinición de _apiPassThrough/v1

Esta plantilla es idéntica a la plantilla `EndEntityCertificate` con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales de la solicitud de la API al certificado si las extensiones no se especifican 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 claves	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.

EndEntityCertificateDefinición de _csrPassthrough/v1

Esta plantilla es idéntica a la plantilla `EndEntityCertificate` con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no se especifican en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

EndEntityCertificate_CSRPassthrough/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]
Uso de claves	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 claves	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.

EndEntityClientAuthCertificateDefinición de _apicsrPassThrough/v1

Esta plantilla amplía EndEntityClientAuthCertificate /V1 para admitir los valores de transferencia de API y CSR.

EndEntityClientAuthCertificate_API CSR Passthrough/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 claves	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.

EndEntityClientAuthCertificateDefinición de _apiPassThrough/v1

Esta plantilla es idéntica a la plantilla `EndEntityClientAuthCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales mediante la API al certificado si las extensiones no se especifican 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 claves	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.

EndEntityClientAuthCertificateDefinición de _csrPassthrough/v1

Esta plantilla es idéntica a la plantilla `EndEntityClientAuthCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no se especifican en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

EndEntityClientAuthCertificate_CSRPassthrough/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]
Uso de claves	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 claves	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.

EndEntityServerAuthCertificateDefinición de _apicsrPassThrough/v1

Esta plantilla amplía EndEntityServerAuthCertificate /V1 para admitir los valores de transferencia de API y CSR.

EndEntityServerAuthCertificate_API CSR Passthrough/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 claves	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.

EndEntityServerAuthCertificateDefinición de _apiPassThrough/v1

Esta plantilla es idéntica a la plantilla `EndEntityServerAuthCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales mediante la API al certificado si las extensiones no se especifican 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 claves	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.

EndEntityServerAuthCertificateDefinición de _csrPassthrough/v1

Esta plantilla es idéntica a la plantilla `EndEntityServerAuthCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no se especifican en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

EndEntityServerAuthCertificate_CSRPassthrough/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]
Uso de claves	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.

Definición de OCSP/V1 SigningCertificate

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.

OCSP /V1 SigningCertificate

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 claves	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.

Definición de OCSP _APICSRPassThrough/v1 SigningCertificate

Esta plantilla amplía el OCSP /V1 para admitir los valores de transferencia de API SigningCertificate y CSR.

OCSP _APICSRPassThrough/v1 SigningCertificate

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 claves	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 OCSP SigningCertificate _APIPassThrough/v1

Esta plantilla es idéntica a la plantilla `OCSPSigningCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales mediante la API al certificado si las extensiones no se especifican en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la API.

OCSP SigningCertificate _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 claves	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.

Definición de OCSP SigningCertificate _CSRPassThrough/v1

Esta plantilla es idéntica a la plantilla `OCSPSigningCertificate` con una diferencia. En esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no se especifican en la plantilla. Las extensiones especificadas en la plantilla siempre anulan las extensiones en la CSR.

OCSP SigningCertificate _CSRPassThrough/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]
Identificador de clave de asunto	[Derivado de CSR]
Uso de claves	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 RootCACertificate/V1

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.

RootCACertificate/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 claves	Firma digital crítica keyCertSign, signo CRL
Punto de distribución de CRL	N/A

Definición de RootCACertificate_APIPassthrough/V1

Esta plantilla amplía RootCACertificate/V1 para admitir los valores de acceso directo de la API.

RootCACertificate_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`
Identificador de clave de entidad	[Acceso directo desde la API]
Identificador de clave de asunto	[Derivado de CSR]
Uso de claves	Firma digital crítica keyCertSign, signo CRL
Punto de distribución de CRL*	N/A

BlankRootDefinición de CAcertificate_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_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`
Identificador de clave de asunto	[Derivado de CSR]

BlankRootDefinición de CACertificate_ PathLen 0_APIPassthrough/v1

Para obtener información general sobre las plantillas de CA raíz vacías, consulte. ???

BlankRootCACertificate_ 0_APIPassthrough/v1 PathLen

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]

BlankRootDefinición de CACertificate_ PathLen 1_APIPassThrough/v1

Para obtener información general sobre las plantillas de CA raíz vacías, consulte. ???

BlankRootCACertificate_ 1_APIPassthrough/v1 PathLen

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]

BlankRootDefinición de CACertificate_ PathLen 2_APIPassThrough/v1

Para obtener información general sobre las plantillas de CA raíz vacías, consulte. ???

BlankRootCAcertificate_ 2_APIPassthrough/v1 PathLen

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]

BlankRootDefinición de CACertificate_ PathLen 3_APIPassThrough/v1

Para obtener información general sobre las plantillas de CA raíz en blanco, consulte. ???

BlankRootCAcertificate_ 3_APIPassthrough/v1 PathLen

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 Subordinate_ACertificate_ 0/V1 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.

Certificado CA subordinado_ 0/V1 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 claves	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.

Certificado CA subordinado_ 0_Definición de APICSRPassThrough/v1 PathLen

Esta plantilla amplía SubordinateCAcertificate_ 0/V1 para admitir los valores de transferencia de API y CSR. PathLen

Certificado PathLen CA subordinado_ 0_APICSR PassThrough/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]
Uso de claves	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 certificado CA subordinado _ 0_API PassThrough/v1 PathLen

Esta plantilla amplía SubordinateACertificate_ 0/V1 para admitir los valores de transferencia de la API. PathLen

Certificado PathLen CA subordinado_ 0_APIPassthrough/v1

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 claves	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 certificado CA subordinado_ 0_CSRPassthrough/v1 PathLen

Esta plantilla es idéntica a la plantilla `SubordinateCACertificate_PathLen0` con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no se especifican 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.

Certificado CA subordinado_ 0_CSRPassthrough/v1 PathLen

Parámetro X509v3	Valor
Nombre alternativo de asunto	[Acceso directo desde CSR]
Asunto	[Acceso directo desde CSR]

Parámetro X509v3	Valor
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 claves	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 certificado CA subordinado_ 1/V1 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.

Certificado CA subordinado_ 1/V1 PathLen

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]

Parámetro X509v3	Valor
Identificador de clave de asunto	[Derivado de CSR]
Uso de claves	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.

Certificado CA subordinado_ 1_Definición de APICSRPassThrough/v1 PathLen

Esta plantilla amplía SubordinateCAcertificate_ 1/V1 para admitir los valores de transferencia de API y CSR. PathLen

Certificado PathLen CA subordinado_ 1_APICSR PassThrough/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]
Uso de claves	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 certificado CA subordinado _ 1_API PassThrough/v1 PathLen

Esta plantilla amplía SubordinateACertificate_ 0/V1 para admitir los valores de transferencia de la API. PathLen

Certificado PathLen CA subordinado_ 1_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]
Uso de claves	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 certificado CA subordinado_ 1_CSRPassThrough/v1 PathLen

Esta plantilla es idéntica a la plantilla `SubordinateCACertificate_PathLen1` con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no se especifican 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.

Certificado CA subordinado_ 1_CSRPassThrough/v1 PathLen

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 claves	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 del certificado CA subordinado_ 2/V1 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.

Certificado CA subordinado_ 2/V1 PathLen

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: 2`
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de claves	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.

Certificado CA subordinado_ 2_Definición de APICSRPassThrough/v1 PathLen

Esta plantilla amplía SubordinateCAcertificate_ 2/V1 para admitir los valores de transferencia de API y CSR. PathLen

Certificado PathLen CA subordinado_ 2_APICSR PassThrough/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]
Uso de claves	Crítico, firma digital, `keyCertSign` , firma de CRL

Parámetro X509v3	Valor
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 Subordinate_2_API PassThrough/v1 PathLen

Esta plantilla amplía SubordinateACertificate_ 2/V1 para admitir los valores de transferencia de la API. PathLen

Certificado PathLen CA subordinado_ 2_API Passthrough/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]
Uso de claves	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 certificado CA subordinado_ 2_CSRPassThrough/v1 PathLen

Esta plantilla es idéntica a la plantilla `SubordinateCACertificate_PathLen2` con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales

de la solicitud de firma de certificados (CSR) al certificado si las extensiones no se especifican 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.

Certificado CA subordinado_ 2_CSRPassThrough/v1 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 claves	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 del certificado CA subordinado_ 3/V1 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.

Certificado CA subordinado_ 3/V1 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 claves	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.

Certificado CA subordinado_ 3_Definición de APICSRPassThrough/v1 PathLen

Esta plantilla amplía SubordinateCAcertificate_ 3/V1 para admitir los valores de transferencia de API y CSR. PathLen

Certificado PathLen CA subordinado_ 3_API CSR PassThrough/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`

Parámetro X509v3	Valor
Identificador de clave de entidad	[SKI del certificado de CA]
Identificador de clave de asunto	[Derivado de CSR]
Uso de claves	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 Subordinate_3_API PassThrough/v1 PathLen

Esta plantilla amplía SubordinateACertificate_ 3/V1 para admitir los valores de transferencia de la API. PathLen

Certificado PathLen CA subordinado_ 3_API Passthrough/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]
Identificador de clave de asunto	[Derivado de CSR]
Uso de claves	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 certificado CA subordinado_ 3_CSRPassThrough/v1 PathLen

Esta plantilla es idéntica a la plantilla `SubordinateCACertificate_PathLen3` con una diferencia: en esta plantilla, Autoridad de certificación privada de AWS pasa extensiones adicionales de la solicitud de firma de certificados (CSR) al certificado si las extensiones no se especifican 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.

Certificado CA subordinado_ 3_CSRPassThrough/v1 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 claves	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.

Uso de la API de Autoridad de certificación privada de AWS (ejemplos de Java)

Puede utilizar la API de AWS Private Certificate Authority para interactuar con el servicio mediante programación enviando 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 de HTTP, puede utilizar las herramientas de línea de comandos y los SDK de AWS para interactuar con 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 for 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 for Java en el blog para AWSdesarrolladores.

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 las CA y los certificados con nombres de asunto personalizados
- Cree certificados con extensiones personalizadas

Crear y activar una CA raíz mediante programación

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

- 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 API Autoridad de certificación privada de AWS:

- 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

En el siguiente ejemplo de Java, se 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
- 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 almacén Enterprise NTauth de Microsoft Active Directory (AD).

La operación crea una entidad de certificación (CA) de raíz privada mediante identificadores de objeto (OID) personalizados. Para obtener más información y un ejemplo de AWS CLI 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 de una entidad de certificación privada a una entidad principal de servicio de AWS designada. 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 director 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.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 AWS 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;
        }
    }
}
```

DescribeCertificateAuthority

En el siguiente ejemplo de Java, se 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 la 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 el certificado de entidad de certificación privada firmado en 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 Descripción de las plantillas de certificados.

```
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

En el siguiente ejemplo de Java, se muestra cómo se utiliza la operación ListCertificateAuthorities.

Esta operación muestra las entidades de certificación (CA) privadas que se crearon utilizando la operación CreateCertificateAuthority.

```
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 `GetCertificateListPermissions`, y, 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 marcas que se utilizan para identificar y organizar las CA. 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 lo autorice una política, una entidad principal que resida en otra cuenta de AWS puede emitir y renovar certificados de entidad final privada mediante una CA 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 habilita una lista de revocación de certificados (Certificate Revocation List, CRL) al crear o actualizar la CA privada, se incluirá información sobre los certificados revocados en la CRL. Autoridad de certificación privada de AWS escribirá la CRL en el bucket 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

El siguiente ejemplo de Java muestra cómo utilizar la TagCertificateAuthorityoperación.

Esta operación agrega una o varias etiquetas a la entidad de certificación privada. Las etiquetas son marcas que se utilizan para identificar y organizar los recursos de AWS. 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 por una relación común entre esas entidades de certificación, puede aplicar la misma etiqueta a varias entidades de certificación privadas. 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 las CA y los certificados con nombres de asunto personalizados

El objeto CustomAttribute permite a los administradores pasar identificadores de objeto (Object Identifier, OID) personalizados a los certificados y CA privadas. Los OID personalizados se pueden utilizar para crear jerarquías de nombres de sujetos especializadas 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 Variedades de plantillas. Para obtener más información sobre el uso de atributos personalizados, consulte Emisión de certificados de entidad final privadas y Procedimiento para crear una CA (CLI) .

No se puede utilizar `StandardAttributes` con `CustomAttributes`. Sin embargo, puede pasar los OID estándares como parte de un `CustomAttributes`. Los OID de nombre de asunto predeterminados se enumeran 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

Nombre del asunto	ID de objeto
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 una 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 objeto CustomExtension permite a los administradores configurar extensiones X.509 personalizadas en certificados privados. Los certificados personalizados se deben crear con una de las plantillas ApiPassthrough. Para obtener más información acerca de las plantillas, consulte Variedades de plantillas. Para obtener más información acerca de las extensiones personalizadas, consulte Emisión de certificados de entidad final privadas.

Temas

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

Activar 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);
    }
}
```

Uso de la API Autoridad de certificación privada de AWS para implementar el estándar Matter (ejemplos de Java)

Puede usar la API AWS Private Certificate Authority para crear certificados que se ajusten al 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.

Los ejemplos de Java de esta sección interactúan con el servicio al enviar 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 de HTTP, puede utilizar las herramientas de línea de comandos y los SDK de AWS para interactuar con 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 for Java para programar la API de Autoridad de certificación privada de AWS.

Tanto el equipo de GetCertificateAuthorityCsrDescribeCertificateAuthorityAuditReportoperaciones como GetCertificateel de apoyo 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 for Java en el blog para AWSdesarrolladores.

Matter 1.2, publicado en octubre de 2023, admite la revocación del DAC mediante listas de revocación de certificados (CRL). Para ayudarle a cumplir con el estándar Matter actual, al activar la revocación de la CRL para las CA que emiten certificados Matter, en el `CrlConfiguration` objeto, en la estructura, establecido en. `CrlDistributionPointExtensionConfiguration` `OmitExtension` `true`

Por lo general, las CA incorporan 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 obtienen los CDP del registro de cumplimiento distribuido de Matter (DCL), el almacén de datos de Matter de confianza. Debe cargar el URI del CDP en la DCL de Matter para poder detectarlo al validar los DAC. Para obtener más información

sobre cómo determinar el URI del CDP, consulte. Determinar el URI del punto de distribución CRL (CDP) Para obtener más información sobre Matter, consulte la documentación de Matter DCL.

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 RootCACertificate_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 a través de a. CustomExtension

El ejemplo llama a las siguientes acciones de API de Autoridad de certificación privada de AWS:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Si tiene problemas, consulte Uso del estándar 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 BlankSubordinateDefinición de CACertificate_ PathLen 0_APIPassThrough/v1 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 de Autoridad de certificación privada de AWS:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr

- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate
- GetCertificateAuthorityCertificate

Si tiene problemas, consulte Uso del estándar 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 valor codificado en KeyUsage Base64 y pasarlo por un. CustomExtension

El ejemplo llama a la siguiente acción de API de Autoridad de certificación privada de AWS:

- IssueCertificate

Si tiene problemas, consulte Uso del estándar 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 RootCACertificate_APIPassthrough/V1 plantilla para crear e instalar un certificado de CA Matter Root para emitir 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 de Autoridad de certificación privada de AWS:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Si tiene problemas, consulte Uso del estándar 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 BlankSubordinateDefinición de CACertificate_ PathLen 0_APIPassThrough/v1 plantilla para emitir e instalar un certificado de CA de Matter Subordinate para emitir NOCs. Debe generar un KeyUsage valor codificado en Base64 y pasarlo por un. CustomExtension

El ejemplo llama a las siguientes acciones de API de Autoridad de certificación privada de AWS:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr

- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate
- GetCertificateAuthorityCertificate

Si se producen problemas, consulte la sección de solución Uso del estándar Matter 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 valor codificado en Base64 y pasarlo por KeyUsage un. CustomExtension

El ejemplo llama a la siguiente acción de API de Autoridad de certificación privada de AWS:

- IssueCertificate

Si tiene problemas, consulte Uso del estándar 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);
   }
}
```

Uso de la Autoridad de certificación privada de AWS API para implementar el estándar de licencias de conducir móviles (mDL) (ejemplos de Java)

Puede usar 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.

Los ejemplos de Java de esta sección interactúan con el servicio al enviar 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, también puede utilizar los AWS SDK y AWS CLI las herramientas de administraciónAutoridad de certificación privada de AWS. Te recomendamos usar el SDK o AWS CLI la API HTTP. Para obtener más información, consulte Herramientas para Amazon Web Services. En los temas siguientes se muestra cómo utilizar AWS SDK for Java para programar la API de Autoridad de certificación privada de AWS.

Las `DescribeCertificateAuthorityAuditReport`operaciones `GetCertificateAuthorityCsrGetCertificate`, y dan soporte 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 y Waiters in the AWS SDK for Java en el blog para AWS desarrolladores.

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 BlankRootDefinición de CACertificate_ PathLen 0_APIPassthrough/v1 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 API de Autoridad de certificación privada de AWS:

- 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 BlankEndEntityCertificate_APIPassthrough/V1 para crear un certificado de firma de documentos que cumpla con la norma mDL de la ISO/IEC. Debe generar valores codificados en base64 para y transferirlos. `KeyUsage` `IssuerAlternativeName` `CRLDistributionPoint` `CustomExtensions`

El ejemplo llama a la siguiente acción de API de 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);
    }
}
```

Certificados de CA privados firmados externamente

Si la raíz de confianza de su jerarquía de entidad de certificación privada debe ser una entidad de certificación fuera de Autoridad de certificación privada de AWS, puede crear y autofirmar su propia entidad de certificación raíz. Alternativamente, puede obtener un certificado de entidad de certificación privada firmado por una entidad de certificación privada externa operada por su organización. Independientemente de la fuente, utilice esta entidad de certificación obtenida externamente para firmar un certificado de entidad de certificación subordinado privado que administre Autoridad de certificación privada de AWS.

> **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 con 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 AWS Management Console o el AWS CLI.

Para obtener e instalar un certificado de CA firmado externamente (consola)

1. (Opcional) Si aún no se encuentra en la página de detalles de la CA, abra la consola de Autoridad de certificación privada de AWS en https://console.aws.amazon.com/acm-pca/home.

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**
   >
   > Al copiar y pegar, se debe conservar el formato exacto del texto de la CSR.

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 de certificado (CSR) de su CA privada. Si desea enviar la CSR a la pantalla, utilice la opción `--output text` para eliminar los caracteres CR/LF al 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.

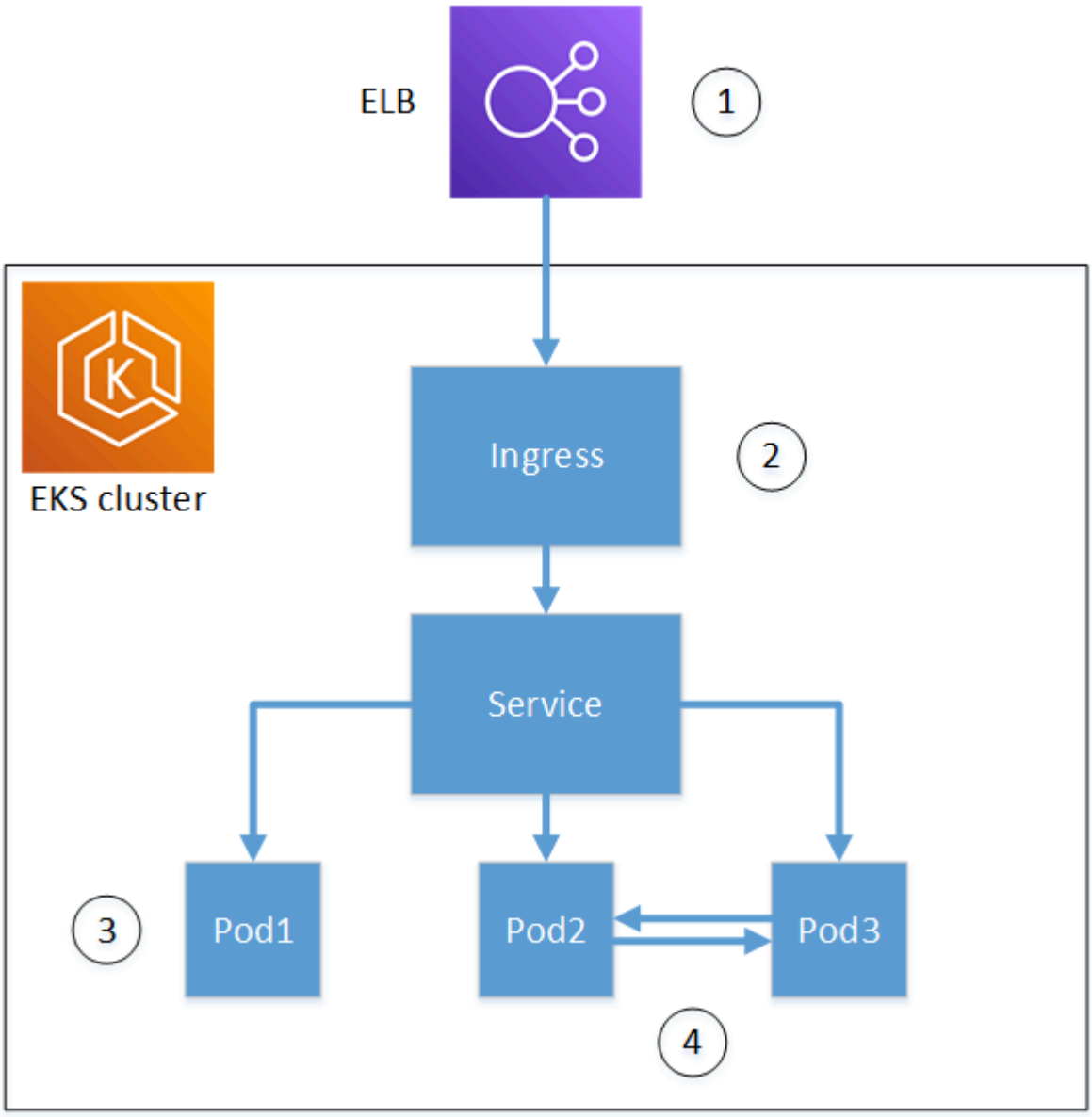
```
$ aws acm-pca import-certificate-authority-certificate \
--certificate-authority-arn arn:aws:acm-pca:region:account:\
certificate-authority/12345678-1234-1234-1234-123456789012 \
--certificate file://C:\example_ca_cert.pem \
--certificate-chain file://C:\example_ca_cert_chain.pem
```

Protección de Kubernetes con Autoridad de certificación privada de AWS

Los contenedores y las aplicaciones de Kubernetes utilizan certificados digitales para proporcionar una autenticación y un cifrado seguros mediante TLS. Una solución ampliamente adoptada para la gestión del ciclo de vida de los certificados TLS en Kubernetes es cert-manager, un complemento de Kubernetes que solicita certificados, los distribuye a los contenedores de Kubernetes y automatiza la renovación de los certificados.

Autoridad de certificación privada de AWSproporciona un complemento de código abierto para cert-manager, aws-privateca-issuer, para los usuarios de cert-manager que desean configurar una CA sin almacenar claves privadas en el clúster. Los usuarios con requisitos normativos para controlar el acceso a sus operaciones de CA y auditarlas pueden utilizar esta solución para mejorar la auditabilidad y respaldar el cumplimiento. Puede usar el complemento AWS Private CA Issuer con Amazon Elastic Kubernetes Service (Amazon EKS), un Kubernetes autogestionado en AWS o en un Kubernetes en las instalaciones.

En el siguiente diagrama, se muestran algunas de las opciones disponibles para usar TLS en un clúster de Amazon EKS. Este clúster de ejemplo, que contiene varios recursos, está situado detrás de un equilibrador de carga. Los números identifican los posibles puntos de conexión de las comunicaciones protegidas por TLS, como el equilibrador de carga externo, el controlador de entrada, un pod individual dentro de un servicio y un par de pods que se comunican de forma segura entre sí.



1. Terminar conexiones en el equilibrador de carga.

   Elastic Load Balancing (ELB) es un servicio integrado de AWS Certificate Manager, lo que significa que puede aprovisionar ACM con una CA privada, firmar un certificado con él e instalarlo mediante la consola de ELB. Esta solución proporciona una comunicación cifrada entre un cliente remoto y el equilibrador de carga. Los datos se transmiten sin cifrar al clúster de EKS. Como alternativa, puede proporcionar un certificado privado a un equilibrador de carga que no sea de AWS para finalizar TLS.

2. Terminación en el controlador de entrada de Kubernetes.

El controlador de entrada reside dentro del clúster de EKS como equilibrador de carga y enrutador nativos. Si ha instalado cert-manager y ha aprovisionado el clúster con una CA privada aws-privateca-issuer, Kubernetes puede instalar un certificado TLS firmado en el controlador, lo que le permitirá servir como punto final del clúster para las comunicaciones externas. Las comunicaciones entre el equilibrador de carga y el controlador de entrada están cifradas y, tras la entrada, 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 has instalado un cert-manager y has aprovisionado el clúster con una CA privada aws-privateca-issuer, Kubernetes puede instalar un certificado TLS firmado en los pods según sea necesario. De forma predeterminada, una conexión TLS que termina en un pod no está disponible en otros pods del clúster.

4. Proteja las comunicaciones entre los pods.

   También puede aprovisionar varios pods con certificados que les permitan comunicarse entre sí. Los siguientes escenarios son posibles:

   - Aprovisionamiento con certificados firmados automáticamente y generados por Kubernetes. Esto protege las comunicaciones entre los pods, pero los certificados firmados automáticamente no cumplen con los requisitos de la HIPAA ni del FIPS.
   - Aprovisionamiento con certificados firmados por una CA privada. Como se indica en los números 2 y 3 anteriores, para ello es necesario instalar un cert-manager y aws-privateca-issueraprovisionar el clúster con una CA privada. Luego, Kubernetes puede instalar certificados TLS firmados en los pods según sea necesario.

Uso de cert-manager entre cuentas

Los administradores con acceso entre cuentas a una CA pueden usar cert-manager para aprovisionar un clúster de Kubernetes. Para obtener más información, consulte Prácticas recomendadas de seguridad para el acceso entre cuentas a CA privadas.

> **Note**
>
> Solo se pueden usar determinadas plantillas de certificados de Autoridad de certificación privada de AWS en escenarios con varias cuentas. Para ver la lista de plantillas disponibles, consulte the section called “Plantillas de certificados compatibles ”.

Plantillas de certificados compatibles

En la siguiente tabla se enumeran las plantillas de Autoridad de certificación privada de AWS que se pueden usar con cert-manager para aprovisionar un clúster de Kubernetes.

Plantillas compatibles con Kubernetes	Compatible con la modalidad entre cuentas
BlankEndEntityCertificateDefinición de _CSRPassThrough/v1	
CodeSigningCertificateDefinición de /V1	
EndEntityCertificateDefinición de /V1	✓
EndEntityClientAuthCertificateDefinición de /V1	✓
EndEntityServerAuthCertificateDefinición de /V1	✓
Definición de OCSP/V1 SigningCertificate	

Soluciones de ejemplo

Las siguientes soluciones de integración muestran cómo configurar el acceso a Autoridad de certificación privada de AWS en un clúster de Amazon EKS.

- 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

Conector Autoridad de certificación privada de AWS para Active Directory

¿Qué es el Conector AWS Private CA para Active Directory?

AWS Private CA puede emitir y administrar los certificados requeridos por AWS Managed Microsoft AD. Con el Conector Autoridad de certificación privada de AWS para Active Directory (Conector para AD), puede sustituir las CA empresariales en las instalaciones o de terceros 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 Conector para AD con AWS Managed Microsoft AD para eliminar la infraestructura en las instalaciones migrando su infraestructura de AD y de clave pública a la nube. Para los clientes que desean utilizar AWS Private CA con su AD en las instalaciones, esta característica también se integra con el Conector AWS Managed Microsoft AD.

Temas

¿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 AWS Directory Service?

Acceder a Conector para AD

Puede acceder a Connector for AD a través de la consola y las API. AWS CLI Puedes acceder al conector de la consola desde la AWS Private CA consola, desde tu AWS Directory Service consola o buscando Connector for AD en la barra de AWS Management Console búsqueda.

Precio del Conector para AD

El Conector para AD se ofrece como una característica de Autoridad de certificación privada de AWS sin costo adicional. Solo paga por las entidades de certificación privadas y los certificados que emita a través de ellas.

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

Introducción a Conector AWS Private CA para Active Directory

Con el Conector AWS Private CA para Active Directory, puede emitir certificados desde su CA 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

Requisitos previos del Conector AD

Lo siguiente se solicita para el Conector para AD.

Para crear un Conector para AD, debe configurar una AWS Private Certificate Authority (CA) y un directorio. A continuación, debe compartir la CA privada y el directorio con el servicio del Conector para AD. También debe configurar los grupos de seguridad y las políticas de IAM correctamente para crear un conector.

Autoridad de certificación privada de AWS

Configure un Autoridad de certificación privada de AWS para emitir certificados para sus objetos de directorio. Para obtener más información, consulte Administración de CA privada.

Autoridad de certificación privada de AWS debe estar en el estado `Active` para 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.

Active Directory

Además de un Autoridad de certificación privada de AWS, necesita Active Directory en una nube privada virtual (VPC). El Conector para AD admite los siguientes tipos de directorios que ofrece AWS Directory Service:

- AWSMicrosoft Active Directory administrado: con AWS Directory Service él puede ejecutar Microsoft Active Directory (AD) como un servicio administrado. AWS Directory Service for Microsoft Active Directorytambién conocido comoAWS Managed Microsoft AD, funciona con Windows Server 2019. Con AWS Managed Microsoft AD, puede ejecutar cargas de trabajo compatibles con el directorio en la nube de Nube de AWS, incluidas aplicaciones de Microsoft SharePoint y aplicaciones .NET y basadas en SQL Server personalizadas.
- 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

> **Note**
>
> No se admite la inscripción de controladores de dominio cuando se utiliza el Conector para AD con AWS Managed Microsoft AD.

Cuenta de servicio

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 la Autoridad de AuthCertificates Certificación (CA) de NT. Nota: si el objeto AuthCertificates CA de NT 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
```

> **Note**
>
> Si utiliza AWS Managed Microsoft AD, los permisos adicionales se delegarán automáticamente cuando autorice el servicio de Conector para AD con su directorio. Puede omitir este paso de requisito previo.

Puede utilizar este PowerShell script para delegar los permisos adicionales. Creará el objeto de la AuthCertifiates entidad emisora de certificados de NT. Sustituya “myconnectoraccount” por el nombre de la cuenta de servicio.

```
$AccountName = 'myconnectoraccount'
# DO NOT modify anything below this comment.
# Getting Active Directory information.
Import-Module -Name 'ActiveDirectory'
$RootDSE = Get-ADRootDSE

# Getting AD Connector service account Information
$AccountProperties = Get-ADUser -Identity $AccountName
$AccountSid = New-Object -TypeName 'System.Security.Principal.SecurityIdentifier'
 $AccountProperties.SID.Value
```

```
[System.GUID]$ServicePrincipalNameGuid = (Get-ADObject -SearchBase
 $RootDse.SchemaNamingContext -Filter { lDAPDisplayName -eq 'servicePrincipalName' } -
Properties 'schemaIDGUID').schemaIDGUID
$AccountAclPath = $AccountProperties.DistinguishedName

# Getting ACL settings for AD Connector service account.
$AccountAcl = Get-ACL -Path "AD:\$AccountAclPath"

# Setting ACL allowing the AD Connector service account the ability to add and remove a
 Service Principal Name (SPN) to itself
$AccountAccessRule = New-Object -TypeName
 'System.DirectoryServices.ActiveDirectoryAccessRule' $AccountSid, 'WriteProperty',
 'Allow', $ServicePrincipalNameGuid, 'None'
$AccountAcl.AddAccessRule($AccountAccessRule)
Set-ACL -AclObject $AccountAcl -Path "AD:\$AccountAclPath"

# Add ACLs allowing AD Connector service account the ability to create certification
 authorities
[System.GUID]$CertificationAuthorityGuid = (Get-ADObject -SearchBase
 $RootDse.SchemaNamingContext -Filter { lDAPDisplayName -eq 'certificationAuthority' }
 -Properties 'schemaIDGUID').schemaIDGUID
$CAAccessRule = New-Object -TypeName
 'System.DirectoryServices.ActiveDirectoryAccessRule' $AccountSid,
 'ReadProperty,WriteProperty,CreateChild,DeleteChild', 'Allow',
 $CertificationAuthorityGuid, 'None'
$PKSDN = "CN=Public Key Services,CN=Services,CN=Configuration,
$($RootDSE.rootDomainNamingContext)"
$PKSACL = Get-ACL -Path "AD:\$PKSDN"
$PKSACL.AddAccessRule($CAAccessRule)
Set-ACL -AclObject $PKSACL -Path "AD:\$PKSDN"

$AIADN = "CN=AIA,CN=Public Key Services,CN=Services,CN=Configuration,
$($RootDSE.rootDomainNamingContext)"
$AIAACL = Get-ACL -Path "AD:\$AIADN"
$AIAACL.AddAccessRule($CAAccessRule)
Set-ACL -AclObject $AIAACL -Path "AD:\$AIADN"

$CertificationAuthoritiesDN = "CN=Certification Authorities,CN=Public Key
 Services,CN=Services,CN=Configuration,$($RootDSE.rootDomainNamingContext)"
$CertificationAuthoritiesACL = Get-ACL -Path "AD:\$CertificationAuthoritiesDN"
$CertificationAuthoritiesACL.AddAccessRule($CAAccessRule)
Set-ACL -AclObject $CertificationAuthoritiesACL -Path "AD:\$CertificationAuthoritiesDN"
```

```
$NTAuthCertificatesDN = "CN=NTAuthCertificates,CN=Public Key
 Services,CN=Services,CN=Configuration,$($RootDSE.rootDomainNamingContext)"
If (-Not (Test-Path -Path "AD:\$NTAuthCertificatesDN")) {
New-ADObject -Name 'NTAuthCertificates' -Type 'certificationAuthority' -OtherAttributes
 @{certificateRevocationList=[byte[]]'00';authorityRevocationList=[byte[]]'00';cACertificate=[b
 -Path "CN=Public Key Services,CN=Services,CN=Configuration,
$($RootDSE.rootDomainNamingContext)" }

$NTAuthCertificatesACL = Get-ACL -Path "AD:\$NTAuthCertificatesDN"
$NullGuid = [System.GUID]'00000000-0000-0000-0000-000000000000'
$NTAuthAccessRule = New-Object -TypeName
 'System.DirectoryServices.ActiveDirectoryAccessRule' $AccountSid,
 'ReadProperty,WriteProperty', 'Allow', $NullGuid, 'None'
$NTAuthCertificatesACL.AddAccessRule($NTAuthAccessRule)
Set-ACL -AclObject $NTAuthCertificatesACL -Path "AD:\$NTAuthCertificatesDN"
```

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:

```
{
    "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": {
                "StringLike": {
                    "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/*"
        }
    ]
}
```

El Conector para AD requiere permisos adicionales de AWS RAM, tanto para su uso en la consola como en la línea de comandos.

```
{
    "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": "*"
        }
    ]
}
```

Compartir Autoridad de certificación privada de AWS con el Conector para AD

Deberá compartir su AWS Private CA con el servicio de conectores mediante el uso compartido de la entidad principal del servicioAWS Resource Access Manager.

Al crear un conector en la AWS consola, el recurso compartido se crea automáticamente.

Cuando cree un recurso compartido mediante el AWS CLI, utilizará el comando AWS RAM create-resource-share.

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`.

Autorizar al Conector para AD con su 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 los registros de directorios

Grupos de seguridad

La comunicación entre la VPC y el conector del Conector para AD es a través de AWS PrivateLink, lo que requiere un grupo o grupos de seguridad con reglas de entrada que abran los puertos 443 TCP y UDP 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).

Crear un conector

Para obtener instrucciones, consulte el procedimiento Crear un conector

Configuración de las políticas de AD

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 Autoridad de certificación privada de AWS del cliente 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.

En Active Directory, siga los pasos que se indican a continuación 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.

Configure los GPO.

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 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.

Una vez que haya terminado de configurar los GPO y las políticas de claves públicas, los objetos del dominio solicitarán certificados al Conector Autoridad de certificación privada de AWS para AD y recibirán los certificados emitidos por Autoridad de certificación privada de AWS.

Crear una plantilla

Para obtener instrucciones, consulte el procedimiento Creación de una plantilla de conector

Administrar los permisos de grupo de AD

Para obtener instrucciones, consulte el procedimiento Administración de grupos de AD y permisos para plantillas

Procedimientos del Conector Autoridad de certificación privada de AWS para Active Directory

En los procedimientos de esta sección, se describe cómo crear conectores, configurar plantillas e integrarlos con Autoridad de certificación privada de AWS y Active Directory. Puede realizar estas operaciones desde la consola del Conector Autoridad de certificación privada de AWS para AD, mediante la sección Conector para AD de la AWS CLI o mediante la API del Conector de Autoridad de certificación privada de AWS para AD.

> **Note**
>
> Si bien el Conector Autoridad de certificación privada de AWS para AD está estrechamente integrado con Autoridad de certificación privada de AWS, los dos servicios tienen API independientes. Para obtener más información, consulte la Referencia de la AWS Private Certificate AuthorityAPI y la Referencia de la API de Autoridad de certificación privada de AWS Connector for Active Directory.

Procedimientos

- Crear un conector
- Creación de una plantilla de conector
- Enumerar conectores para Active Directory
- Enumeración de plantillas de conector
- Ver los detalles del conector

- Ver detalles de la plantilla de conectores
- Administrar los registros de directorios
- Administración de grupos de AD y permisos para plantillas
- Configuración del nombre de la entidad principal del servicio
- Etiquetado de los recursos del Conector para AD

Crear un conector

Utilice los siguientes procedimientos para crear un conector usando la consola, la línea de comandos o la API del Conector AWS Private CA para Active Directory.

Crear un conector (consola)

Use el siguiente procedimiento para crear y configurar un conector mediante la consola de AWS.

Tareas

- Abrir la consola
- Abrir Creación de conector
- Creación o selección de un directorio
- Seleccione Entidades de certificación privadas
- Configuración de etiquetas
- Revisar y crear

Abrir la consola

Inicie sesión en su cuenta de AWS y abra la consola del Conector AWS Private CA para Active Directory en **`https://console.aws.amazon.com/pca-connector-ad/home`**.

Abrir Creación de conector

En la página de inicio del primer servicio o en la página Conectores para Active Directory, elija Crear conector.

Creación o selección de un directorio

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 porAWS 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 consola de AWS Directory Service en una ventana nueva. Cuando termine de crear un directorio nuevo, vuelva a la consola del Conector AWS Private CA para 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 AWS Directory Service:
  >
  > - Microsoft AD administrado por AWS
  > - 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, puede elegir Crear grupo de seguridad, que abre la consola de Amazon EC2 en la página Crear grupo de seguridad en una ventana nueva. Cuando termine de crear un grupo de seguridad, vuelva a la consola del Conector AWS Private CA para Active Directory y actualice la lista de grupos de seguridad. El nuevo grupo de seguridad debería estar disponible para su selección.

Seleccione Entidades de certificación privadas

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 consola de Autoridad de certificación privada de AWS en la página de Autoridades de certificación privadas en una ventana nueva. Cuando termine de crear un CA, vuelva a la consola del Conector AWS Private CA para Active Directory y actualice la lista de CA. El nuevo CA debería estar disponible para su selección.

Configuración de etiquetas

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.

Revisar y crear

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.

Crear un conector para Active Directory (AWS CLI)

Para crear un conector para Active Directory con la CLI, utilice el comando create-connector de la sección Conector AWS Private CA para Active Directory de la AWS CLI.

Crear un conector para Active Directory (API)

Para crear un conector para Active Directory con la API, utilice la CreateConnectoracción de la API AWS Private CA Connector for Active Directory.

Creación de una plantilla de conector

Crear una plantilla de conector (consola)

Complete los siguientes procedimientos para crear y configurar un conector mediante la consola de AWS.

Tareas

- Abrir la consola
- Elegir conector
- Buscar sección de plantillas
- Método de creación de plantillas
- Configuración de plantillas
- Configure los ajustes del certificado
- Configure los ajustes de registro y gestión de solicitudes
- Configuración de las extensiones de uso clave
- Asignar políticas de aplicación
- Configurar políticas de aplicación personalizadas
- Configuración de los ajustes de criptografía
- Configurar grupos y permisos
- Configurar plantillas sustitutivas
- Configuración de etiquetas
- Revisar y crear

Abrir la consola

Inicie sesión en su cuenta de AWS y abra la consola del Conector AWS Private CA para Active Directory en **`https://console.aws.amazon.com/pca-connector-ad/home`**.

Elegir conector

Elija un conector de la lista de Conectores para Active Directory y, a continuación, seleccione Ver detalles.

Buscar sección de plantillas

En la página de detalles del conector, busque la sección Plantillas y, a continuación, elija Crear plantilla.

Método de creación de plantillas

En la página Crear plantilla, en la sección Método de creación de plantillas, elija una de las opciones del método.

- Comience 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
- Comience a partir de una plantilla existente que creó: elija de una lista de plantillas personalizadas que haya creado anteriormente.
- Empezar desde una plantilla en blanco: elija esta opción para empezar a crear una plantilla completamente nueva.

Configuración de plantillas

En la sección Configuración opcional, puede proporcionar la siguiente información opcional:

- Nombre de la plantilla: el nombre de la plantilla
- Versión del esquema de la plantilla: la versión del esquema de la plantilla. La versión del esquema afecta a la disponibilidad de las opciones de plantilla de la siguiente manera:

  Versión de esquema 2
  - Es compatible con Windows XP y Windows Server 2003 y versiones posteriores.
  - Solo es compatible con proveedores de servicios criptográficos heredados.

  Versión de esquema 3
  - Es compatible con Windows Vista y Windows Server 2008 y versiones posteriores.

- Permite que el solicitante renueve con la clave existente.
- Solo admite proveedores de almacenamiento de claves.

Versión de esquema 4

- Es compatible con Windows 8 y Windows Server 2012 y versiones posteriores.
- Permite que el solicitante renueve con la clave existente.
- Es compatible con los proveedores de servicios criptográficos heredados y los proveedores de almacenamiento de claves.

- Compatibilidad con los clientes: el nivel mínimo de sistema operativo compatible con la plantilla. Elija una de las siguientes opciones:
  - Windows XP y Windows Server 2003
  - Windows Vista y Windows Server 2008
  - Windows 7 y Windows Server 2008 R2
  - Windows 8 y versiones posteriores, y Windows Server 2012
  - Windows 8 y versiones posteriores, y Windows Server 2012 R2
  - Windows 8 y versiones posteriores, y Windows Server 2016 y versiones posteriores

Configure los ajustes del certificado

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)

Configure los ajustes de registro y gestión de solicitudes

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
- Encryption (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

Configuración de las extensiones de uso clave

En la sección de Configuración de la extensión de uso de claves, elija la opción para el uso de la firma y la clave de cifrado.

Uso de la clave de firma

- Firma digital
- La firma es una prueba de origen (no repudio)

Uso de claves de cifrado

- Permitir el intercambio de claves sin cifrado de claves (acuerdo de claves)
- Permitir el intercambio de claves únicamente con el cifrado de claves (cifrado de claves)
- Permitir el cifrado de los datos de los usuarios (cifrado de datos)

También puede optar por hacer que las extensiones de uso de claves sean fundamentales para ambos tipos de clave.

Asignar políticas de aplicación

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.

Configurar políticas de aplicación personalizadas

En la sección Políticas de aplicación personalizadas, puede agregar OID personalizados a la plantilla y especificar si las extensiones de las políticas de aplicación son críticas.

Configuración de los ajustes de criptografía

En la sección Configuración de criptografía, elija las siguientes categorías de configuración de criptografía para los certificados basados en esta plantilla.

1. El contenido de la parte superior de la sección viene determinado por la Método de creación de plantillas y Configuración de plantillas que haya elegido anteriormente.

   - Si ha aceptado la plantilla predeterminada en la versión 2Configuración de plantillas, aquí se muestran los siguientes mensajes de estado:
     - Categoría de proveedor de criptografía
     - Proveedor de servicios criptográficos heredados

     En este caso, no hay ajustes que configurar y puede pasar al paso siguiente.
   - Si ha aceptado la versión 3 de la plantillaConfiguración de plantillas, aquí se muestran los siguientes mensajes de estado:
     - Categoría de proveedor de criptografía
     - Proveedor de almacenamiento de claves

     También debe elegir un algoritmo clave de las opciones enumeradas ECDH_P256, ECDH_P384, ECDH_P521 y RSA.
   - Si especificó la versión 4 de la plantilla en Configuración de plantillas, debe elegir entre un proveedor de almacenamiento de claves y un proveedor de servicios criptográficos heredado. Si elige el suministro de almacenamiento de claves, también debe elegir un algoritmo clave de entre las opciones enumeradas ECDH_P256, ECDH_P384, ECDH_P521 y RSA.
2. Tamaño mínimo de clave (bits): especifique el tamaño mínimo de clave. Esta configuración afectará a los proveedores de criptografía disponibles.

3. Elija qué proveedores de cifrado se pueden usar para las solicitudes: elija una de las dos opciones disponibles:

   - Las solicitudes pueden utilizar cualquier proveedor disponible en la computadora del sujeto
   - Las solicitudes deben utilizar uno de los siguientes proveedores seleccionados

     Al seleccionar esta opción, se abre una lista de proveedores de criptografía. Puede seleccionar y priorizar los proveedores mediante los botones de la columna Pedido. Se admiten las siguientes opciones de proveedores:

     - Proveedor criptográfico base Microsoft v1.0
     - Proveedor criptográfico Microsoft Base DSS y Diffie-Hellman
     - Proveedor criptográfico de tarjeta inteligente base Microsoft
     - Proveedor criptográfico Microsoft DH SChannel
     - Proveedor criptográfico mejorado Microsoft v1.0
     - Proveedor criptográfico Diffie-Hellman y DSS mejorado de Microsoft
     - Proveedor criptográfico AES y RSA mejorado Microsoft
     - Proveedor criptográfico Microsoft RSA SChannel

Configurar grupos y permisos

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

Configurar plantillas sustitutivas

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.

Configuración de etiquetas

En el panel Etiquetas: opcional, puede aplicar y eliminar metadatos en tu 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.

Revisar y crear

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.

Creación de una plantilla de conector (CLI)

Utilice el comando create-template de la sección Conector AWS Private CA para Active Directory de AWS CLI.

Creación de una plantilla de conector (API)

Use la acción CreateTemplate en la API del Conector AWS Private CA para Active Directory.

Enumerar conectores para Active Directory

Puede usar la consola del Conector AWS Private CA para Active Directory o AWS CLI para enumerar los conectores que le pertenecen.

Para enumerar los conectores mediante la consola

1. Inicie sesión en su cuenta de AWS y abra la consola del Conector AWS Private CA para 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 | Activar | Eliminar | Fallido.
- Estado de registro en el directorio: estado de registro del director asociado. Los valores posibles son Crear | Activar | 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.

Para enumerar los conectores mediante la AWS CLI

Utilice el comando list-connectors para enumerar los conectores.

Para enumerar los conectores mediante la API

Use la acción ListConnectors en la API del Conector AWS Private CA para Active Directory.

Enumeración de plantillas de conector

Puede usar la consola del Conector AWS Private CA para Active Directory o AWS CLI para enumerar los conectores que le pertenecen. Las plantillas de conectores se basan en las plantillas AWS Private CA BlankEndentityCertificate_APIPassThrough/V1.

Para actualizar el oyente desde la consola

1. Inicie sesión en su cuenta de AWS y abra la consola del Conector AWS Private CA para 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.

Para enumerar las plantillas mediante AWS CLI

Utilice el comando list-templates para enumerar las plantillas del conector especificado.

Para enumerar las plantillas mediante la API

Enumerar: acción ListTemplates en la API del Conector AWS Private CA para Active Directory.

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 del Conector AWS Private CA para Active Directory.

Ver conector (consola)

Para ver los detalles de un conector (consola)

1. Inicie sesión en su cuenta de AWS y abra la consola del Conector AWS Private CA para 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
- asunto de Autoridad de certificación privada de AWS
- estado Autoridad de certificación privada de AWS
- 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.

Ver el conector (CLI)

Utilice el comando get-connector de la sección Conector AWS Private CA para Active Directory de AWS CLI.

Ver el conector (API)

Use la acción GetConnector en la API del Conector AWS Private CA para Active Directory.

Ver detalles de la plantilla de conectores

Utilice los siguientes procedimientos para ver los detalles de configuración de una plantilla de conector en la consola, la línea de comandos o la API del Conector AWS Private CA para Active Directory

Ver plantilla (consola)

Para ver los detalles de una plantilla de contenedor (consola)

1. Inicie sesión en su cuenta de AWS y abra la consola del Conector AWS Private CA para 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
   - Eliminar

- 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.

Para ver los detalles de una plantilla de conector (AWS CLI)

Ver plantilla (CLI)

Utilice el comando get-template de la sección Conector AWS Private CA para Active Directory de AWS CLI.

Ver plantilla (API)

Para ver los detalles de una plantilla de conector (API)

Use la acción GetTemplate en la API del Conector AWS Private CA para Active Directory.

Administrar los registros de directorios

Para administrar registros de directorios (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 cuenta de AWS y abra la consola del Conector AWS Private CA para 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: 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. Al elegir el enlace, accederá a la consola de AWS Directory Service.
   - 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 de la entidad principal del servicio (SPN), puede administrar los SPN del conector. Para obtener más información, consulte Ver detalles del conector.
7. En el panel Etiquetas: opcional, puede aplicar y eliminar metadatos en tu 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.

Para administrar registros de directorios (CLI)

Crear: utilice el comando create-directory-registration de la sección Conector AWS Private CA para Active Directory del AWS CLI.

Recuperar: utilice el comando get-directory-registration de la sección Conector AWS Private CA para Active Directory del AWS CLI.

Enumerar: utilice el comando list-directory-registrations de la sección Conector AWS Private CA para Active Directory del AWS CLI.

Eliminar: utilice el comando delete-directory-registration de la sección Conector AWS Private CA para Active Directory del AWS CLI.

Para administrar registros de directorios (API)

Crear: acción CreateDirectoryRegistration en la API del Conector AWS Private CA para Active Directory.

Recuperar: la acción GetDirectoryRegistration en la API del Conector AWS Private CA para Active Directory.

Enumerar: acción ListDirectoryRegistrations en la API del Conector AWS Private CA para Active Directory.

Eliminar: acción DeleteDirectoryRegistration en la API del Conector AWS Private CA para Active Directory.

Administración de grupos de AD y permisos para plantillas

Para administrar los grupos de plantillas y los permisos (consola)

Los grupos y permisos de una plantilla existente se pueden gestionar desde la página de detalles de la plantilla. Para obtener más información, consulte Ver detalles de la plantilla del conector.

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".

Busque el identificador de seguridad del grupo en Active Directory

Puede usar el siguiente script para buscar el identificador de seguridad del grupo en Active Directory.

```
$ Get-ADGroup -Identity "my_active_directory_group_name"
```

Para administrar los grupos de plantillas y los permisos (CLI)

Crear: el comando create-template-group-access-control-entry en la sección Conector AWS Private CA para Active Directory del AWS CLI.

Actualizar: el comando update-template-group-access-control-entry en la sección Conector AWS Private CA para Active Directory del AWS CLI.

Recuperar: el comando get-template-group-access-control-entry en la sección Conector AWS Private CA para Active Directory del AWS CLI.

Enumerar: el comando list-template-group-access-control-entries en la sección Conector AWS Private CA para Active Directory del AWS CLI.

Eliminar: el comando delete-template-group-access-control-entries en la sección Conector AWS Private CA para Active Directory del AWS CLI.

Para administrar los grupos de plantillas y los permisos (API)

Crear: acción CreateTemplateGroupAccessControlEntry en la API del Conector AWS Private CA para Active Directory.

Actualizar: el comando UpdateTemplateGroupAccessControlEntry en la API del Conector AWS Private CA para Active Directory.

Recuperar: la acción GetTemplateGroupAccessControlEntry en la API del Conector AWS Private CA para Active Directory.

Enumerar: acción ListTemplateGroupAccessControlEntries en la API del Conector AWS Private CA para Active Directory.

Eliminar: acción DeleteTemplateGroupAccessControlEntry en la API del Conector AWS Private CA para Active Directory.

Configuración del nombre de la entidad principal del servicio

Para administrar los nombres de las entidades principales del servicio (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

Para administrar los nombres de las entidades principales del servicio (consola)

Crear: el comando create-service-principal-name en la sección del Conector AWS Private CA para Active Directory de la sección de AWS CLI.

Recuperar: el comando get-service-principal-name en la sección del Conector AWS Private CA para Active Directory de la sección de AWS CLI.

Enumerar: el comando list-service-principal-names en la sección del Conector AWS Private CA para Active Directory de la sección de AWS CLI.

Eliminar: el comando delete-service-principal-name en la sección del Conector AWS Private CA para Active Directory de la sección de AWS CLI.

Para administrar los nombres de las entidades principales del servicio (API)

Crear: acción CreateServicePrincipalName en la API del Conector AWS Private CA para Active Directory.

Recuperar: la acción GetServicePrincipalName en la API del Conector AWS Private CA para Active Directory.

Enumerar: acción ListServicePrincipalNames en la API del Conector AWS Private CA para Active Directory.

Eliminar: acción DeleteServicePrincipalName en la API del Conector AWS Private CA para Active Directory.

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.

Para administrar el etiquetado de recursos (consola)

El etiquetado de los recursos existentes se gestiona en la página de detalles del recurso. Para obtener más información, consulte los siguientes procedimientos:

- Visualización de detalles de la plantilla de conectores
- Administrar los registros de directorios

Para administrar el etiquetado de recursos (CLI)

Etiqueta: comando tag-resource de la sección Conector AWS Private CA para Active Directory del AWS CLI.

Enumeración de etiquetas: comando list-tags-for-resource de la sección Conector AWS Private CA para Active Directory del AWS CLI.

Eliminación de etiqueta: comando untag-resource de la sección Conector AWS Private CA para Active Directory del AWS CLI.

Para administrar el etiquetado de recursos (API)

Etiqueta: acción TagResource en la API de Conector AWS Private CA para Active Directory.

Enumeración de etiquetas: acción ListTagsForResource en la API del Conector AWS Private CA para Active Directory.

Eliminación de etiqueta: acción UntagResource en la API de Conector AWS Private CA para 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.

Solución de problemas

Consulte los siguientes temas si tiene problemas cuando utiliza Autoridad de certificación privada de AWS.

Temas

Creación y firma de un certificado de entidad de certificación privada

Una vez que haya creado la CA privada, debe recuperar la CSR y enviarla a una CA raíz o intermedia de la infraestructura de X.509. La CA utiliza la CSR para crear el certificado de la CA privada, que firma antes de devolverlo.

Lamentablemente, no es posible proporcionar consejos concretos acerca de los problemas relacionados con la creación y la firma del certificado de la CA privada. Los detalles de la infraestructura de X.509 y su jerarquía de CA en ella exceden el alcance de esta documentación de Autoridad de certificación privada de AWS .

Latencia en las respuestas de 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.

Configuración de Amazon S3 para permitir la creación de un bucket de CRL

Es posible que su entidad de certificación privada no pueda crear un bucket de CRL si el Bloqueo de acceso público de Amazon S3 (configuración del bucket) se colocó en su cuenta. Compruebe su configuración de Amazon S3 si esto ocurre. Para obtener más información, consulte Uso del Bloqueo de acceso público de Amazon S3.

Revocar un certificado autofirmado de CA

No puede revocar un certificado de entidad de certificación autofirmado. En su lugar, debe eliminar la CA.

Tratamiento de excepciones

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	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

Excepción devuelta por Autoridad de certificación privada de AWS	Descripción	Corrección
		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 privada asociada al certificado designado ha alcanzado su cuota. Para obtener más información, consulte Service Quotas en la Guía de Referencia general de AWS .	Póngase en contacto con el AWS Support Centro para solicitar un aumento de cuota.
`MalformedCSRException`	La solicitud de firma de certificados (CSR) que se envió a Autoridad de certificación privada de AWS no se puede verificar ni validar.	Confirme que su CSR se generó y configuró correctamente.

Excepción devuelta por Autoridad de certificación privada de AWS	Descripción	Corrección
`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.
`RequestFailedException`	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).
`ResourceNotFoundException`	La entidad de certificación privada que emitió el certificado 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	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	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 Actualización de su entidad de certificación privada.

Uso del estándar 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 Uso de la API Autoridad de certificación privada de AWS para implementar el estándar Matter (ejemplos de Java).

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` deben marcarse como críticas.	Asegúrese de haber seleccionado la plantilla correcta para su c uso.

Código de error	Significado	Corrección
0x000000 0	La extensión del identificador de clave de autoridad debe estar presente.	Autoridad de certificación privada de AWS no establece la exte del identificador de la clave de autoridad en los certificados raíz generar un AuthorityKeyIdentifier valor codificado en Base64 m la CSR y, a continuación, pasarlo por un. CustomExtension Par obtener más información, consulte Active una CA raíz para los dos operativos de nodo (NOC). y Activar una autoridad de certi de productos (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, consulte 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.

Conector para errores y fallos de AD

Utilice esta información como ayuda para diagnosticar y corregir errores y errores de creación cuando trabaje con Connector for AD.

Temas

- Errores
- Fallos en la creación del conector
- Fallos al crear el SPN

Errores

Connector for AD envía mensajes de error por varios motivos. Para obtener información sobre cada error y recomendaciones para resolverlos, consulte la tabla siguiente. 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	Significado	Corrección
0x8FFFA000	Error en la autenticación de Kerberos.	Si utiliza la inscripción automátic a, corrija la entidad principal de servicio de recursos de AWS . Si utiliza la interfaz de usuario de Active Directory para obtener un certificado, ejecute `gpupdate /force` .
0x8FFFA001		Agregue una acción de encabezado.

Código de error	Significado	Corrección
	El mensaje SOAP debe contener un encabezado de acción.	
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 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 directori o. 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 `dnsHostNa me` a su objeto de AD.

Código de error	Significado	Corrección
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
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.

Código de error	Significado	Corrección
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.
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` .

Código de error	Significado	Corrección
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.
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.

Código de error	Significado	Corrección
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.
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.

Fallos en la creación del conector

La creación del conector puede fallar por varios motivos. Si se produce un error en la creación del conector, recibirá 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.	Debe conceder a Connector for AD el acceso a su directorio. Revise la 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.
`INTERNAL_FAILURE`	El conector para AD ha sufrido un error interno.	Inténtelo de nuevo más tarde. Elimine el conector defectuoso y cree uno nuevo.

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 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 fallido 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 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 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.

Fallos al crear el SPN

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 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	El conector para AD no encuentra el directorio especificado.	Asegúrese de especificar el identificador de directorio correcto y, a continuación, elimine el conector defectuos

Estado del fallo	Descripción	Corrección
		o 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 ha alcanzado el límite de SPN por directorio. El número máximo de SPN por directorio es 10.	Elimine uno o más SPN de su cuenta e intente crear el SPN de nuevo.

Error al crear un conector para AD

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.

Términos y conceptos

Los siguientes términos y conceptos pueden ayudarlo mientras trabaja con AWS Private Certificate Authority (Autoridad de certificación privada de AWS).

Temas

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 hash criptográfica como 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 utilizar Autoridad de certificación privada de AWS para crear una entidad de certificación privada y utilizarla para emitir certificados. La CA privada solamente genera certificados SSL/TLS para utilizarlos dentro de la 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 sólo las entidades de certificación en las 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 utilizar Autoridad de certificación privada de AWS para crear una entidad de certificación raíz privada o una entidad de certificación subordinada privada, cada una respaldada por un certificado de entidad de certificación. 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 más habituales de un certificado de entidad de certificación X.509 de Autoridad de certificación privada de AWS. 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

Una entidad de certificación (CA) normalmente existe dentro de una estructura jerárquica que contiene otras muchas CA con relaciones principal-secundario claramente definidas entre ellas. Las CA secundarias o subordinadas están certificadas por su CA principal, 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 de «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

Los certificados de Autoridad de certificación privada de AWS son certificados SSL/TLS privados que se pueden utilizar en la organización, pero no son confiables 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 ninguna CA subordinada. Una entidad de certificación con una restricción de longitud de ruta de uno puede tener hasta un nivel de entidades de certificación subordinadas debajo de ella. 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.

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
Guía de solución de problemas de nuevos conectores	Se han añadido dos 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	Adopción de nuevos nombres para las políticas de IAM AWS	13 de febrero de 2023

que coincidan con el nombre del nuevo servicio	administradas que especific an los permisos estándar para. Autoridad de certificación privada de AWS Para obtener más información, consulte Políticas administradas de AWS.	
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 certificación privada de AWS.	11 de noviembre de 2022
Soporte de API y CLI para las CA que emiten 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	Se cambia el nombre del servicio a 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 las CA y los certificados 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
Compatible con la característica Bloqueo de acceso público en S3 para CRL	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 mediante entidades de certificación privadas que no son de su propiedad. Para obtener más información, consulte Acceso entre cuentas a CA privadas.	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 AWS CloudFormation. Para obtener más información, consulte Referencia del tipo de recurso ACMPCA en la guía del usuario de AWS 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	Permisos basados en etiquetas compatibles con las nuevas API `TagResource` , `UntagResource` y `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 las plantillas para la firma de 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	La compatibilidad para crear y alojar CA raíz elimina la necesidad de una matriz externa.	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 entidades de certificación eliminadas	La restauración de CA privadas permite a los clientes restaurar entidades de certificación (CA) durante un máximo de 30 días después de que se hayan eliminado. 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.