Manual do usuário

AWS Private Certificate Authority

Versão latest

AWS Private Certificate Authority: Manual do usuário

Table of Contents

O que CA privada da AWSé

CA privada da AWS permite a criação de hierarquias de autoridade de certificação (CA) privada, incluindo CAs raiz e subordinadas, sem os custos de investimento e manutenção da operação de uma CA local. Suas CAs privadas podem emitir certificados X.509 de entidade final úteis em cenários que incluem:

- Criar canais de comunicação TLS criptografados
- Autenticar usuários, computadores, endpoints de API e dispositivos de IoT
- Assinar código de forma criptográfica
- Implementar o protocolo OCSP para obter o status de revogação de certificados

CA privada da AWS as operações podem ser acessadas a partir do AWS Management Console, usando a CA privada da AWS API ou usando AWS CLI o.

Tópicos

Qual é o melhor serviço de certificado para minhas necessidades?

Há dois AWS serviços para emitir e implantar certificados X.509. Escolha o que melhor se adapta às suas necessidades. As considerações incluem se você precisa de certificados públicos ou privados, certificados personalizados, certificados que deseja implantar em outros AWS serviços ou gerenciamento e renovação automatizados de certificados.

1. CA privada da AWS: este serviço destina-se a clientes corporativos que criam uma infraestrutura de chave pública (PKI) dentro da nuvem da AWS e destina-se a uso privado dentro de uma

organização. Com CA privada da AWS, você pode criar sua própria hierarquia de CA e emitir certificados com ela para autenticar usuários internos, computadores, aplicativos, serviços, servidores e outros dispositivos e para assinar códigos de computador. Os certificados emitidos por uma CA privada são confiáveis apenas na sua organização, não na Internet.

Depois de criar uma CA privada, você tem a capacidade de emitir certificados diretamente (ou seja, sem obter validação de uma CA de terceiros) e personalizá-los para atender às necessidades internas da sua organização. Por exemplo, você pode querer:

- Criar certificados com qualquer nome de entidade.
- Criar certificados com qualquer data de expiração.
- Use qualquer algoritmo de chave privada e comprimento de chave compatíveis.
- Use qualquer algoritmo de assinatura compatível.
- Controlar a emissão de certificados usando modelos.

Você está no lugar certo para esse serviço. Para começar, faça login no console https://console.aws.amazon.com/acm-pca/.

2. AWS Certificate Manager (ACM) — Esse serviço gerencia certificados para clientes corporativos que precisam de uma presença pública confiável na Web usando TLS. Você pode implantar certificados do ACM no AWS Elastic Load Balancing, na Amazon, no CloudFront Amazon API Gateway e em outros serviços integrados. A aplicação mais comum desse tipo é um site público seguro com requisitos de tráfego significativos.

   Com esse serviço, é possível usar certificados públicos fornecidos pelo ACM (certificados do ACM) ou certificados importados para o ACM. Se você usa CA privada da AWS para criar uma CA, o ACM pode gerenciar a emissão de certificados dessa CA privada e automatizar as renovações de certificados.

   Para mais informações, consulte o Guia do usuário do AWS Certificate Manager.

Regiões

Como a maioria dos AWS recursos, as autoridades de certificação (CAs) privadas são recursos regionais. Para usar CAs privadas em mais de uma região, você deve criá-las nessas regiões. Não é possível copiar CAs privadas entre regiões. Acesse Regiões e endpoints da AWS na Referência geral da AWS ou na Tabela de regiões da AWS para ver a disponibilidade regional do CA privada da AWS.

> **Note**
>
> Atualmente, o ACM está disponível em algumas regiões que não CA privada da AWS estão.

Serviços integrados com AWS Private Certificate Authority

Se você costuma AWS Certificate Manager solicitar um certificado privado, poderá associar esse certificado a qualquer serviço integrado ao ACM. Isso se aplica tanto aos certificados encadeados a uma CA privada da AWS raiz quanto aos certificados encadeados a uma raiz externa. Para obter mais informações, consulte Serviços integrados no Guia AWS Certificate Manager do usuário.

Você também pode integrar CAs privadas ao Amazon Elastic Kubernetes Service para fornecer a emissão de certificados dentro de um cluster do Kubernetes. Para ter mais informações, consulte Proteger o Kubernetes com o CA privada da AWS.

> **Note**
>
> O Amazon Elastic Kubernetes Service não é um serviço integrado do ACM.

Se você usar a CA privada da AWS API ou AWS CLI emitir um certificado ou exportar um certificado privado do ACM, poderá instalar o certificado em qualquer lugar que desejar.

Algoritmos criptográficos com suporte

CA privada da AWS suporta os seguintes algoritmos criptográficos para geração de chave privada e assinatura de certificados.

Algoritmo compatível

Algoritmos de chave privada	Algoritmos de assinatura
RSA_2048	SHA256WITHECDSA
RSA_4096	SHA384WITHECDSA
EC_prime256v1	SHA512WITHECDSA
EC_secp384r1	SHA256WITHRSA

Algoritmos de chave privada	Algoritmos de assinatura
	SHA384WITHRSA SHA512WITHRSA

Essa lista se aplica somente aos certificados emitidos diretamente por CA privada da AWS meio de seu console, API ou linha de comando. Quando AWS Certificate Manager emite certificados usando um CA de CA privada da AWS, ele suporta alguns, mas não todos esses algoritmos. Para obter mais informações, consulte Solicitar um certificado privado no Guia AWS Certificate Manager do usuário.

> **Note**
>
> Em todos os casos, a família especificada de algoritmos de assinatura (RSA ou ECDSA) deve corresponder à família de algoritmos da chave privada da CA.

Cotas

CA privada da AWS atribui cotas ao seu número permitido de certificados e autoridades de certificação. As taxas de solicitação para ações de API também estão sujeitas a cotas. CA privada da AWS as cotas são específicas para uma AWS conta e região.

CA privada da AWS regula as solicitações de API em taxas diferentes, dependendo da operação da API. A limitação significa que CA privada da AWS rejeita uma solicitação válida porque a solicitação excede a cota da operação para o número de solicitações por segundo. Quando uma solicitação é limitada, CA privada da AWS retorna um ThrottlingExceptionerro. CA privada da AWS não garante uma taxa mínima de solicitação para APIs.

Para ver quais cotas podem ser ajustadas, consulte a tabela de CA privada da AWS cotas no. Referência geral da AWS

É possível visualizar cotas atuais e solicitar aumentos de cotas usando o AWS Service Quotas.

Para ver uma up-to-date lista de suas CA privada da AWS cotas

1. Faça login na sua AWS conta.
2. Abra o console Service Quotas em https://console.aws.amazon.com/servicequotas/.

3. Na lista Serviços, escolha AWS Certificate Manager Private Certificate Authority (ACM PCA). Cada cota na lista de Cotas de serviço mostra o valor da cota atualmente aplicada, o valor da cota padrão e se ela é ajustável ou não. Selecione o nome de uma cota para obter mais informações sobre ela.

Para solicitar um aumento da cota

1. Na lista Cotas de serviço, escolha o botão de opção de uma cota ajustável.
2. Escolha o botão Solicitar aumento de cota.
3. Preencha e envie o formulário Solicitar aumento de cota.

CA privada da AWS está integrado com AWS Certificate Manager. Você pode usar o console do ACM ou a API do ACM para solicitar certificados privados de uma CA privada existente. AWS CLI Esses certificados de PKI privada, que são gerenciados pelo ACM, estão sujeitos a cotas do PCA e às cotas que o ACM impõe sobre certificados públicos e importados. Para obter mais informações sobre os requisitos do ACM, consulte Solicitar um certificado privado e cotas no Guia do AWS Certificate Manager usuário.

Conformidade com RFC

CA privada da AWS não impõe certas restrições definidas na RFC 5280. A situação inversa também é verdadeira: determinadas restrições adicionais apropriadas a uma CA privada são impostas.

Impostas

- Não depois da data. Em conformidade com a RFC 5280, o CA privada da AWS impede a emissão de certificados com uma `Not After` data posterior `Not After` à data do certificado da CA emissora.
- Restrições básicas. CA privada da AWS impõe restrições básicas e comprimento do caminho em certificados CA importados.

  As restrições básicas indicam se o recurso identificado pelo certificado é ou não uma CA e pode emitir certificados. Os certificados CA importados no CA privada da AWS devem incluir a extensão das restrições básicas e a extensão deve ser marcada como `critical`. Além da `critical` bandeira, `CA=true` deve ser definida. CA privada da AWS impõe restrições básicas ao falhar com uma exceção de validação pelos seguintes motivos:

- A extensão não está incluída no certificado CA.
- A extensão não está marcada como `critical`.

O comprimento do caminho (caminho LenConstraint) determina quantas CAs subordinadas podem existir a jusante do certificado de CA importado. CA privada da AWS impõe o comprimento do caminho ao falhar com uma exceção de validação pelos seguintes motivos:

- Importar um certificado CA violaria a restrição de comprimento de caminho no certificado CA ou em qualquer certificado CA na cadeia.
- A emissão de um certificado violaria uma restrição de comprimento de caminho.

- As restrições de nome indicam um espaço de nomes dentro do qual todos os nomes de assuntos nos certificados subsequentes em um caminho de certificação devem estar localizados. As restrições se aplicam ao nome distinto do sujeito e aos nomes alternativos do assunto.

Não impostas

- Políticas de certificação. As políticas de certificado regulam as condições sob as quais uma CA emite certificados.
- Iniba qualquer política. Usado em certificados emitidos para CAs.
- Nome alternativo do emissor. Permite que identidades adicionais sejam associadas ao emissor do certificado CA.
- Restrições políticas. Essas restrições limitam a capacidade de uma CA para emitir certificados CA subordinados.
- Mapeamentos de políticas. Usado em certificados CA. Lista um ou mais pares de OIDs; cada par inclui um issuerDomainPolicy e um assuntoDomainPolicy.
- Atributos do diretório de assuntos. Usado para transmitir atributos de identificação do sujeito.
- Acesso às informações do assunto. Como acessar informações e serviços sobre o assunto do certificado no qual a extensão aparece.
- Identificador de chave de assunto (SKI) e Identificador de chave de autoridade (AKI). A RFC requer um certificado CA para conter a extensão do SKI. Os certificados emitidos pela CA devem conter uma extensão AKI correspondente à SKI do certificado CA. AWS não impõe esses requisitos. Se o certificado CA não contiver um SKI, a entidade final emitida ou o AKI do certificado CA subordinado será o hash SHA-1 da chave pública do emissor.
- SubjectPublicKeyInfoe Nome alternativo do assunto (SAN). Ao emitir um certificado, CA privada da AWS copia as extensões SAN SubjectPublicKeyInfo e a CSR fornecida sem realizar a validação.

Definição de preço

Um preço mensal é cobrado em sua conta por cada CA privada começando no momento em que ela é criada. Você também será cobrado por cada certificado emitido. Essa cobrança inclui certificados que você exporta do ACM e certificados que você cria da CA privada da AWS API ou da CA privada da AWS CLI. Você não será cobrado por uma CA privada depois que ela for excluída. No entanto, ao restaurar uma CA privada, você é cobrado pelo tempo entre a exclusão e a restauração. Os certificados privados a cuja chave privada você não tem acesso são gratuitos. Isso inclui certificados que são usados com serviços integrados, como Elastic Load Balancing e API CloudFront Gateway.

Para obter as informações mais recentes sobre CA privada da AWS preços, consulte AWS Private Certificate Authority Preços. Você também pode usar a Calculadora de preços da AWS para estimar os custos.

Segurança em AWS Private Certificate Authority

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de data centers e arquiteturas de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isso como a segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de AWS de . Para saber mais sobre os programas de conformidade aplicáveis AWS Private Certificate Authority, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS em Escopo por Programa .
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar CA privada da AWS. Os tópicos a seguir mostram como configurar para atender CA privada da AWS aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros Serviços da AWS que o ajudem a monitorar e proteger seus CA privada da AWS recursos.

Tópicos

Identity and Access Management (IAM) para AWS Private Certificate Authority

O acesso a CA privada da AWS requer credenciais que AWS possam ser usadas para autenticar suas solicitações. Os tópicos a seguir fornecem detalhes sobre como você pode usar o AWS Identity and Access Management (IAM) para ajudar a proteger suas autoridades de certificação (CAs) privadas controlando quem pode acessá-las.

Em CA privada da AWS, o principal recurso com o qual você trabalha é uma autoridade de certificação (CA). Cada CA privada que você possui ou controla é identificada por um nome de recurso da Amazon (ARN), que tem o seguinte formato.

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

O proprietário do recurso é a entidade principal da AWS conta na qual um AWS recurso é criado. Os exemplos a seguir ilustram como isso funciona.

- Se você usar suas credenciais Usuário raiz da conta da AWS para criar uma CA privada, sua AWS conta é proprietária da CA.

  > **Important**
  > - Não recomendamos o uso de um Usuário raiz da conta da AWS para criar CAs.
  > - É altamente recomendável usar a autenticação multifator (MFA) sempre que você acessar. CA privada da AWS

- Se você criar um usuário do IAM em sua AWS conta, poderá conceder a esse usuário permissão para criar uma CA privada. Porém, a conta à qual esse usuário pertence é proprietária da CA.
- Se você criar uma função do IAM em sua AWS conta e conceder a ela permissão para criar uma CA privada, qualquer pessoa que possa assumir a função poderá criar a CA. Porém, a conta à qual essa função pertence será proprietária da CA privada.

A política de permissões descreve quem tem acesso a quê. A seção a seguir explica as opções disponíveis para a criação das políticas de permissões.

> **Note**
>
> Esta documentação discute o uso do IAM no contexto de CA privada da AWS. Não são fornecidas informações detalhadas sobre o serviço IAM. Para concluir a documentação do IAM, consulte o Guia do usuário do IAM. Para obter mais informações sobre a sintaxe e as descrições de política do IAM, consulte AWS Referência de política do IAM.

CA privada da AWS Operações e permissões de API

Ao configurar o controle de acesso e as políticas de permissões a serem anexadas a uma identidade do IAM (políticas baseadas em identidade), use a tabela a seguir como referência. A primeira coluna na tabela lista cada operação CA privada da AWS da API. Especifique as ações em um elemento `Action` de política. As colunas remanescentes fornecem informações adicionais.

CA privada da AWS Operações de API	Permissões obrigatórias	Recursos
CreateCertificateAuthority	`acm-pca:CreateCertificateAuthority` `acm-pca:TagCertificateAuthority` (exigido somente ao criar uma CA com 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`
CreatePermission	`acm-pca:CreatePermission`	`arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-`

CA privada da AWS Operações de API	Permissões obrigatórias	Recursos
		1234-1122-2233-112 233445566
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*
DescribeCertificateAuthority	acm-pca:DescribeCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

CA privada da AWS Operações de API	Permissões obrigatórias	Recursos
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*
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*

CA privada da AWS Operações de API	Permissões obrigatórias	Recursos
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/D
ListPermissions	acm-pca:ListPermissions	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListTags	acm-pca:ListTags	N/D
PutPolicy	acm-pca:PutPolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

CA privada da AWS Operações de API	Permissões obrigatórias	Recursos
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*
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 conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .

- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criar um perfil para um provedor de identidades de terceiros (federação) no Guia do usuário do IAM.
- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

AWS políticas gerenciadas

CA privada da AWS inclui um conjunto de políticas AWS gerenciadas predefinidas para CA privada da AWS administradores, usuários e auditores. A compreensão dessas políticas pode ajudar a implementar Políticas gerenciadas pelo cliente.

Escolha qualquer uma das políticas abaixo para ver detalhes e exemplos de código de política.

AWSPrivateCAFullAccess

Concede controle administrativo irrestrito.

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

AWSPrivateCAReadOnly

Concede acesso limitado a operações de API somente leitura.

```
{
   "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

Concede capacidade de emitir e revogar certificados de CA. Essa política não tem outros recursos administrativos nem capacidade de emitir certificados de entidade final. As permissões são mutuamente exclusivas com a política de usuário.

```
{
   "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

Concede a capacidade de emitir e revogar certificados de entidade final. Essa política não tem capacidades administrativas nem capacidade de emitir certificados CA. As permissões são mutuamente exclusivas da 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

Concede acesso a operações de API somente leitura e permissão para gerar um relatório de auditoria 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":"*"
      }
   ]
}
```

Atualizações nas políticas AWS gerenciadas para CA privada da AWS

Na tabela a seguir, veja os detalhes sobre as atualizações das políticas AWS gerenciadas CA privada da AWS desde que o serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre todas as alterações CA privada da AWS, assine o feed RSS na Histórico do documento página.

Alterações de políticas gerenciadas

Alteração	Descrição	Data
Novos nomes de políticas: • `AWSPrivateCAFullAccess` • `AWSPrivateCAReadOnly` • `AWSPrivateCAPrivilegedUser` • `AWSPrivateCAAuditor` • `AWSPrivateCAUser`	Os prefixos do nome da política foram alterados de `AWSCertificateManagerPrivateCA` para `AWSPrivateCA` . A funcionalidade permanece a mesma.	13 de fevereiro de 2023

Políticas gerenciadas pelo cliente

Como prática recomendada, não use o seu Usuário raiz da conta da AWS para interagir com AWS, inclusive CA privada da AWS. Em vez disso, use AWS Identity and Access Management (IAM) para criar um usuário do IAM, uma função do IAM ou um usuário federado. Crie um grupo de administradores e adicione-se a ele. Depois, faça login como administrador. Adicione outros usuários ao grupo, conforme necessário.

Outra melhor prática é criar uma política do IAM gerenciada pelo cliente que você possa atribuir aos usuários. As políticas gerenciadas pelo cliente são independentes e baseadas em identidade que você cria e pode anexar a vários usuários, grupos ou funções na sua conta da AWS . Essa política restringe os usuários a realizar somente as CA privada da AWS ações que você especificar.

A política gerenciada pelo cliente de exemplo a seguir permite que um usuário crie um relatório de auditoria de CA. Isso é somente um exemplo. Você pode escolher qualquer CA privada da AWS operação que desejar. Para obter mais exemplos, consulte Políticas em linha.

Para criar uma política gerenciada pelo cliente

1. Faça login no console do IAM usando as credenciais de um administrador da AWS .
2. No painel de navegação do console, escolha Policies (Políticas).
3. Escolha Criar política.

4. Escolha a guia JSON.
5. Copie a seguinte política e cole-a no editor.

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

6. Escolha Revisar política.
7. Para Name (Nome), digite `PcaListPolicy`.
8. (Opcional) Digite uma descrição.
9. Escolha Criar política.

Um administrador pode anexar a política a qualquer usuário do IAM para limitar as ações do CA privada da AWS que o usuário pode executar. Para saber como aplicar uma política de permissões, consulte Alterar as permissões para um usuário do IAM, no Guia do usuário do IAM.

Políticas em linha

As políticas em linha são políticas criadas, gerenciadas e incorporadas diretamente a um usuário, grupo ou função. Os exemplos de políticas a seguir mostram como atribuir permissões para realizar CA privada da AWS ações. Para obter informações gerais sobre políticas em linha, consulte Trabalhar com políticas em linha no Guia do usuário do IAM. Você pode usar a API AWS Management Console, the AWS Command Line Interface (AWS CLI) ou IAM para criar e incorporar políticas em linha.

> **Important**
>
> É altamente recomendável usar a autenticação multifator (MFA) sempre que você acessar. CA privada da AWS

Tópicos

- Listar CAs privadas
- Recuperar um certificado de CA privada
- Importar um certificado de CA privada
- Excluir uma CA privada
- Tag-on-create: Anexando tags a uma CA no momento da criação
- Tag-on-create: Marcação restrita
- Controlar o acesso à CA privada usando etiquetas
- Acesso somente para leitura a CA privada da AWS
- Acesso total ao CA privada da AWS
- Acesso de administrador a todos os recursos da AWS

Listar CAs privadas

A política a seguir permite que um usuário liste todas as CAs privadas em uma conta.

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

Recuperar um certificado de CA privada

A política a seguir permite que um usuário recupere um certificado de CA privada específico.

```
{
   "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 um certificado de CA privada

A política a seguir permite que um usuário importe um certificado de 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"
   }
}
```

Excluir uma CA privada

A política a seguir permite que um usuário exclua uma CA privada específica.

```
{
   "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: Anexando tags a uma CA no momento da criação

A política a seguir permite que um usuário aplique etiquetas durante a criação da CA.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": [
            "acm-pca:CreateCertificateAuthority",
            "acm-pca:TagCertificateAuthority"
         ],
```

```
         "Effect": "Allow",
         "Resource": "*"
      }
   ]
}
```

Tag-on-create: Marcação restrita

A tag-on-create política a seguir impede o uso do par de valores-chave Environment=Prod durante a criação da CA. A marcação com outros pares de chave/valor é permitida.

```
{
   "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 o acesso à CA privada usando etiquetas

A política a seguir permite acesso somente às CAs com o par de valores-chave Environment=. PreProd Ela também exige que novas CAs incluam essa etiqueta.

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

```
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:*"
         ],
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "aws:ResourceTag/Environment":[
                  "PreProd"
               ]
            }
         }
      }
   ]
}
```

Acesso somente para leitura a CA privada da AWS

A política a seguir permite que um usuário descreva e liste autoridades de certificação privadas e recupere o certificado CA privado e a cadeia 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":"*"
   }
}
```

Acesso total ao CA privada da AWS

A política a seguir permite que um usuário execute qualquer CA privada da AWS ação.

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

Acesso de administrador a todos os recursos da AWS

A política a seguir permite que um usuário execute qualquer ação em qualquer AWS recurso.

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

Práticas de segurança recomendadas para acesso entre contas a CAs privadas

Um CA privada da AWS administrador pode compartilhar uma CA com diretores (usuários, funções etc.) em outra AWS conta. Quando um compartilhamento é recebido e aceito, o principal pode usar a CA para emitir certificados de entidade final usando CA privada da AWS nossos AWS Certificate Manager recursos. O diretor pode usar a CA para emitir certificados de CA subordinados usando CA privada da AWS.

> **Important**
>
> As cobranças associadas a um certificado emitido em um cenário de várias contas são cobradas na AWS conta que emite o certificado.

Para compartilhar o acesso a uma CA, CA privada da AWS os administradores podem escolher um dos seguintes métodos:

- Use AWS Resource Access Manager (RAM) para compartilhar a CA como um recurso com um diretor em outra conta ou com AWS Organizations. A RAM é um método padrão para compartilhar AWS recursos entre contas. Para obter mais informações sobre o RAM, consulte o Guia do usuário do AWS RAM. Para obter mais informações sobre AWS Organizations, consulte o Guia AWS Organizations do usuário.
- Use a CA privada da AWS API ou a CLI para anexar uma política baseada em recursos a uma CA, concedendo acesso a um principal em outra conta. Para ter mais informações, consulte Políticas baseadas em recurso.

A seção Controlar o acesso a uma CA privada deste guia fornece fluxos de trabalho para conceder acesso a CAs em cenários de conta única e entre contas.

Políticas baseadas em recurso

Políticas baseadas em recurso são políticas de permissões que você cria e anexa manualmente a um recurso (nesse caso, uma CA privada), em vez de a uma identidade ou um perfil de usuário. Ou, em vez de criar suas próprias políticas, você pode usar políticas AWS gerenciadas para AWS Private CA. Usando AWS RAM para aplicar uma política baseada em recursos, um CA privada da AWS administrador pode compartilhar o acesso a uma CA com um usuário em uma AWS conta diferente, diretamente ou por meio dela. AWS Organizations Como alternativa, um CA privada da AWS administrador pode usar as APIs do PCA,, e PutPolicyGetPolicyDeletePolicy, ou os AWS CLI comandos correspondentes put-policy, get-policy e delete-policy, para aplicar e gerenciar políticas baseadas em recursos.

Para obter informações gerais sobre políticas baseadas em recurso, consulte Políticas baseadas em identidade e políticas baseadas em recurso e Controlar o acesso usando políticas.

Para ver a lista de políticas baseadas em recursos AWS gerenciados para AWS Private CA, navegue até a biblioteca de permissões gerenciadas no AWS Resource Access Manager console e pesquise

por. CertificateAuthority Como acontece com qualquer política, antes de aplicá-la, recomendamos aplicar a política em um ambiente de teste para garantir que ela atenda aos seus requisitos.

AWS Certificate Manager Os usuários (ACM) com acesso compartilhado entre contas a uma CA privada podem emitir certificados gerenciados assinados pela CA. Emissores entre contas estão limitados por uma política baseada em recursos e têm acesso somente aos seguintes modelos de certificado de entidade final:

- EndEntityCertificate/V1
- EndEntityClientAuthCertificate/V1
- EndEntityServerAuthCertificate/V1
- BlankEndEntityCertificate_Passagem de API/v1
- BlankEndEntityCertificate_APICSRPassthrough/v1
- Certificado CA subordinado_ 0/V1 PathLen

Exemplos de políticas

Esta seção inclui exemplos de políticas entre contas para várias necessidades. Em todos os casos, o padrão de comando a seguir é usado para aplicar uma 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
```

Além de especificar o ARN de uma CA, o administrador fornece AWS uma ID de conta ou AWS Organizations uma ID que terá acesso à CA. O JSON de cada uma das políticas a seguir é formatado como um arquivo para facilitar a leitura, mas também pode ser fornecido como argumentos internos da CLI.

> **Note**
>
> A estrutura das políticas baseadas em recurso JSON mostradas abaixo deve ser seguida com precisão. Somente os campos de ID dos principais (o número da AWS conta ou o ID do AWS Organizations) e os CA ARNs podem ser configurados pelos clientes.

1. Arquivo: policy1.json: compartilhamento do acesso a uma CA com um usuário em uma conta diferente

   Substitua *555555555555* pela ID da AWS conta que está compartilhando a CA.

   Para o ARN do recurso, substitua o seguinte pelos seus próprios valores:

   - *aws*- A AWS divisória. Por exemplo,`aws`,`aws-us-gov`,`aws-cn`, etc.
   - *us-east-1*- A AWS região em que o recurso está disponível, como`us-west-1`.
   - *111122223333*- O ID da AWS conta do proprietário do recurso.
   - *11223344-1234-1122-2233-112233445566*- O ID do recurso da autoridade de certificação.

```
{
  "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. Arquivo: policy2.json — Compartilhando o acesso a uma CA por meio de AWS Organizations

   Substitua *o-a1b2c3d4z5* pelo ID. AWS Organizations

   Para o ARN do recurso, substitua o seguinte pelos seus próprios valores:

   - *aws*- A AWS divisória. Por exemplo,`aws`,`aws-us-gov`,`aws-cn`, etc.
   - *us-east-1*- A AWS região em que o recurso está disponível, como`us-west-1`.
   - *111122223333*- O ID da AWS conta do proprietário do recurso.
   - *11223344-1234-1122-2233-112233445566*- O ID do recurso da autoridade de certificação.

```
{
   "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"
         }
      }
   ]
}
```

Proteção de dados em AWS Private Certificate Authority

O modelo de responsabilidade AWS compartilhada modelo se aplica à proteção de dados em AWS Private Certificate Authority. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as Perguntas frequentes sobre privacidade de dados. Para mais informações sobre a proteção de dados na Europa, consulte o artigo AWS Shared Responsibility Model and GDPR no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access

Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos. AWS Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para ter mais informações sobre endpoints do FIPS, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com CA privada da AWS ou Serviços da AWS usa o console, a API ou AWS os SDKs. AWS CLI Quaisquer dados inseridos em tags ou campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, recomendamos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

Conformidade de armazenamento e segurança de chaves CA privada da AWS privadas

As chaves privadas para CAs privadas são armazenadas em módulos AWS gerenciados de segurança de hardware (HSMs). Os HSMs estão em conformidade com os requisitos de segurança FIPS PUB 140-2 de nível 3 para módulos criptográficos.

Criptografia de dados no AWS Private CA Connector for Active Directory

AWS Private CA O Connector for AD armazena dados de configuração do cliente em relação a conectores, modelos, registros de diretórios, nomes principais de serviços e entradas de controle

de acesso de grupos de modelos. Esses dados são criptografados em trânsito e em repouso. Informações sobre certificados emitidos por meio do Connector for AD podem ser descobertas usando a GetCertificateação na AWS Private CA API. Nenhuma informação sobre os certificados emitidos, ou sobre o cliente ou a máquina que está solicitando um certificado, é armazenada pelo AWS.

Validação de conformidade do AWS Private Certificate Authority

Auditores terceirizados avaliam a segurança e a conformidade AWS Private Certificate Authority como parte de vários programas de AWS conformidade. Isso inclui SOC, PCI, FedRAMP, HIPAA e outros.

Para obter uma lista de AWS serviços no escopo de programas de conformidade específicos, consulte AWS Serviços no escopo por programa de conformidade Serviços da AWS . Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar CA privada da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Para organizações que precisam criptografar seus buckets do Amazon S3, os tópicos a seguir descrevem como configurar a criptografia para acomodar ativos: CA privada da AWS
  - Criptografar relatórios de auditoria
  - Criptografar as CRLs
- Guias de início rápido sobre sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos com foco em segurança e conformidade em. AWS
- Documento técnico sobre arquitetura para segurança e conformidade com a HIPAA — Este whitepaper descreve como as empresas podem usar para criar aplicativos compatíveis com a HIPAA. AWS
- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.

- Avaliar recursos com regras no Guia do desenvolvedor do AWS Config : o serviço AWS Config avalia como as configurações de recursos estão em conformidade com práticas internas, diretrizes do setor e regulamentos.
- AWS Security Hub— Esse AWS serviço fornece uma visão abrangente do seu estado de segurança interno, AWS que ajuda você a verificar sua conformidade com os padrões e as melhores práticas do setor de segurança.

Usar relatórios de auditoria com sua CA privada

Você pode criar um relatório de auditoria para listar todos os certificados que sua CA privada emitiu ou revogou. O relatório é salvo em um bucket do S3 novo ou existente que você especifica na entrada.

Para obter informações sobre como adicionar proteção de criptografia aos relatórios de auditoria, consulte Criptografar relatórios de auditoria .

O arquivo do relatório de auditoria tem o seguinte caminho e nome de arquivo. O ARN de um bucket do Amazon S3 é o valor para `bucket-name`, `CA_ID` é o identificador exclusivo de uma CA emissora. `UUID` é o identificador exclusivo de um relatório de auditoria.

```
bucket-name/audit-report/CA_ID/UUID.[json|csv]
```

Você pode gerar um novo relatório a cada 30 minutos e fazer download do bucket. O exemplo a seguir mostra um relatório separado por 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
```

O exemplo a seguir mostra um relatório formatado em 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

Quando AWS Certificate Manager renova um certificado, o relatório de auditoria privado da CA preenche o `requestedByServicePrincipal` campo com. `acm.amazonaws.com` Isso indica que o AWS Certificate Manager serviço convocou a `IssueCertificate` ação da CA privada da AWS API em nome de um cliente para renovar o certificado.

Preparar um bucket do Amazon S3 para relatórios de auditoria

Para armazenar relatórios de auditoria, é necessário preparar um bucket do Amazon S3. Para obter mais informações, consulte Como criar um bucket do S3?

Seu bucket do S3 deve ser protegido por uma política de permissões anexada. Usuários autorizados e diretores de serviços precisam de `Put` permissão CA privada da AWS para colocar objetos no bucket e `Get` permissão para recuperá-los. Recomendamos que você aplique a política mostrada abaixo, que restringe o acesso a uma conta da AWS e ao ARN de uma CA privada. Para obter mais informações, consulte Adicionar uma política de bucket usando o console do Amazon S3.

> **Note**
>
> Durante o procedimento do console para criar um relatório de auditoria, você pode optar por deixar CA privada da AWS criar um novo bucket e aplicar uma política de permissões padrão. A política padrão não aplica restrição de `SourceArn` à CA e, portanto, é mais permissiva que a política recomendada. Se você escolher a política padrão, sempre poderá modificá-la posteriormente.

```
{
   "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"
            }
         }
      }
   ]
}
```

Criar um relatório de auditoria

É possível criar um relatório de auditoria do console ou da AWS CLI.

Como criar um relatório de auditoria (console)

1. Faça login na sua AWS conta e abra o CA privada da AWS console em https://console.aws.amazon.com/acm-pca/home.
2. Na página Autoridades de certificação privadas, escolha sua CA privada na lista.
3. No menu Ações, selecione Gerar relatório de auditoria.
4. Em Destino do relatório de auditoria, para Criar um novo bucket do S3?, escolha Sim e digite um nome de bucket exclusivo ou escolha Não e escolha um bucket existente na lista.

   Se você escolher Sim, CA privada da AWS cria e anexa a política padrão ao seu bucket. Caso selecione No (Não), você deverá anexar uma política ao bucket antes de gerar um relatório de auditoria. Use o padrão de política descrito em Preparar um bucket do Amazon S3 para relatórios de auditoria. Para obter informações sobre como anexar uma política, consulte Adicionar uma política de bucket usando o console do Amazon S3
5. Em Formato de saída, escolha JSON para notação de JavaScript objeto ou CSV para valores separados por vírgula.
6. Selecione Generate audit report (Gerar relatório de auditoria).

Como criar um relatório de auditoria (AWS CLI)

1. Se você ainda não tiver um bucket do S3 para usar, crie um.
2. Anexe uma política ao bucket. Use o padrão de política descrito em Preparar um bucket do Amazon S3 para relatórios de auditoria. Para obter informações sobre como anexar uma política, consulte Adicionar uma política de bucket usando o console do Amazon S3
3. Use o comando create-certificate-authority-audit-report para criar o relatório de auditoria e colocá-lo no bucket 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 um relatório de auditoria

Para recuperar um relatório de auditoria para inspeção, use o console do Amazon S3, a API, a CLI ou o SDK. Para ter mais informações, consulte Baixar um objeto no Guia do usuário do Amazon Simple Storage Service.

Criptografar relatórios de auditoria

Opcionalmente, você pode configurar a criptografia no bucket do Amazon S3 contendo seus relatórios de auditoria. CA privada da AWS suporta dois modos de criptografia para ativos no S3:

- Criptografia automática no lado do servidor com chaves AES-256 gerenciadas pelo Amazon S3.
- Criptografia gerenciada pelo cliente usando AWS Key Management Service e AWS KMS key configurada de acordo com suas especificações.

> **Note**
>
> CA privada da AWS não suporta o uso de chaves KMS padrão geradas automaticamente pelo S3.

Os procedimentos a seguir descrevem como configurar cada uma das opções de criptografia.

Como configurar uma criptografia automática

Siga as etapas para habilitar a criptografia do servidor do S3.

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Na tabela Buckets, escolha o bucket que conterá seus CA privada da AWS ativos.
3. Na página do bucket, escolha a guia Propriedades.
4. Escolha o cartão Criptografia padrão.

5. Escolha Habilitar.
6. Escolh Chave do Amazon S3 (SSE-S3).
7. Escolha Salvar alterações.

Como configurar a criptografia personalizada

Siga as etapas para habilitar a criptografia usando uma chave personalizada.

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Na tabela Buckets, escolha o bucket que conterá seus CA privada da AWS ativos.
3. Na página do bucket, escolha a guia Propriedades.
4. Escolha o cartão Criptografia padrão.
5. Escolha Habilitar.
6. Escolha a AWS Key Management Service chave (SSE-KMS).
7. Escolha Escolher entre suas AWS KMS chaves ou Inserir AWS KMS key ARN.
8. Escolha Salvar alterações.
9. (Opcional) Se ainda não tiver uma chave KMS, crie uma usando o seguinte comando da AWS CLI create-key:

```
$ aws kms create-key
```

A saída contém o ID de chave e o nome do recurso da Amazon (ARN) da chave do KMS. Veja a seguir um exemplo de saída:

```
{
    "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. Usando as etapas a seguir, você concede permissão ao responsável pelo CA privada da AWS serviço para usar a chave KMS. Por padrão, todas as chaves do KMS são privadas; somente o proprietário do recurso pode usá-las para criptografar e descriptografar dados. No entanto, o proprietário do recurso pode conceder permissões para acessar a chave do KMS a outros usuários e recursos. A entidade principal desse serviço deve estar na mesma região em que a chave do KMS está armazenada.

    a. Primeiro, salve a política padrão para sua chave KMS `policy.json` usando o seguinte get-key-policycomando:

    ```
    $ aws kms get-key-policy --key-id key-id --policy-name default --output text
     > ./policy.json
    ```

    b. Abra o arquivo `policy.json` em um editor de textos. Selecione uma das declarações de políticas a seguir e adicione-a à política existente.

    Se a sua chave de bucket do Amazon S3 estiver habilitada, use a seguinte declaração:

    ```
    {
       "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"
          }
       }
    }
    ```

    Se a sua chave de bucket do Amazon S3 estiver desabilitada, use a seguinte declaração:

    ```
    {
       "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 fim, aplique a política atualizada usando o seguinte put-key-policycomando:

```
$ aws kms put-key-policy --key-id key_id --policy-name default --policy file://
policy.json
```

Segurança da infraestrutura em AWS Private Certificate Authority

Como serviço gerenciado, AWS Private Certificate Authority é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar AWS Private CA pela rede. Os clientes precisam oferecer suporte para:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com sigilo de encaminhamento perfeito (perfect forward secrecy, ou PFS) como DHE (Ephemeral Diffie-Hellman, ou Efêmero Diffie-Hellman) ou ECDHE (Ephemeral Elliptic

Curve Diffie-Hellman, ou Curva elíptica efêmera Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas utilizando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou é possível usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

CA privada da AWS Endpoints de VPC ()AWS PrivateLink

Você pode criar uma conexão privada entre sua VPC e CA privada da AWS configurar uma interface VPC endpoint. Os endpoints de interface são alimentados por AWS PrivateLink, uma tecnologia para acessar de forma privada as operações de CA privada da AWS API. AWS PrivateLink roteia todo o tráfego de rede entre sua VPC e CA privada da AWS através da rede Amazon, evitando exposição na Internet aberta. Cada VPC endpoint é representado por uma ou mais interfaces de rede elástica com endereços IP privados em suas sub-redes da VPC.

A interface VPC endpoint conecta sua VPC diretamente CA privada da AWS sem um gateway de internet, dispositivo NAT, conexão VPN ou conexão. AWS Direct Connect As instâncias na sua VPC não precisam de endereços IP públicos para se comunicarem com a CA privada da AWS API.

Para usar CA privada da AWS por meio de sua VPC, você deve se conectar a partir de uma instância que esteja dentro da VPC. Como alternativa, você pode conectar sua rede privada à sua VPC usando um AWS Virtual Private Network (AWS VPN) ou. AWS Direct Connect Para obter informações sobre isso AWS VPN, consulte Conexões VPN no Guia do usuário da Amazon VPC. Para obter informações sobre AWS Direct Connect, consulte Criação de uma conexão no Guia AWS Direct Connect do usuário.

CA privada da AWS não requer o uso de AWS PrivateLink, mas nós o recomendamos como uma camada adicional de segurança. Para obter mais informações sobre endpoints AWS PrivateLink de VPC, consulte Acessando serviços por meio de. AWS PrivateLink

Considerações sobre CA privada da AWS VPC endpoints

Antes de configurar a interface para VPC endpoints CA privada da AWS, esteja ciente das seguintes considerações:

- CA privada da AWS pode não oferecer suporte a VPC endpoints em algumas zonas de disponibilidade. Ao criar um endpoint de VPC, primeiro verifique o suporte no console de

gerenciamento. Zonas de disponibilidade sem suporte são marcadas como “Serviço sem suporte nesta zona de disponibilidade”.

- Os VPC endpoints não são compatíveis com solicitações entre regiões. Garanta a criação do seu endpoint na mesma Região onde planeja emitir as chamadas de API para o CA privada da AWS.
- Os endpoints da VPC oferecem suporte somente a DNS fornecidos pela Amazon por meio do Amazon Route 53. Se quiser usar seu próprio DNS, você pode usar o encaminhamento de DNS condicional. Para obter mais informações, consulte Conjuntos de Opções de DHCP no Manual do Usuário da Amazon VPC.
- O grupo de segurança anexado ao endpoint da VPC deve permitir entrada conectada a porta 443 na sub-rede privada da VPC.
- AWS Certificate Manager não oferece suporte a endpoints VPC.
- Os endpoints FIPS (e suas regiões) não são compatíveis com VPC endpoints.

CA privada da AWS Atualmente, a API oferece suporte a endpoints VPC no seguinte: Regiões da AWS

- Leste dos EUA (Ohio)
- Leste dos EUA (N. da Virgínia)
- Oeste dos EUA (N. da Califórnia)
- Oeste dos EUA (Oregon)
- África (Cidade do Cabo)
- Ásia-Pacífico (Hong Kong)
- Ásia-Pacífico (Mumbai)
- Ásia-Pacífico (Osaka)
- Ásia-Pacífico (Seul)
- Ásia-Pacífico (Singapura)
- Ásia-Pacífico (Sydney)
- Ásia-Pacífico (Tóquio)
- Canadá (Central)
- Europa (Frankfurt)
- Europa (Irlanda)

- Europa (Londres)
- Europa (Paris)
- Europa (Estocolmo)
- Europa (Milão)
- Israel (Tel Aviv)
- Middle East (Bahrain)
- South America (São Paulo)

Criar os endpoints da VPC para o CA privada da AWS

Você pode criar um VPC endpoint para o CA privada da AWS serviço usando o console VPC em https://console.aws.amazon.com/vpc/ ou o. AWS Command Line Interface Para obter mais informações, consulte o procedimento Criação de um endpoint de interface no Guia do usuário da Amazon VPC. CA privada da AWS suporta fazer chamadas para todas as suas operações de API dentro da sua VPC.

Se você habilitou nomes de host DNS privados para o endpoint, o endpoint padrão do CA privada da AWS agora será resolvido para o endpoint de VPC. Para obter uma lista abrangente de endpoints de serviço padrão, consulte Endpoints e cotas de serviço.

Se você não habilitar nomes de host DNS privados, a Amazon VPC fornecerá um nome de endpoint DNS que você poderá usar no seguinte formato:

```
vpc-endpoint-id.acm-pca.region.vpce.amazonaws.com
```

> **Note**
>
> A *região* de valor representa o identificador de uma AWS região suportada por CA privada da AWS, como `us-east-2` a região Leste dos EUA (Ohio). Para obter uma lista de CA privada da AWS, consulte AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.

Para obter mais informações, consulte CA privada da AWS VPC endpoints (AWS PrivateLink) no Guia do usuário do Amazon VPC.

Criar uma política de VPC endpoint para o CA privada da AWS

Você pode criar uma política para endpoints do Amazon VPC CA privada da AWS para especificar o seguinte:

- A entidade principal que pode executar ações
- As ações que podem ser executadas
- Os recursos nos quais as ações podem ser executadas

Para obter mais informações, consulte Controlar o acesso a serviços com endpoint da VPCs no Manual do usuário da Amazon VPC.

Exemplo — política de VPC endpoint para ações CA privada da AWS

Quando anexada a um endpoint, a política a seguir concede acesso a todos os diretores às CA privada da AWS ações`IssueCertificate`,`DescribeCertificateAuthority`,`GetCertificate`, `GetCertificateAuthorityCertificate`e. `ListPermissions` `ListTags` O recurso em cada estrofe é uma CA privada. A primeira estrofe autoriza a criação de certificados de entidade final usando a CA privada especificada e o modelo de certificado. Se você não quiser controlar o modelo que está sendo usado, a seção `Condition` não será necessária. No entanto, remover isso permite que todos os principais criem certificados CA bem como certificados de entidade 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"
            ]
         }
      ]
   }
```

Registro em log e monitoramento da AWS Private Certificate Authority

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho de AWS Private Certificate Authority suas AWS soluções. Você deve coletar dados de monitoramento de todas as partes da sua AWS solução para poder depurar com mais facilidade uma falha de vários pontos, caso ocorra.

Os tópicos a seguir descrevem as ferramentas AWS de monitoramento de nuvem disponíveis para uso com. CA privada da AWS

Tópicos

- CloudWatch Métricas suportadas
- Usando CloudWatch eventos
- Usando CloudTrail

CloudWatch Métricas suportadas

A Amazon CloudWatch é um serviço de monitoramento de AWS recursos. Você pode usar CloudWatch para coletar e monitorar métricas, definir alarmes e reagir automaticamente às mudanças em seus AWS recursos. CloudWatch as métricas são publicadas pelo menos uma vez.

CA privada da AWS suporta as seguintes CloudWatch métricas.

Métrica	Descrição
`CRLGenerated`	Uma lista de revogação de certificados (CRL) foi gerada. Esta métrica se aplica apenas a uma CA privada.
`MisconfiguredCRLBucket`	O bucket do S3 especificado para o CRL não está configurado corretamente. Verifique a política de bucket. Esta métrica se aplica apenas a uma CA privada.
`Time`	O tempo em milissegundos entre uma solicitação de emissão e a conclusão (ou falha) dessa emissão. Essa métrica se aplica somente à IssueCertificateoperação.
`Success`	Um certificado foi emitido com sucesso. Essa métrica se aplica somente à IssueCertificateoperação.
`Failure`	Ocorreu uma falha na operação. Essa métrica se aplica somente à IssueCertificateoperação.

Para obter mais informações sobre CloudWatch métricas, consulte os tópicos a seguir:

- Usando o Amazon CloudWatch Metrics
- Criação de CloudWatch alarmes da Amazon

Usando CloudWatch eventos

Você pode usar o Amazon CloudWatch Events para automatizar seus AWS serviços e responder automaticamente a eventos do sistema, como problemas de disponibilidade de aplicativos ou alterações de recursos. Os eventos dos AWS serviços são entregues aos CloudWatch Eventos quase em tempo real. Você pode escrever regras simples para indicar quais eventos são de seu interesse e as ações automatizadas a serem tomadas quando um evento corresponde a uma regra. CloudWatch Os eventos são publicados pelo menos uma vez. Para obter mais informações, consulte Criação de uma regra de CloudWatch eventos que é acionada em um evento.

CloudWatch Os eventos são transformados em ações usando a Amazon EventBridge. Com EventBridge, você pode usar eventos para acionar alvos, incluindo AWS Lambda funções, AWS Batch trabalhos, tópicos do Amazon SNS e muitos outros. Para obter mais informações, consulte O que é a Amazon EventBridge?

Sucesso ou falha ao criar uma CA privada

Esses eventos são acionados pela CreateCertificateAuthorityoperação.

Bem-sucedida

Em caso de sucesso, a operação retorna o ARN da nova 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":"success"
   }
}
```

Falha

Em caso de falha, a operação retorna um ARN para a CA. Usando o ARN, você pode ligar DescribeCertificateAuthoritypara determinar o status da 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"
   }
}
```

Sucesso ou falha ao emitir um certificado

Esses eventos são acionados pela IssueCertificateoperação.

Bem-sucedida

Em caso de sucesso, a operação retorna os ARNs da CA e do novo 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"
   }
}
```

Falha

Em caso de falha, a operação retorna o ARN de um certificado e o ARN da CA. Com o certificado ARN, você pode ligar GetCertificatepara ver o motivo da falha.

```
{
   "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"
   }
}
```

Sucesso ao revogar um certificado

Esse evento é acionado pela RevokeCertificateoperação.

Nenhum evento será enviado se a revogação falhar ou se o certificado já tiver sido revogado.

Bem-sucedida

Em caso de êxito, a operação retorna os ARNs da CA e do certificado revogado.

```
{
   "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"
   }
}
```

Sucesso ou falha ao gerar uma CRL

Esses eventos são acionados pela RevokeCertificateoperação, o que deve resultar na criação de uma lista de revogação de certificados (CRL).

Bem-sucedida

Em caso de sucesso, a operação retorna o ARN da CA associada à 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"
   }
}
```

Falha 1: a CRL não pôde ser salva no Amazon S3 devido a um erro de permissão

Verifique as permissões do bucket do Amazon S3 caso esse erro ocorra.

```
{
   "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."
   }
}
```

Falha 2: a CRL não pôde ser salva no Amazon S3 devido a um erro interno

Tente a operação novamente caso esse erro ocorra.

```
{
   "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."
   }
}
```

Falha 3 — CA privada da AWS falha ao criar uma CRL

Para solucionar esse erro, verifique as Métricas do 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."
   }
}
```

Êxito ou falha ao criar um relatório de auditoria da CA

Esses eventos são acionados pela CreateCertificateAuthorityAuditReportoperação.

Bem-sucedida

Em caso de sucesso, a operação retorna o ARN da CA e o ID do relatório de auditoria.

```
{
   "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"
   }
}
```

Falha

Um relatório de auditoria pode falhar quando CA privada da AWS não há PUT permissões em seu bucket do Amazon S3, quando a criptografia está habilitada no bucket ou por outros 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

Você pode usar AWS CloudTrailpara gravar chamadas de API feitas por AWS Private Certificate Authority. Para obter mais informações, consulte os tópicos a seguir.

Tópicos

- Criação de uma política
- Recuperar uma política
- Excluir uma política
- Criar uma autoridade de certificação
- GenerateCRL
- GenerateOCSPResponse
- Criar um relatório de auditoria
- Excluir uma autoridade de certificação
- Restaurar uma autoridade de certificação

- Descrever uma autoridade de certificado
- Recuperar um certificado de autoridade de certificado
- Recuperar a solicitação de assinatura de autoridade de certificado
- Recuperação de um certificado
- Importar certificado da autoridade de certificação
- Emitir um certificado
- Listar autoridades de certificação
- Listar tags
- Revogar um certificado
- Marcar autoridades de certificação privadas
- Remover etiquetas de uma autoridade de certificação privada
- Atualizar uma autoridade de certificação

Criação de uma política

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a PutPolicyoperação.


```
{
   "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 uma política

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a GetPolicyoperação.

```
{
   "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"
}
```

Excluir uma política

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a DeletePolicyoperação.

```
{
   "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"
}
```

Criar uma autoridade de certificação

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a CreateCertificateAuthorityoperação.

```
{
   "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

O CloudTrail exemplo a seguir mostra o registro de um 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

O CloudTrail exemplo a seguir mostra o registro de um evento GenerateOCSPRponse.


```
{
   "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"
      }
   ]
}
```

Criar um relatório de auditoria

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a CreateCertificateAuthorityAuditReportoperação.

```
{
   "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"
}
```

Excluir uma autoridade de certificação

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a DeleteCertificateAuthorityoperação. Neste exemplo, a autoridade de certificação não pode ser excluída, pois está no 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"
```

```
}
```

Restaurar uma autoridade de certificação

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a RestoreCertificateAuthorityoperação. Neste exemplo, a autoridade de certificação não pode ser restaurada, pois não está no 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"
}
```

Descrever uma autoridade de certificado

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a DescribeCertificateAuthorityoperação.

```
{
```

```
   "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 um certificado de autoridade de certificado

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a GetCertificateAuthorityCertificateoperação.


```
{
   "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 a solicitação de assinatura de autoridade de certificado

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a GetCertificateAuthorityCsroperação.

```
{
   "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"
}
```

Recuperação de um certificado

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a GetCertificateoperação.


```
{
   "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"
}
```


Importar certificado da autoridade de certificação

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a ImportCertificateAuthorityCertificateoperação.


```
{
   "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 um certificado

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a IssueCertificateoperação.

```
{
   "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"
}
```

Listar autoridades de certificação

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a ListCertificateAuthoritiesoperação.

```
{
   "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"
}
```

Listar tags

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a ListTagsoperação.

```
{
   "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"
}
```

Revogar um certificado

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a RevokeCertificateoperação.

```
{
   "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"
}
```

Marcar autoridades de certificação privadas

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a TagCertificateAuthorityoperação.

```
{
   "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"
}
```

Remover etiquetas de uma autoridade de certificação privada

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a UntagCertificateAuthorityoperação.

```
{
   "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"
}
```


Atualizar uma autoridade de certificação

O CloudTrail exemplo a seguir mostra os resultados de uma chamada para a UpdateCertificateAuthorityoperação.


```
{
   "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"
}
```

Planejando sua CA privada da AWS implantação

CA privada da AWS oferece controle completo baseado em nuvem sobre a PKI (infraestrutura de chave pública) privada de sua organização, desde uma autoridade de certificação raiz (CA), passando por CAs subordinadas, até certificados de entidade final. O planejamento completo é essencial para uma PKI ser segura, sustentável, extensível e adequada às necessidades da sua organização. Esta seção fornece orientações sobre como projetar uma hierarquia de CA, gerenciar os ciclos de vida de seus certificados de CA privada e de entidades finais privadas e aplicar as melhores práticas de segurança.

Esta seção descreve como se preparar CA privada da AWS para o uso antes de criar uma autoridade de certificação (CA) privada. Também explica a opção de adicionar suporte para revogação por meio do Online Certificate Status Protocol (OCSP) ou de uma lista de revogação de certificados (CRL).

Além disso, você deve determinar se sua organização prefere hospedar suas credenciais de CA raiz privada no local em vez de hospedar com. AWS Nesse caso, você precisa configurar e proteger uma PKI privada autogerenciada antes de usá-la. CA privada da AWS Nesse cenário, você então cria uma CA subordinada CA privada da AWS apoiada por uma CA principal fora da CA privada da AWS. Para obter mais informações, consulte Instalar um certificado de CA subordinada assinado por uma CA pai externa.

Tópicos

- Configurando sua AWS conta e o AWS CLI
- Criar uma hierarquia de CA
- Gerenciar o ciclo de vida da CA privada
- Configurar um método de revogação de certificado
- Modos de autoridades certificadoras
- Planejamento da resiliência

Configurando sua AWS conta e o AWS CLI

Se você ainda não for cliente da Amazon Web Services (AWS), deverá cadastrar-se para poder usar o CA privada da AWS. Sua conta tem acesso automaticamente a todos os serviços disponíveis, mas você é cobrado apenas pelos serviços que usa.

> **Note**
>
> CA privada da AWS não está disponível no nível AWS gratuito.

Tópicos

- Inscreva-se para um Conta da AWS
- Criar um usuário com acesso administrativo
- Instale o AWS Command Line Interface

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções on-line.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica e digitar um código de verificação no teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como uma prática recomendada de segurança, atribua o acesso administrativo para um usuário e use somente o usuário-raiz para executar tarefas que requerem o acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, digite sua senha.

   Para obter ajuda ao fazer login usando o usuário-raiz, consulte Signing in as the root user (Fazer login como usuário-raiz) no Guia do usuário do Início de Sessão da AWS .
2. Habilite a autenticação multifator (MFA) para o usuário-raiz.

   Para obter instruções, consulte Habilitar um dispositivo de MFA virtual para seu usuário Conta da AWS raiz (console) no Guia do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .
2. No Centro de Identidade do IAM, conceda o acesso administrativo para um usuário.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com seu usuário do Centro de Identidade do IAM, use a URL de login que foi enviada ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso para usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

Para obter instruções, consulte Create a permission set no Guia do usuário do AWS IAM Identity Center .

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Add groups no Guia do usuário do AWS IAM Identity Center .

Instale o AWS Command Line Interface

Se você não instalou o AWS CLI , mas deseja usá-lo, siga as instruções em AWS Command Line Interface. Neste guia, presumimos que você tenha configurado seu endpoint, região e detalhes de autenticação, e omitimos esses parâmetros dos comandos de amostra.

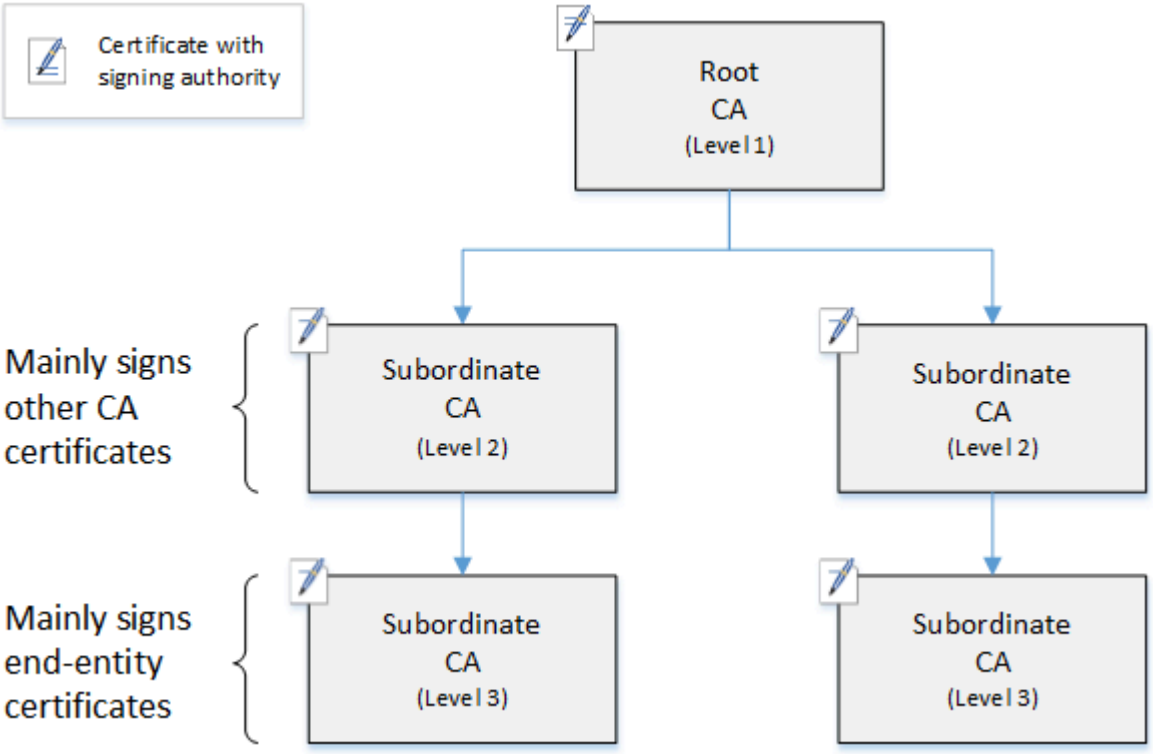
Criar uma hierarquia de CA

Com CA privada da AWS, você pode criar uma hierarquia de autoridades de certificação com até cinco níveis. A CA raiz, na parte superior de uma árvore de hierarquia, pode ter qualquer número de ramificações. A CA raiz pode ter até quatro níveis de CAs subordinadas em cada ramificação. Você também pode criar várias hierarquias, cada uma com sua própria raiz.

Uma hierarquia de CA bem projetada oferece os seguintes benefícios:

- Controles de segurança granulares apropriados para cada CA
- Divisão de tarefas administrativas para melhor balanceamento de carga e segurança
- Uso de CAs com confiança limitada e revogável para operações diárias
- Períodos de validade e limites de caminho de certificado

O diagrama a seguir ilustra uma hierarquia de CA de três níveis simples.



Cada CA na árvore é apoiada por um certificado X.509 v3 com autoridade de assinatura (simbolizada pelo ícone). pen-and-paper Isso significa que, como CAs, elas podem assinar outros certificados subordinados a elas. Quando uma CA assina um certificado CA de nível inferior, ela confere autoridade limitada e revogável no certificado assinado. A autoridade de certificação raiz no nível 1 assina certificados CA subordinados de alto nível no nível 2. Essas CAs, por sua vez, assinam certificados para CAs no nível 3 que são usados por administradores de PKI (infraestrutura de chave pública) que gerenciam certificados de entidade final.

A segurança em uma hierarquia de CA deve ser configurada para ser mais forte na parte superior da árvore. Essa disposição protege o certificado CA raiz e sua chave privada. A CA raiz ancora a confiança para todas as CAs subordinadas e para os certificados de entidade final abaixo dela. Embora danos localizados possam resultar do comprometimento de um certificado de entidade final, o comprometimento da raiz destrói a confiança em toda a PKI. As CAs raiz e subordinadas de nível superior são usadas com pouca frequência (geralmente para assinar outros certificados CA). Consequentemente, são rigorosamente controladas e auditadas para garantir um risco menor de comprometimento. Nos níveis inferiores da hierarquia, a segurança é menos restritiva. Essa abordagem permite as tarefas administrativas de rotina de emissão e revogação de certificados de entidade final para usuários, hosts de computador e serviços de software.

> **Note**
>
> Usar uma CA raiz para assinar um certificado subordinado é um evento raro que ocorre apenas em algumas circunstâncias:
>
> - Quando a PKI é criada
> - Quando uma CA de nível superior precisa ser substituída
> - Quando um respondedor de lista de revogação de certificados (CRL) ou de protocolo OCSP precisa ser configurado
>
> As CAs raiz e outras CAs de alto nível exigem processos operacionais altamente seguros e protocolos de controle de acesso.

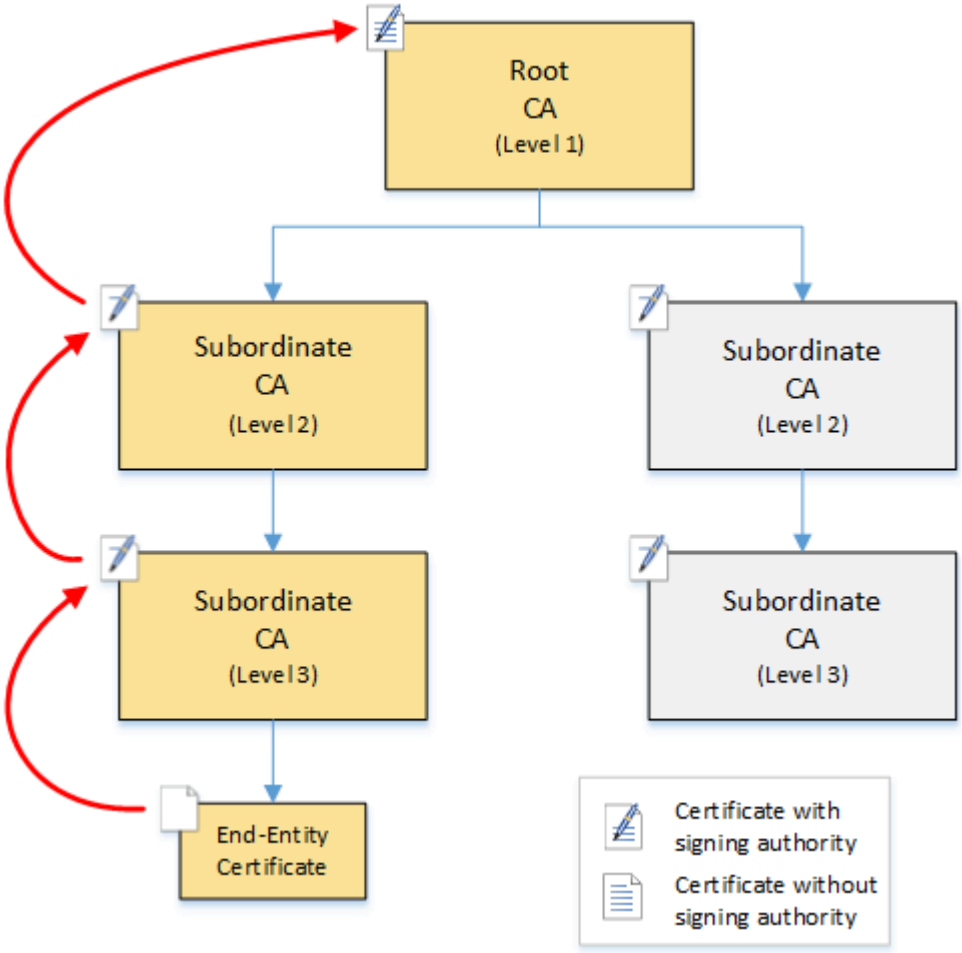
Tópicos

Validar certificados de entidade final

Os certificados de entidade final derivam sua confiança de um caminho de certificação que leva de volta por meio das CAs subordinadas para uma CA raiz. Quando um navegador da Web ou outro cliente é apresentado com um certificado de entidade final, ele tenta construir uma cadeia de confiança. Por exemplo, ele pode verificar se o nome distinto do emissor do certificado e o nome distinto do assunto correspondem aos campos correspondentes do certificado CA emissor. A correspondência continuaria em cada nível sucessivo na hierarquia até que o cliente atinja uma raiz confiável contida em seu armazenamento de confiança.

O armazenamento de confiança é uma biblioteca de CAs confiáveis que o navegador ou o sistema operacional contém. Para uma PKI privada, a TI da sua organização deve garantir que cada navegador ou sistema tenha previamente adicionado a CA raiz privada ao seu armazenamento de confiança. Caso contrário, o caminho de certificação não poderá ser validado, resultando em erros de cliente.

O diagrama seguinte mostra o caminho de validação que um navegador segue quando apresentado com um certificado X.509 de entidade final. Observe que o certificado de entidade final não possui autoridade de assinatura e serve apenas para autenticar a entidade que o possui.



O navegador inspeciona o certificado de entidade final. O navegador descobre que o certificado oferece uma assinatura de CA subordinada (nível 3) como sua credencial de confiança. Os certificados para as CAs subordinadas devem ser incluídos no mesmo arquivo PEM. Como alternativa, eles também podem estar em um arquivo separado que contém os certificados que compõem a cadeia de confiança. Ao encontrá-los, o navegador verifica o certificado de CA subordinada (nível 3) e descobre que ele oferece uma assinatura de CA subordinada (nível 2). Por sua vez, a CA subordinada (nível 2) oferece uma assinatura da CA raiz (nível 1) como sua credencial de confiança. Se o navegador encontrar uma cópia do certificado CA raiz privado pré-instalado em seu armazenamento de confiança, ele validará o certificado de entidade final como confiável.

Normalmente, o navegador também verifica cada certificado em relação a uma lista de revogação de certificados (CRL). Um certificado expirado, revogado ou configurado incorretamente é rejeitado e a validação falha.

Planejar estrutura de uma hierarquia de CA

Em geral, sua hierarquia de CA deve refletir a estrutura de sua organização. Aponte para um comprimento de caminho (ou seja, o número de níveis de CA) não maior do que o necessário para delegar funções administrativas e de segurança. Adicionar uma CA à hierarquia significa aumentar o número de certificados no caminho de certificação, o que aumenta o tempo de validação. Manter o comprimento do caminho no mínimo também reduz o número de certificados enviados do servidor ao cliente ao validar um certificado de entidade final.

Em teoria, uma CA raiz, que não tem pathLenConstraintparâmetros, pode autorizar níveis ilimitados de CAs subordinadas. Uma CA subordinada pode ter quantas CAs subordinadas filhas forem permitidas por sua configuração interna. CA privada da AWS hierarquias gerenciadas oferecem suporte a caminhos de certificação da CA com até cinco níveis de profundidade.

As estruturas de CA bem projetadas têm vários benefícios:

- Controles administrativos separados para diferentes unidades organizacionais
- A capacidade de delegar acesso a CAs subordinadas
- Uma estrutura hierárquica que protege as CAs de nível superior com controles de segurança adicionais

Duas estruturas comuns de CA realizam tudo isso:

- Dois níveis de CA: CA raiz e CA subordinada

  Essa é a estrutura de CA mais simples que permite políticas separadas de administração, controle e segurança para a CA raiz e uma CA subordinada. Você pode manter controles e políticas restritivos para sua CA raiz enquanto permite acesso mais permissivo para a CA subordinada. A última é utilizada para emissão em massa de certificados de entidade final.

- Três níveis de CA: CA raiz e duas camadas de CA subordinada

  Semelhante à acima, essa estrutura adiciona uma camada de CA para separar ainda mais a CA raiz das operações da CA de nível inferior. A camada de CA média é usada apenas para assinar CAs subordinadas que realizam a emissão de certificados de entidade final.

Estruturas de CA menos comuns incluem o seguinte:

- Quatro ou mais níveis de CA

  Embora menos comuns do que hierarquias de três níveis, as hierarquias de CA com quatro ou mais níveis são possíveis e podem ser necessárias para permitir delegação administrativa.

- Um nível de CA: somente CA raiz

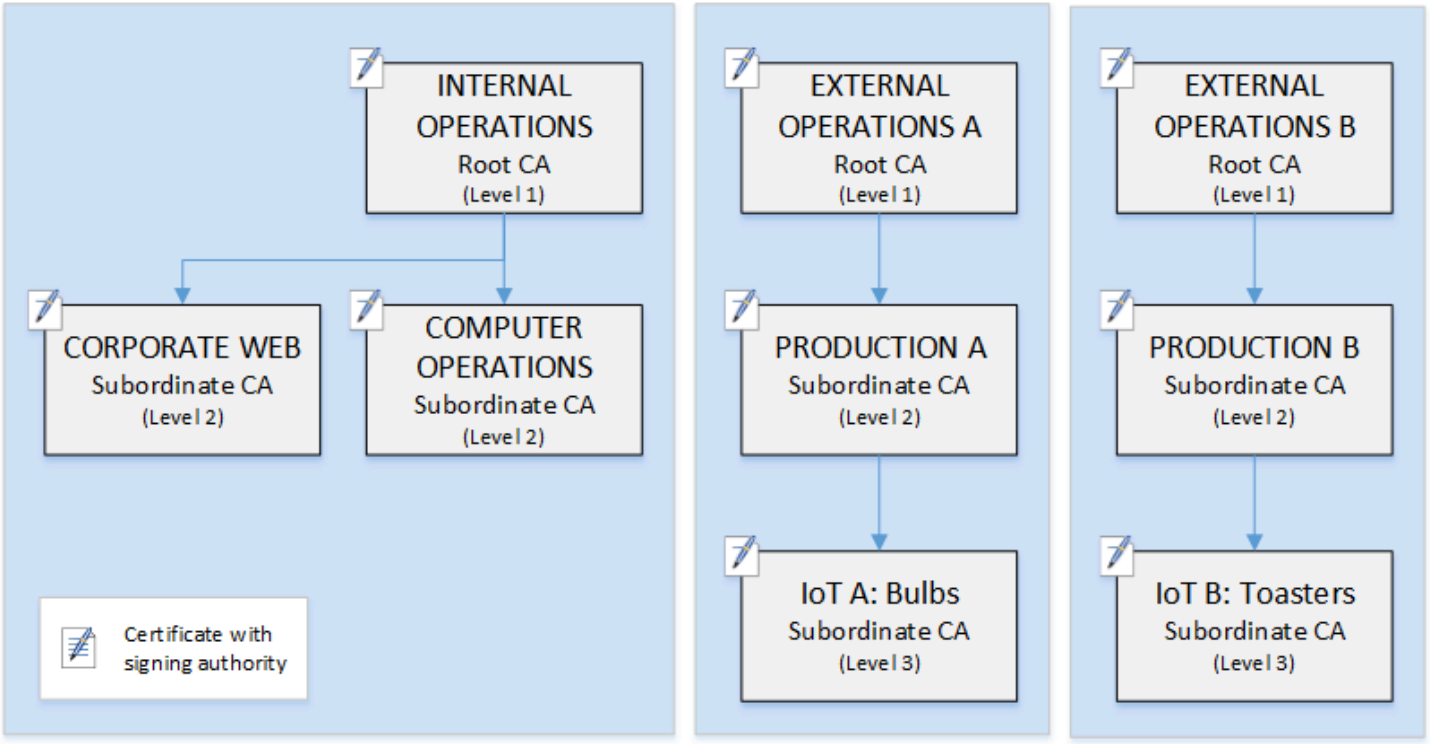
  Essa estrutura é comumente usada para desenvolvimento e teste quando uma cadeia completa de confiança não é necessária. Usada na produção, é atípica. Além disso, ela viola a prática recomendada de manter políticas de segurança separadas para a CA raiz e as CAs que emitem certificados de entidade final.

  No entanto, se você já estiver emitindo certificados diretamente de uma CA raiz, poderá migrar para o. CA privada da AWS Isso fornece vantagens de segurança e controle sobre o uso de uma CA raiz gerenciada com OpenSSL ou outro software.

Exemplo de uma PKI privada para um fabricante

Neste exemplo, uma empresa de tecnologia hipotética fabrica dois produtos da Internet das Coisas (IoT), uma lâmpada inteligente e uma torradeira inteligente. Durante a produção, cada dispositivo recebe um certificado de entidade final para que possa se comunicar de forma segura pela Internet com o fabricante. A PKI da empresa também protege sua infraestrutura de computação, incluindo o site interno e vários serviços de computação auto-hospedados que executam operações financeiras e comerciais.

Consequentemente, a hierarquia de CA modela estreitamente esses aspectos administrativos e operacionais do negócio.



Essa hierarquia contém três raízes, uma para Operações internas e duas para Operações externas (uma CA raiz para cada linha de produto). Ela também ilustra vários comprimentos de caminho de certificação, com dois níveis de CA para Operações internas e três níveis para Operações externas.

O uso de CAs raiz separadas e camadas de CAs subordinadas adicionais no lado de Operações externas é uma decisão de design que atende às necessidades de negócios e de segurança. Com várias árvores de CA, a PKI tem um futuro comprovado contra reorganizações corporativas, alienações ou aquisições. Quando ocorrem alterações, uma hierarquia de CA raiz inteira pode ser movida de forma limpa com a divisão que ela protege. E com dois níveis de CA subordinada, as CAs raiz têm um alto nível de isolamento das CAs de nível 3 que são responsáveis pela assinatura em massa dos certificados para milhares ou milhões de itens fabricados.

No lado interno, as operações corporativas da Web e de computação interna completam uma hierarquia de dois níveis. Esses níveis permitem que administradores da Web e engenheiros de operações gerenciem a emissão de certificados de forma independente para seus próprios domínios de trabalho. A compartimentação da PKI em domínios funcionais distintos é uma prática recomendada de segurança e protege cada um de um comprometimento que pode afetar o outro. Os administradores da Web emitem certificados de entidade final para uso por navegadores da Web em toda a empresa, autenticando e criptografando comunicações no site interno. Os engenheiros de operações emitem certificados de entidade final que autenticam hosts de datacenter e serviços de

computação entre si. Esse sistema ajuda a manter os dados confidenciais seguros, criptografando-os na LAN.

Definir restrições de comprimento no caminho de certificação

A estrutura de uma hierarquia de CA é definida e aplicada pela extensão das restrições básicas que cada certificado contém. A extensão define duas restrições:

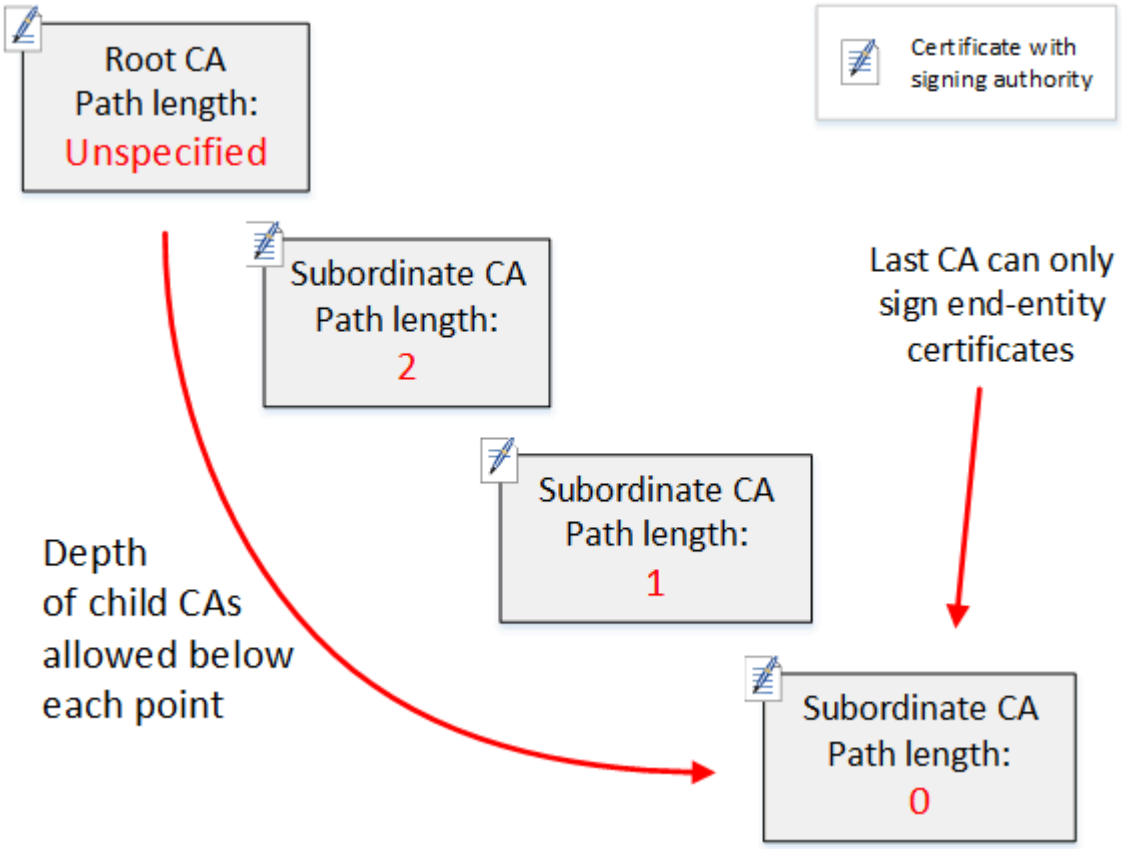
- `cA`: se o certificado definir uma CA. Se esse valor for falso (o padrão), o certificado será um certificado de entidade final.
- `pathLenConstraint`: o número máximo de CAs subordinadas de nível inferior que podem existir em uma cadeia de confiança válida. O certificado da entidade final não conta, porque não é um certificado de CA.

Um certificado CA raiz precisa de flexibilidade máxima e não inclui uma restrição de comprimento de caminho. Isso permite que a raiz defina um caminho de certificação de qualquer comprimento.

> **Note**
>
> CA privada da AWS limita o caminho de certificação a cinco níveis.

As CAs subordinadas têm valores de `pathLenConstraint` iguais ou superiores a zero, dependendo da localização no posicionamento da hierarquia e dos recursos desejados. Por exemplo, em uma hierarquia com três CAs, nenhuma restrição de caminho é especificada para a CA raiz. A primeira CA subordinada tem um comprimento de caminho de 1 e, portanto, pode assinar as CAs filhas. Cada uma dessas CAs filhas deve necessariamente ter um valor igual a `pathLenConstraint`. Isso significa que elas podem assinar certificados de entidade final, mas não podem emitir certificados CA adicionais. Limitar o poder de criar novas CAs é um controle de segurança importante.

O diagrama a seguir ilustra essa propagação de autoridade limitada pela hierarquia.



Nesta hierarquia de quatro níveis, a raiz é irrestrita (como sempre). Mas a primeira CA subordinada tem um `pathLenConstraint` valor igual a 2, o que limita suas CAs filhas de terem mais de dois níveis de profundidade. Consequentemente, para um caminho de certificação válido, o valor da restrição deve diminuir para zero nos próximos dois níveis. Se um navegador da Web encontrar um certificado de entidade final desta ramificação que tenha um comprimento de caminho superior a quatro, a validação falhará. Esse certificado pode ser o resultado de uma CA criada acidentalmente, de uma CA configurada incorretamente ou de uma emissão não autorizada.

Gerenciar o comprimento do caminho com modelos

CA privada da AWS fornece modelos para a emissão de certificados raiz, subordinados e de entidade final. Esses modelos encapsulam as melhores práticas para os valores de restrições básicas, incluindo o comprimento do caminho. Os modelos incluem o seguinte:

- 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

A API `IssueCertificate` retornará um erro se você tentar criar uma CA com um comprimento de caminho maior ou igual ao comprimento de caminho de seu certificado CA emissor.

Para obter mais informações sobre modelos de certificado, consulte Noções básicas sobre modelos de certificados.

Automatizar a configuração da hierarquia da CA com o AWS CloudFormation

Depois de escolher um design para sua hierarquia de CA, você pode testá-lo e colocá-lo em produção usando um AWS CloudFormation modelo. Para obter um exemplo desse modelo, consulte Declarar uma hierarquia de CA privada no Guia do usuário do AWS CloudFormation .

Gerenciar o ciclo de vida da CA privada

Os certificados CA têm uma vida útil ou período de validade fixo. Quando um certificado CA expira, todos os certificados emitidos direta ou indiretamente por CAs subordinadas abaixo dele na hierarquia de CA tornam-se inválidos. Você pode evitar a expiração do certificado CA planejando com antecedência.

Escolher períodos de validade

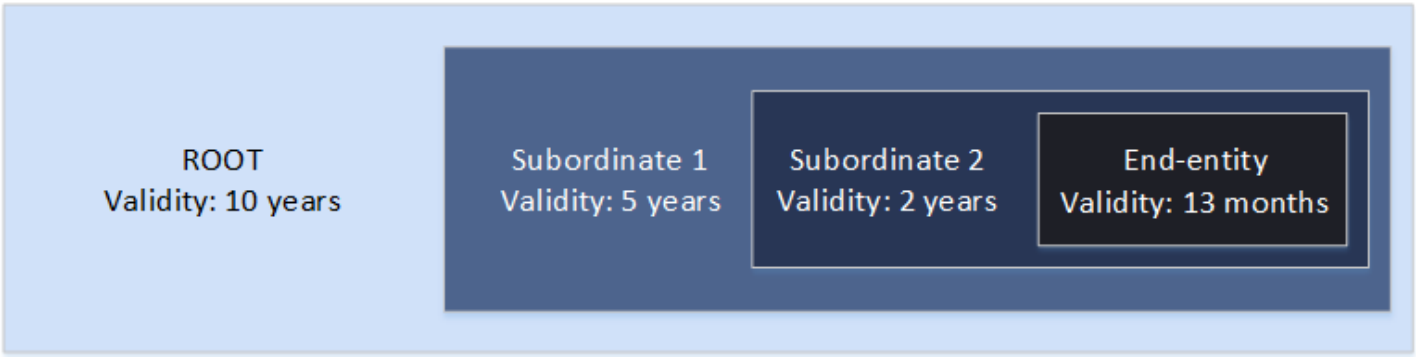
O período de validade de um certificado X.509 é um campo de certificado básico obrigatório. Ele determina o intervalo de tempo durante o qual a CA emissora certifica que o certificado pode ser confiável, sem a revogação. (Um certificado raiz, sendo autoassinado, certifica seu próprio período de validade.)

CA privada da AWS e AWS Certificate Manager auxiliar na configuração dos períodos de validade do certificado, sujeitos às seguintes restrições:

- Um certificado gerenciado por CA privada da AWS deve ter um período de validade menor ou igual ao período de validade da CA que o emitiu. Ou seja, as CAs filhas e os certificados de entidade final não podem sobreviver aos seus certificados pai. A tentativa de usar a API `IssueCertificate` para emitir um certificado CA com um período de validade maior ou igual à CA pai falha.
- Os certificados emitidos e gerenciados por AWS Certificate Manager (aqueles para os quais o ACM gera a chave privada) têm um período de validade de 13 meses (395 dias). O ACM gerencia

o processo de renovação desses certificados. Se você costuma CA privada da AWS emitir certificados diretamente, pode escolher qualquer período de validade.

O diagrama a seguir mostra uma configuração típica de períodos de validade aninhados. O certificado raiz é o mais duradouro; os certificados de entidade final são relativamente curtos; e as autoridades de CAs variam entre esses extremos.



Ao planejar sua hierarquia de CA, determine a vida útil ideal para seus certificados CA. Faça o reverso da vida útil desejada dos certificados de entidade final que você deseja emitir.

Certificados de entidade final

Os certificados de entidade final devem ter um período de validade adequado ao caso de uso. Uma vida útil curta minimiza a exposição de um certificado no caso de sua chave privada ser perdida ou roubada. No entanto, períodos de vida útil curtos significam renovações frequentes. A falha na renovação de um certificado expirado pode resultar em tempo de inatividade.

O uso distribuído de certificados de entidade final também pode apresentar problemas logísticos se houver uma violação de segurança. Seu planejamento deve levar em conta os certificados de renovação e distribuição, a revogação de certificados comprometidos e a rapidez com que as revogações se propagam para clientes que dependem dos certificados.

O período de validade padrão para um certificado de entidade final emitido por meio do ACM é de 13 meses (395 dias). Em CA privada da AWS, você pode usar a `IssueCertificate` API para aplicar qualquer período de validade, desde que seja menor que o da CA emissora.

Certificados CA subordinados

Os certificados CA subordinados devem ter períodos de validade significativamente mais longos do que os dos certificados que eles emitem. Um bom intervalo para a validade de um certificado CA é

duas a cinco vezes o período de qualquer certificado CA filho ou certificado de entidade final que ele emite. Por exemplo, suponha que você tenha uma hierarquia de CA de dois níveis (CA raiz e uma CA subordinada). Se você quiser emitir certificados de entidade final com um tempo de vida útil de um ano, poderá configurar a vida útil da CA emissora subordinada para que seja de três anos. Esse é o período de validade padrão para um certificado CA subordinado em CA privada da AWS. Certificados CA subordinados podem ser alterados sem substituir o certificado CA raiz.

Certificados CA raiz

As alterações em um certificado CA raiz afetam toda a PKI (infraestrutura de chave pública) e exigem que você atualize todos os armazenamentos de confiança dependentes do sistema operacional cliente e do navegador. Para minimizar o impacto operacional, você deve escolher um período de validade longo para o certificado raiz. O CA privada da AWS padrão para certificados raiz é de dez anos.

Gerenciar a sucessão da CA

Você tem duas maneiras de gerenciar a sucessão da CA: substituir a CA antiga ou reemitir a CA com um novo período de validade.

Substituir uma CA antiga

Para substituir uma CA antiga, crie uma nova CA e encadeie-a à mesma CA pai. Depois disso, você emite certificados na nova CA.

Os certificados emitidos na nova CA têm uma nova cadeia de CA. Depois que a nova CA está estabelecida, é possível desabilitar a CA antiga para impedir a emissão de novos certificados. Enquanto desabilitada, a CA antiga oferece suporte para a revogação de certificados antigos emitidos pela CA e, se configurada para isso, continua a validar certificados por meio de OCSP e/ou listas de certificados revogados (CRLs). Quando o último certificado emitido da CA antiga expirar, você poderá excluir a CA antiga. É possível gerar um relatório de auditoria para todos os certificados emitidos pela CA para confirmar que todos os certificados emitidos expiraram. Se a CA antiga tiver CAs subordinadas, você também deve substituí-las, porque as CAs subordinadas expiram ao mesmo tempo ou antes da CA pai. Comece substituindo a CA superior na hierarquia que precisa ser substituída. Depois, crie novas CAs subordinadas de substituição em cada nível inferior subsequente.

AWS recomenda que você inclua um identificador de geração de CA nos nomes das CAs, conforme necessário. Por exemplo, suponha que você nomeie a CA de primeira geração como “CA raiz

corporativa". Quando você cria a CA de segunda geração, nomeie-a como "CA raiz corporativa G2". Essa convenção de nomenclatura simples pode ajudar a evitar confusão quando as duas CAs não expiram.

Esse método de sucessão de CA é preferido porque ele alterna a chave privada da CA. Alternar a chave privada é uma prática recomendada para chaves de CA. A frequência de alternância deve ser proporcional à frequência de utilização da chave: as CAs que emitem mais certificados devem ser alternadas com maior frequência.

> **Note**
>
> Certificados privados emitidos por meio do ACM não podem ser renovados se você substituir a CA. Se utilizar o ACM para emissão e renovação, você precisará emitir novamente o certificado CA para prolongar a vida útil da CA.

Reemitir uma CA antiga

Quando uma CA está se aproximando do vencimento, um método alternativo de prolongar sua vida útil é reemitir o certificado de CA com uma nova data de expiração. A reemissão mantém todos os metadados da CA em vigor e preserva as chaves públicas e privadas existentes. Nesse cenário, a cadeia de certificados existente e os certificados de entidade final não expirados emitidos pela CA permanecem válidos até expirarem. A emissão de novos certificados também pode continuar sem interrupção. Para atualizar uma CA com um certificado reemitido, siga os procedimentos de instalação comuns descritos em Criar e instalar o certificado de CA.

> **Note**
>
> Recomendamos substituir uma CA expirada em vez de reemitir seu certificado devido às vantagens de segurança obtidas com a alternância para um novo par de chaves.

Revogar uma CA

É possível revogar uma CA revogando seu certificado subjacente. Isso também revoga efetivamente todos os certificados emitidos por essa CA. As informações de revogação são distribuídas aos clientes via OCSP ou CRL. Revogue um certificado de CA somente se quiser revogar efetivamente todos os seus certificados de CA subordinada ou de entidade final emitidos.

Configurar um método de revogação de certificado

Ao planejar sua PKI privada com CA privada da AWS, você deve considerar como lidar com situações em que não deseja mais que os endpoints confiem em um certificado emitido, como quando a chave privada de um endpoint é exposta. As abordagens comuns a esse problema são usar certificados de curta duração ou configurar a revogação de certificados. Certificados de curta duração expiram em um período tão curto, em horas ou dias, que a revogação não faz sentido, com o certificado tornando-se inválido aproximadamente no mesmo tempo necessário para notificar um endpoint sobre a revogação. Esta seção descreve as opções de revogação para cliente do CA privada da AWS , incluindo configuração e práticas recomendadas.

Clientes à procura de um método de revogação podem escolher o Online Certificate Status Protocol (OCSP), listas de revogação de certificados (CRLs) ou ambos.

> **Note**
>
> Se você criar uma CA sem configurar a revogação, sempre poderá configurá-la mais tarde. Para ter mais informações, consulte Atualizar a CA privada.

- Online Certificate Status Protocol (OCSP)

  CA privada da AWS fornece uma solução OCSP totalmente gerenciada para notificar os endpoints de que os certificados foram revogados sem a necessidade de os próprios clientes operarem a infraestrutura. Os clientes podem habilitar o OCSP em CAs novas ou existentes com uma única operação usando o CA privada da AWS console, a API, a CLI ou por meio de. AWS CloudFormation Embora CRLs sejam armazenadas e processadas no endpoint e possam se tornar obsoletas, os requisitos de armazenamento e processamento do OCSP são tratados de forma síncrona no back-end do agente de resposta.

  Quando você habilita o OCSP para uma CA, CA privada da AWS inclui a URL do respondente OCSP na extensão de Acesso às Informações da Autoridade (AIA) de cada novo certificado emitido. A extensão permite que clientes, como navegadores da Web, consultem o agente de resposta e determinem se uma entidade final ou um certificado de CA subordinada pode ser confiável. O agente de reposta retorna uma mensagem de status assinada criptograficamente para garantir sua autenticidade.

  O respondente CA privada da AWS OCSP é compatível com o RFC 5019.

Considerações sobre OCSP

- As mensagens de status do OCSP são assinadas usando o mesmo algoritmo de assinatura que a CA emissora foi configurada para usar. As CAs criadas no console do CA privada da AWS usam o algoritmo de assinatura SHA256WITHRSA por padrão. Outros algoritmos compatíveis podem ser encontrados na documentação CertificateAuthorityConfigurationda API.
- Os modelos de certificado APIPassthrough e CSRPassthrough não funcionarão com a extensão AIA se o respondente OCSP estiver habilitado.
- O endpoint do serviço OCSP gerenciado pode ser acessado na Internet pública. Os clientes que desejarem o OCSP, mas preferem não ter um endpoint público, precisarão operar sua própria infraestrutura OCSP.

- Listas de revogação de certificados (CRLs)

  Uma CRL contém uma lista de certificados revogados. Quando você configura uma CA para gerar CRLs, CA privada da AWS inclui a extensão CRL Distribution Points em cada novo certificado emitido. Essa extensão fornece o URL da CRL. A extensão permite que clientes, como navegadores da Web, consultem a CRL e determinem se uma entidade final ou um certificado de CA subordinada pode ser confiável.

Como um cliente precisa baixar CRLs e processá-las localmente, seu uso consome mais memória em comparação ao OCSP. CRLs podem consumir menos largura de banda da rede, pois a lista de CRLs é baixada e armazenada em cache, em comparação com o OCSP, que verifica o status de revogação para cada nova tentativa de conexão.

> **Note**
>
> Tanto o OCSP quanto as CRLs apresentam algum atraso entre a revogação e a disponibilidade da mudança de status.
>
> - As respostas do OCSP podem levar até 60 minutos para refletir o novo status quando um certificado é revogado. Em geral, o OCSP tende a oferecer suporte à distribuição mais rápida de informações de revogação pois, ao contrário das CRLs, que podem ser armazenadas em cache pelos clientes por vários dias, as respostas do OCSP normalmente não são armazenadas em cache pelos clientes.

- Uma CRL normalmente é atualizada aproximadamente 30 minutos depois que um certificado é revogado. Se por algum motivo uma atualização da CRL falhar, CA privada da AWS faça novas tentativas a cada 15 minutos.

Requisitos gerais para configurações de revogação

Os seguintes requisitos se aplicam a todas as configurações de revogação.

- Uma configuração que desabilita CRLs ou OCSP deve conter somente o parâmetro `Enabled=False` e falhará se outros parâmetros, como`CustomCname` ou `ExpirationInDays`, forem incluídos.
- Em uma configuração de CRL, o parâmetro `S3BucketName` deve estar em conformidade com as regras de nomenclatura de bucket do Amazon Simple Storage Service.
- Uma configuração contendo um parâmetro de nome canônico (CNAME) personalizado para CRLs ou OCSP deve estar em conformidade com as restrições RFC7230 sobre o uso de caracteres especiais em um CNAME.
- Em uma configuração de CRL ou OCSP, o valor de um parâmetro CNAME não deve incluir um prefixo de protocolo como "http://" ou "https://".

Tópicos

- Planejar uma lista de revogação de certificados (CRL)
- Configurar um URL personalizado para OCSP da CA privada da AWS

Planejar uma lista de revogação de certificados (CRL)

Antes de configurar uma CRL como parte do processo de criação da CA, talvez seja necessária uma certa configuração prévia. Esta seção explica os pré-requisitos e as opções que você deve entender antes de criar uma CA com uma CRL anexada.

Para obter informações sobre como usar o Online Certificate Status Protocol (OCSP) como alternativa ou suplemento a uma CRL, consulte e Opções de revogação de certificados e .Configurar um URL personalizado para OCSP da CA privada da AWS

Tópicos

- Estrutura da CRL

- Políticas de acesso para CRLs no Amazon S3
- Habilitando o S3 Block Public Access (BPA) com CloudFront
- Criptografar as CRLs
- Determinando o URI do ponto de distribuição da CRL (CDP)

Estrutura da CRL

Cada CRL é um arquivo DER codificado. Para fazer download do arquivo e usar o OpenSSL para visualizá-lo, use um comando como o seguinte:

```
openssl crl -inform DER -in path-to-crl-file -text -noout
```

As CRLs têm o formato a seguir:

```
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**
>
> A CRL só será depositada no Amazon S3 depois que um certificado que se refere a ela for emitido. Antes disso, só haverá um arquivo `acm-pca-permission-test-key` visível no bucket do Amazon S3.

Políticas de acesso para CRLs no Amazon S3

Se você planeja criar uma CRL, precisa preparar um bucket do Amazon S3 para armazená-la. CA privada da AWS deposita automaticamente a CRL no bucket do Amazon S3 que você designar e a atualiza periodicamente. Para mais informações, consulte Criar um bucket.

Seu bucket do S3 deve ser protegido por uma política de permissões do IAM anexada. Usuários autorizados e entidades principais de serviços precisam da permissão `Put` para permitir que o CA privada da AWS coloque objetos no bucket e da permissão `Get` para recuperá-los. Durante o procedimento do console para criar uma CA, você pode optar por permitir a CA privada da AWS criação de um novo bucket e aplicar uma política de permissões padrão.

> **Note**
>
> A configuração da política do IAM depende dos Regiões da AWS envolvidos. Regiões se encaixam em duas categorias:
>
> - Regiões habilitadas por padrão — Regiões que são habilitadas por padrão para todos. Contas da AWS

- Regiões desabilitadas por padrão: regiões que estão desabilitadas por padrão, mas que podem ser manualmente habilitadas pelo cliente.

Para obter mais informações e uma lista das regiões desativadas por padrão, consulte Gerenciando. Regiões da AWS Para uma discussão sobre entidades principais de serviço no contexto do IAM, consulte Entidades principais os de serviços do AWS em regiões opcionais. Quando você configura CRLs como método de revogação de certificado, CA privada da AWS cria uma CRL e a publica em um bucket do S3. O bucket do S3 exige uma política do IAM que permita que o responsável pelo CA privada da AWS serviço grave no bucket. O nome da entidade principal de serviço varia de acordo com as regiões usadas, e não há suporte para todas as possibilidades.

PCA	S3	Entidade principal do serviço
Ambos na mesma região		`acm-pca.amazonaws.com`
Habilitado	Habilitado	`acm-pca.amazonaws.com`
Desabilitado	Habilitado	`acm-pca.`*`Region`*`.amazonaws.com`
Habilitado	Desabilitado	Não suportado

A política padrão não aplica restrição de `SourceArn` à CA. Recomendamos que você aplique manualmente a política menos permissiva mostrada abaixo, que restringe o acesso a uma AWS conta específica e a uma CA privada específica. Para obter mais informações, consulte Adicionar uma política de bucket usando o console do 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"
            }
         }
      }
   ]
}
```

Se você optar por permitir a política padrão, sempre poderá modificá-la mais tarde.

Habilitando o S3 Block Public Access (BPA) com CloudFront

Novos buckets do Amazon S3 são configurados por padrão com o recurso Bloqueio de acesso público (BPA) ativado. Incluído nas práticas recomendadas de segurança do Amazon S3, o BPA é um conjunto de controles de acesso que os clientes podem utilizar para ajustar o acesso aos objetos em seus buckets do S3 e aos buckets como um todo. Quando o BPA está ativo e configurado corretamente, somente AWS usuários autorizados e autenticados têm acesso a um bucket e seu conteúdo.

AWS recomenda o uso de BPA em todos os buckets do S3 para evitar a exposição de informações confidenciais a possíveis adversários. No entanto, um planejamento adicional é necessário se seus clientes de PKI recuperarem CRLs na Internet pública (ou seja, sem estarem conectados a uma conta). AWS Esta seção descreve como configurar uma solução de PKI privada usando a Amazon CloudFront, uma rede de entrega de conteúdo (CDN), para atender CRLs sem exigir acesso autenticado do cliente a um bucket do S3.

> **Note**
>
> O uso CloudFront incorre em custos adicionais em sua AWS conta. Para obter mais informações, consulte Amazon CloudFront Pricing.
> Se você optar por armazenar sua CRL em um bucket do S3 com o BPA ativado e não usar CloudFront, deverá criar outra solução de CDN para garantir que seu cliente de PKI tenha acesso à sua CRL.

Configurar o Amazon S3 com o BPA

No S3, crie um novo bucket para sua CRL, como de costume, e habilite o BPA nele.

Para configurar um bucket do Amazon S3 que bloqueie o acesso público à sua CRL

1. Crie um novo bucket do S3 usando o procedimento em Criar um bucket. Durante o procedimento, selecione a opção Bloquear todo o acesso público.

   Para obter mais informações, consulte Bloquear o acesso público ao seu armazenamento do Amazon S3.
2. Quando o bucket for criado, escolha seu nome na lista, navegue até a guia Permissões, escolha Editar na seção Propriedade do objeto e selecione Proprietário do bucket preferencial.
3. Também na guia Permissões, adicione uma política do IAM ao bucket, conforme descrito em Políticas de acesso para CRLs no Amazon S3 .

Configurado CloudFront para o BPA

Crie uma CloudFront distribuição que tenha acesso ao seu bucket privado do S3 e possa servir CRLs para clientes não autenticados.

Para configurar uma CloudFront distribuição para a CRL

1. Crie uma nova CloudFront distribuição usando o procedimento em Criar uma distribuição no Amazon CloudFront Developer Guide.

   Ao concluir o procedimento, aplique as configurações a seguir:

   - Em Nome de domínio de origem, escolha o bucket do S3.
   - Escolha Sim para Restringir acesso ao bucket.

- Escolha Criar uma nova identidade para Identidade de acesso à origem.
- Escolha Sim, atualizar política do bucket em Conceder permissões de leitura no bucket.

> **Note**
>
> Neste procedimento, CloudFront modifica sua política de bucket para permitir que ela acesse objetos de bucket. Considere editar essa política para permitir o acesso somente aos objetos na pasta `crl`.

2. Depois que a distribuição for inicializada, localize seu nome de domínio no CloudFront console e salve-o para o próximo procedimento.

> **Note**
>
> Se seu bucket do S3 foi criado recentemente em uma região diferente de us-east-1, você pode receber um erro de redirecionamento temporário de HTTP 307 ao acessar seu aplicativo publicado por meio de. CloudFront Pode demorar várias horas para que o endereço do bucket se propague.

Configurar sua CA para o BPA

Ao configurar sua nova CA, inclua o alias em sua CloudFront distribuição.

Para configurar sua CA com um CNAME para CloudFront

- Crie sua CA usando a Procedimento para criar uma CA (CLI) .

  Quando você executa o procedimento, o arquivo de revogação `revoke_config.txt` deve incluir as seguintes linhas para especificar um objeto CRL não público e fornecer uma URL para o endpoint de distribuição em: CloudFront

```
"S3ObjectAcl":"BUCKET_OWNER_FULL_CONTROL",
 "CustomCname":"abcdef012345.cloudfront.net"
```

  Mais tarde, quando você emitir certificados com essa CA, eles conterão um bloco como o seguinte:

```
X509v3 CRL Distribution Points:
```

```
 Full Name:
 URI:http://abcdef012345.cloudfront.net/crl/01234567-89ab-
cdef-0123-456789abcdef.crl
```

> **Note**
>
> Se houver certificados mais antigos emitidos por essa CA, eles não conseguirão acessar a CRL.

Criptografar as CRLs

Opcionalmente, você pode configurar a criptografia no bucket do Amazon S3 contendo suas CRLs. CA privada da AWS suporta dois modos de criptografia para ativos no Amazon S3:

- Criptografia automática no lado do servidor com chaves AES-256 gerenciadas pelo Amazon S3.
- Criptografia gerenciada pelo cliente usando AWS Key Management Service e AWS KMS key configurada de acordo com suas especificações.

> **Note**
>
> CA privada da AWS não suporta o uso de chaves KMS padrão geradas automaticamente pelo S3.

Os procedimentos a seguir descrevem como configurar cada uma das opções de criptografia.

Como configurar uma criptografia automática

Siga as etapas para habilitar a criptografia do servidor do S3.

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Na tabela Buckets, escolha o bucket que conterá seus CA privada da AWS ativos.
3. Na página do bucket, escolha a guia Propriedades.
4. Escolha o cartão Criptografia padrão.
5. Escolha Habilitar.
6. Escolh Chave do Amazon S3 (SSE-S3).

7. Escolha Salvar alterações.

Como configurar a criptografia personalizada

Siga as etapas para habilitar a criptografia usando uma chave personalizada.

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Na tabela Buckets, escolha o bucket que conterá seus CA privada da AWS ativos.
3. Na página do bucket, escolha a guia Propriedades.
4. Escolha o cartão Criptografia padrão.
5. Escolha Habilitar.
6. Escolha a AWS Key Management Service chave (SSE-KMS).
7. Escolha Escolher entre suas AWS KMS chaves ou Inserir AWS KMS key ARN.
8. Escolha Salvar alterações.
9. (Opcional) Se ainda não tiver uma chave KMS, crie uma usando o seguinte comando da AWS CLI create-key:

   ```
   $ aws kms create-key
   ```

   A saída contém o ID de chave e o nome do recurso da Amazon (ARN) da chave do KMS. Veja a seguir um exemplo de saída:

   ```
   {
       "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. Usando as etapas a seguir, você concede permissão ao responsável pelo CA privada da AWS serviço para usar a chave KMS. Por padrão, todas as chaves do KMS são privadas; somente

o proprietário do recurso pode usá-las para criptografar e descriptografar dados. No entanto, o proprietário do recurso pode conceder permissões para acessar a chave do KMS a outros usuários e recursos. A entidade principal desse serviço deve estar na mesma região em que a chave do KMS está armazenada.

a. Primeiro, salve a política padrão para sua chave KMS `policy.json` usando o seguinte get-key-policycomando:

```
$ aws kms get-key-policy --key-id key-id --policy-name default --output text
 > ./policy.json
```

b. Abra o arquivo `policy.json` em um editor de textos. Selecione uma das declarações de políticas a seguir e adicione-a à política existente.

Se a sua chave de bucket do Amazon S3 estiver habilitada, use a seguinte declaração:

```
{
   "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"
      }
   }
}
```

Se a sua chave de bucket do Amazon S3 estiver desabilitada, use a seguinte declaração:

```
{
   "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 fim, aplique a política atualizada usando o seguinte put-key-policycomando:

```
$ aws kms put-key-policy --key-id key_id --policy-name default --policy file://
policy.json
```

Determinando o URI do ponto de distribuição da CRL (CDP)

Se você usar o bucket do S3 como CDP para sua CA, o URI do CDP poderá estar em um dos formatos a seguir.

- `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`

Se você configurou sua CA com um CNAME personalizado, o URI do CDP incluirá o CNAME, por exemplo, `http://alternative.example.com/crl/CA-ID.crl`

Configurar um URL personalizado para OCSP da CA privada da AWS

> **Note**
>
> Esse tópico destina-se a clientes que desejam personalizar a URL pública do endpoint do agente de resposta OCSP para fins de identidade visual ou outros fins. Se você planeja usar

a configuração padrão do OCSP CA privada da AWS gerenciado, pode ignorar este tópico e seguir as instruções de configuração em Configurar revogação.

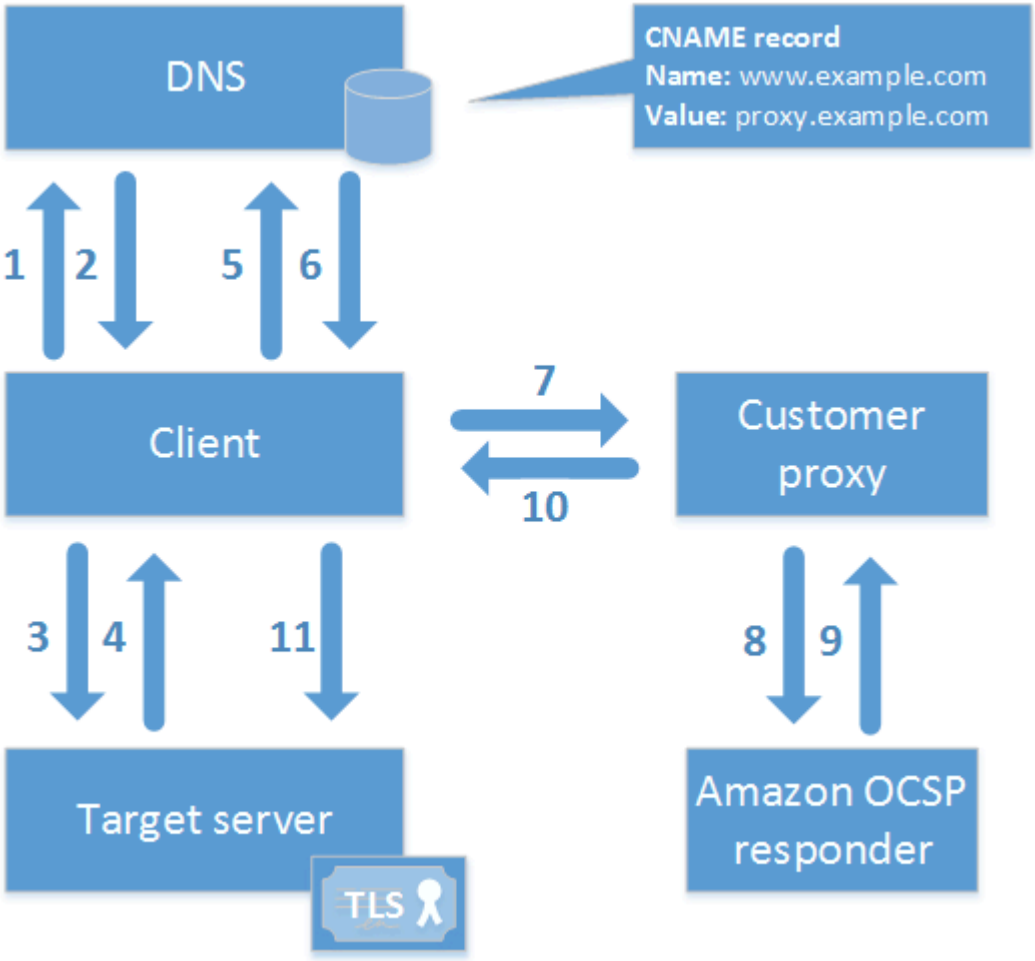
Por padrão, quando você habilita o OCSP para CA privada da AWS, cada certificado emitido contém a URL do respondente AWS OCSP. Isso permite que os clientes que solicitam uma conexão criptograficamente segura enviem consultas de validação OCSP diretamente à AWS. Porém, em alguns casos, pode ser preferível indicar uma URL diferente em seus certificados e, ao mesmo tempo, enviar consultas OCSP à AWS.

> **Note**
>
> Para obter informações sobre como usar uma lista de revogação de certificados (CRL) como alternativa ou suplemento ao OCSP, consulte Configurar a revogação e Planejar uma lista de revogação de certificados (CRL).

Três elementos estão envolvidos na configuração de um URL personalizado para o OCSP.

- Configuração da CA: especifique um URL OCSP personalizado no `RevocationConfiguration` para sua CA, conforme descrito no Exemplo 2: criar uma CA com o OCSP e um CNAME personalizado habilitado em Procedimento para criar uma CA (CLI) .
- DNS: adicione um registro CNAME à configuração do domínio para mapear o URL que aparece nos certificados para o URL de um servidor proxy. Para obter mais informações, consulte Exemplo 2: criar uma CA com o OCSP e um CNAME personalizado habilitado em Procedimento para criar uma CA (CLI) .
- Servidor proxy de encaminhamento: configure um servidor proxy que possa encaminhar de maneira transparente o tráfego OCSP recebido ao agente de resposta OCSP da AWS .

O diagrama a seguir ilustra como esses elementos funcionam em conjunto.



Conforme mostrado no diagrama, o processo de validação personalizado do OCSP envolve as etapas a seguir:

1. O cliente consulta o DNS para o domínio de destino.
2. O cliente recebe o IP de destino.
3. O cliente abre uma conexão TCP com o destino.
4. O cliente recebe o certificado TLS de destino.
5. O cliente consulta o DNS para o domínio OCSP listado no certificado.
6. O cliente recebe o IP do proxy.
7. O cliente envia a consulta OCSP para o proxy.
8. O proxy encaminha a consulta ao agente de resposta OCSP.
9. O agente de resposta retorna o status do certificado para o proxy.
10. O proxy encaminha o status do certificado ao cliente.

11. Se o certificado for válido, o cliente iniciará o handshake de TLS.

> **Tip**
>
> Esse exemplo pode ser implementado usando a Amazon CloudFront e o Amazon Route 53 depois de configurar uma CA conforme descrito acima.
>
> 1. Em CloudFront, crie uma distribuição e configure-a da seguinte forma:
>   - Crie um nome alternativo que corresponda ao CNAME personalizado.
>   - Vincule o certificado a ele.
>   - Defina ocsp.acm-pca.*<region>*.amazonaws.com como origem..
>   - Aplique a política `Managed-CachingDisabled`.
>   - Defina Política de protocolo do visualizador para HTTP e HTTPS.
>   - Defina Métodos HTTP permitidos como GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE.
> 2. No Route 53, crie um registro DNS que mapeie seu CNAME personalizado para a URL da CloudFront distribuição.

Modos de autoridades certificadoras

CA privada da AWS suporta a criação de uma CA em qualquer um dos dois modos. Os modos GENERAL_PURPOSE e SHORT_LIVED_CERTIFICATE influenciam o período de validade permitido dos certificados emitidos pela CA.

> **Note**
>
> CA privada da AWS não executa verificações de validade em certificados de CA raiz.

GENERAL_PURPOSE (padrão)

Nesse modo, o CA pode emitir certificados com qualquer período de validade. A maioria das aplicações usa certificados desse tipo. Em geral, a CA também especifica um mecanismo de revogação.

SHORT_LIVED_CERTIFICATE

Esse modo define uma CA que emite exclusivamente certificados com um período máximo de validade de sete dias. Esses certificados de curta duração expiram tão rapidamente que podem ser implantados sem um mecanismo de revogação. Para algumas aplicações, faz mais sentido implantar com frequência certificados de curta duração do que arcar com a sobrecarga da rede e de processamento decorrente da revogação.

CAs com o modo SHORT_LIVED_CERTIFICATE custam menos que CAs de uso geral. Para obter mais informações, consulte Preços do AWS Private Certificate Authority.

Para criar uma CA que emita certificados de curta duração, defina o `UsageMode` parâmetro como SHORT_LIVED_CERTIFICATE usando o AWS CLIprocedimento para criar uma CA.

> **Note**
>
> AWS Certificate Manager não pode emitir certificados assinados por uma CA privada com modo de vida curta.

O uso de certificados de curta duração tem suporte pelos seguintes serviços da AWS :

- Amazon AppStream
- Amazon WorkSpaces

Planejamento da resiliência

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de datacenter tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Redundância e recuperação de desastres

Considere a redundância e o DR ao planejar sua hierarquia de CA. CA privada da AWS está disponível em várias regiões, o que permite criar CAs redundantes em várias regiões. O CA privada da AWS serviço opera com um acordo de nível de serviço (SLA) de disponibilidade de 99,9%. Há pelo menos duas abordagens que podem ser consideradas para redundância e recuperação de desastres. Você pode configurar a redundância na CA raiz ou na CA subordinada mais alta. Cada abordagem tem prós e contras.

1. Você pode criar duas CAs raiz em duas AWS regiões diferentes para redundância e recuperação de desastres. Com essa configuração, cada CA raiz opera de forma independente em uma AWS região, protegendo você no caso de um desastre em uma única região. No entanto, a criação de CAs raiz redundantes aumenta a complexidade operacional: você precisará distribuir os dois certificados CA raiz para os armazenamentos confiáveis de navegadores e sistemas operacionais em seu ambiente.
2. Você também pode criar CAs subordinadas redundantes para implantação em cada uma das suas regiões da AWS e encadeá-las à mesma CA raiz exclusiva em uma única região da AWS . O benefício dessa abordagem é que você precisa distribuir apenas um único certificado CA raiz para os armazenamentos de confiança em seu ambiente. A limitação é que você não tem uma CA raiz redundante no caso de um desastre que afete a AWS região na qual sua CA raiz existe.

Melhores práticas do CA privada da AWS

As melhores práticas são recomendações que podem ajudá-lo a usar o CA privada da AWS de forma eficaz. As melhores práticas a seguir são baseadas em experiência real de clientes atuais do AWS Certificate Manager e do CA privada da AWS.

Documentar a estrutura e as políticas da CA

A AWS recomenda documentar todas as suas políticas e práticas para operar sua CA. Isso pode incluir:

- O motivo de suas decisões sobre a estrutura da CA
- Um diagrama que mostra suas CAs e suas relações
- Políticas sobre períodos de validade da CA
- Planejamento da sucessão da CA
- Políticas sobre o comprimento do caminho
- Catálogo de permissões
- Descrição das estruturas de controle administrativo
- Segurança

Você pode capturar essas informações em dois documentos, conhecidos como Política de certificação (CP) e Declaração de práticas de certificação (CPS). Consulte a RFC 3647 para obter uma estrutura de trabalho para capturar informações importantes sobre as operações de sua CA.

Minimizar o uso da CA raiz, se possível

Uma CA raiz deve, em geral, ser usada apenas para emitir certificados para CAs intermediárias. Isso permite que a CA raiz seja armazenada fora de perigo, enquanto as CAs intermediárias executam a tarefa diária de emitir certificados de entidade final.

No entanto, se a prática atual da sua organização for emitir certificados de entidade final diretamente de uma CA raiz, o CA privada da AWS poderá ser compatível com esse fluxo de trabalho enquanto melhora os controles operacionais e de segurança. A emissão de certificados de entidade final nesse

cenário requer uma política de permissões do IAM que permita que a CA raiz use um modelo de certificado de entidade final. Para obter mais informações sobre políticas do IAM, consulte Identity and Access Management (IAM) para AWS Private Certificate Authority.

> **Note**
>
> Essa configuração impõe limitações que podem resultar em desafios operacionais. Por exemplo, se a CA raiz for comprometida ou perdida, você deverá criar uma nova CA raiz e distribuí-la a todos os clientes em seu ambiente. Até que esse processo de recuperação esteja concluído, você não poderá emitir novos certificados. A emissão de certificados diretamente de uma CA raiz também impede que você restrinja o acesso e limite o número de certificados emitidos da raiz, que são ambos considerados melhores práticas para gerenciar uma CA raiz.

Forneça sua própria CA raiz Conta da AWS

Criar uma CA raiz e uma CA subordinada em duas contas diferentes da AWS é uma prática recomendada. Isso pode fornecer proteção adicional e controles de acesso para sua CA raiz. Você pode fazer isso exportando a CSR da CA subordinada em uma conta e assinando-a com uma CA raiz em uma conta diferente. O benefício dessa abordagem é que você pode separar o controle de suas CAs por conta. A desvantagem é que você não pode usar o assistente do AWS Management Console para simplificar o processo de assinatura do certificado CA de uma CA subordinada de sua CA raiz.

> **Important**
>
> É altamente recomendável usar a autenticação multifator (MFA) sempre que você acessar o CA privada da AWS.

Perfis separados de administrador e emissor

O perfil de administrador de CA deve ser separado dos usuários que precisam apenas emitir certificados de entidade final. Se o administrador da CA e o emissor do certificado residirem no mesmo localConta da AWS, você pode limitar as permissões do emissor criando um usuário do IAM especificamente para essa finalidade.

Implementar a revogação gerenciada de certificados

A revogação gerenciada notifica automaticamente os clientes de certificados quando um certificado é revogado. Talvez seja necessário revogar um certificado quando suas informações criptográficas foram comprometidas ou quando ele foi emitido por engano. Em geral, os clientes se recusam a aceitar certificados revogados. O CA privada da AWS oferece duas opções padrão para revogação gerenciada: Online Certificate Status Protocol (OCSP) e listas de revogação de certificados (CRLs). Para ter mais informações, consulte Configurar um método de revogação de certificado.

Ativar o AWS CloudTrail

Ative o CloudTrail registro antes de criar e começar a operar uma CA privada. Com CloudTrail, você pode recuperar um histórico de chamadas de AWS API para sua conta para monitorar suas AWS implantações. Esse histórico inclui as chamadas de API feitas pelo AWS Management Console, pelos SDKs da AWS, pela AWS Command Line Interface e pelos serviços da AWS de nível superior. Você também pode identificar quais usuários e contas chamaram as operações de API do PCA, o endereço IP de origem do qual as chamadas foram feitas e quando elas ocorreram. Você pode se CloudTrail integrar aos aplicativos usando a API, automatizar a criação de trilhas para sua organização, verificar o status de suas trilhas e controlar como os administradores ativam e desativam o CloudTrail login. Para obter mais informações, consulte Criação de uma trilha. Acesse Usando CloudTrail para ver trilhas de exemplo para operações do CA privada da AWS.

Alternar a chave privada da CA

É uma melhor prática usada para atualizar periodicamente a chave privada da sua CA privada. Você pode atualizar uma chave importando um novo certificado CA ou substituir a CA privada por uma nova CA.

> **Note**
>
> Se você substituir a CA propriamente dita, lembre-se de que o ARN dessa CA mudará. Isso causará uma falha na automação baseada em um ARN com codificação rígida.

Excluir CAs não utilizadas

Você pode excluir permanentemente uma CA privada. Talvez você deva fazer isso se não precisar mais da CA ou se desejar substituí-la por uma CA com uma nova chave privada. Para excluir com segurança uma CA, recomendamos que você siga o processo descrito em Excluir a CA privada.

> **Note**
>
> A AWS cobra por uma CA até que ela tenha sido excluída.

Bloqueie o acesso público às suas CRLs

O CA privada da AWS recomenda usar o recurso de Bloqueio de Acesso Público (BPA) do Amazon S3 em buckets que contêm CRLs. Isso evita a exposição desnecessária de detalhes da sua PKI privada a possíveis adversários. O BPA é uma prática recomendada do S3 e está habilitado por padrão em novos buckets. Em alguns casos, uma configuração adicional é necessária. Para ter mais informações, consulte Habilitando o S3 Block Public Access (BPA) com CloudFront.

Práticas recomendadas para aplicações do Amazon EKS

Ao usar o CA privada da AWS para provisionar o Amazon EKS com certificados X.509, siga as recomendações para proteger ambientes de multilocatários nos Guias de melhores práticas do Amazon EKS. Para obter informações gerais sobre a integração do CA privada da AWS com o Kubernetes, consulte Proteger o Kubernetes com o CA privada da AWS.

Administração de CAs privadas

Usando CA privada da AWS, você pode criar uma hierarquia totalmente AWS hospedada de autoridades de certificação (CAs) raiz e subordinadas para uso interno de sua organização. Para gerenciar a revogação de certificados, você pode habilitar o Online Certificate Status Protocol (OCSP), as listas de revogação de certificados (CRLs) ou ambos. CA privada da AWS armazena e gerencia seus certificados CA, CRLs e respostas OCSP, e as chaves privadas de suas autoridades raiz são armazenadas com segurança pelo. AWS

> **Note**
>
> A implementação do OCSP em CA privada da AWS não suporta extensões de solicitação OCSP. Se você enviar uma consulta em lote OCSP contendo vários certificados, o respondente AWS OCSP processará somente o primeiro certificado na fila e eliminará os outros. Uma revogação pode demorar até uma hora para aparecer nas respostas do OCSP.

Você pode acessar CA privada da AWS usando a AWS Management Console AWS CLI, a e a CA privada da AWS API. Os tópicos a seguir mostram como usar o console e a CLI. Para saber mais sobre a API, consulte Referência de APIs do AWS Private Certificate Authority. Para exemplos de Java que mostram como usar a API, consulte Usar a API do CA privada da AWS (exemplos Java).

Tópicos

Criar uma CA privada

É possível usar os procedimentos nesta seção para criar CAs raiz ou CAs subordinadas, resultando em uma hierarquia auditável de relacionamentos de confiança que corresponda às suas necessidades organizacionais. Você pode criar uma CA usando AWS Management Console a parte PCA do AWS CLI, ou AWS CloudFormation.

Para obter informações sobre como atualizar a configuração de uma CA que você já criou, consulte Atualizar a CA privada.

Para obter informações sobre como usar uma CA para assinar certificados de entidade final para seus usuários, dispositivos e aplicações, consulte Emitir certificados de entidade final privada.

> **Note**
>
> Um preço mensal é cobrado em sua conta por cada CA privada começando no momento em que ela é criada.
> Para obter as informações mais recentes sobre CA privada da AWS preços, consulte AWS Private Certificate Authority Preços. Você também pode usar a Calculadora de preços da AWS para estimar os custos.

Tópicos

Procedimento para criar uma CA (console)

Conclua as estas etapas para criar uma CA privada usando o AWS Management Console.

Para começar a usar o console

Faça login na sua AWS conta e abra o CA privada da AWS console em**`https://console.aws.amazon.com/acm-pca/home`**.

- Se estiver abrindo o console em uma região em que não possui CAs privadas, a página introdutória será exibida. Escolha Criar uma CA privada.

- Se estiver abrindo o console em uma região em que já criou uma CA, a página Autoridades de certificação privadas será aberta com uma lista das suas CAs. Escolha Criar CA.

Opções de modos

Na seção Opções de modos do console, escolha o modo de expiração dos certificados emitidos pela sua CA.

- Uso geral: emite certificados que podem ser configurados com qualquer data de expiração. Esse é o padrão.
- Certificado de curta duração: emite certificados com um período máximo de validade de sete dias. Em alguns casos, um curto período de validade pode substituir, um mecanismo de revogação.

Opções de tipos de CA

Na seção Opções de tipo do console, escolha o tipo de autoridade de certificação privada que você deseja criar.

- A escolha da Raiz estabelece uma nova hierarquia de CA. Essa CA é baseada em um certificado autoassinado. Ele serve como autoridade de assinatura final para outras CAs e certificados de entidade final na hierarquia.
- A escolha de uma Subordinada cria uma CA que deve ser assinada por uma CA pai acima dela na hierarquia. As CAs subordinadas normalmente são usadas para criar outras CAs subordinadas ou para emitir certificados de entidade final para usuários, computadores e aplicações.

> **Note**
>
> CA privada da AWS fornece um processo de assinatura automatizado quando a CA principal de sua CA subordinada também é hospedada pela CA privada da AWS. Você só precisa escolher a CA principal a ser usada.
> Talvez sua CA subordinada precise ser assinada por um provedor externo de serviços de confiança. Nesse caso, CA privada da AWS fornece uma solicitação de assinatura de certificado (CSR) que você deve baixar e usar para obter um certificado CA assinado. Para ter mais informações, consulte Instalar um certificado de CA subordinada assinado por uma CA pai externa.

Opções de nomes distintos de requerentes

Em Opções de nome distinto do requerente, configure o nome do requerente da CA privada. É necessário inserir um valor para pelo menos uma das seguintes opções:

- Organização (O): por exemplo, o nome de uma empresa
- Unidade organizacional (UO): por exemplo, uma divisão dentro de uma empresa
- Nome do país (C): código do país com duas letras
- Nome do estado ou província: nome completo de um estado ou província
- Nome da localidade: o nome de uma cidade
- Nome comum (CN) — Uma string legível por humanos para identificar a CA.

> **Note**
>
> É possível personalizar ainda mais o nome do requerente de um certificado, aplicando um modelo APIPassthrough no momento da emissão. Para obter informações e um exemplo detalhado, consulte Emita um certificado com um nome de requerente personalizado usando um modelo APIPassthrough.

Como o certificado de suporte é autoassinado, as informações de requerente que você fornece para uma CA privada provavelmente são mais escassas do que as que uma CA pública conteria. Para obter mais informações sobre cada um dos valores que compõem um nome distinto de objeto, consulte RFC 5280.

Opções de algoritmos de chave

Em Opções de algoritmos de chave, escolha o algoritmo de chave e o tamanho de bits da chave. O valor padrão é um algoritmo RSA com um comprimento de chave de 2048 bits. É possível escolher entre os seguintes algoritmos:

- RSA 2048
- RSA 4096
- ECDSA P256
- ECDSA P384

Opções de revogação de certificados

Em Opções de revogação de certificados, você pode selecionar entre dois métodos de compartilhamento do status de revogação com clientes que usam seus certificados:

- Ativar distribuição de CRL
- Ativar OCSP

É possível configurar uma, nenhuma ou ambas as opções de revogação para a sua CA. Embora opcional, a revogação gerenciada é recomendada como melhor prática. Antes de concluir essa etapa, consulte Configurar um método de revogação de certificado para obter informações sobre as vantagens de cada método, a configuração preliminar que talvez seja necessária e recursos adicionais de revogação.

> **Note**
>
> Se você criar uma CA sem configurar a revogação, sempre poderá configurá-la mais tarde. Para ter mais informações, consulte Atualizar a CA privada.

Para configurar uma CRL

1. Em Opções de revogação de certificados, escolha Ativar distribuição de CRL.
2. Para criar um bucket do Amazon S3 para as entradas da sua CRL, selecione Criar um novo bucket do S3 e digite um nome de bucket exclusivo. (Você não precisa incluir o caminho para o bucket.) Caso contrário, em URI de bucket do S3, escolha um bucket existente na lista.

   Quando você cria um novo bucket por meio do console, o CA privada da AWS tenta anexar a política de acesso necessária ao bucket e desabilitar a configuração padrão de Bloqueio de acesso público (BPA) do S3. Em vez disso, se você especificar um bucket existente, deverá garantir que o BPA esteja desabilitado para a conta e o bucket. Caso contrário, a operação de criação da CA falhará. Se a CA for criada com sucesso, você ainda deverá anexar manualmente uma política a ela antes de começar a gerar CRLs. Use um dos padrões de política descritos em Políticas de acesso para CRLs no Amazon S3 . Para obter mais informações, consulte Adicionar uma política de bucket usando o console do Amazon S3.

> **Important**
>
> Uma tentativa de criar uma CA usando o CA privada da AWS console falhará se todas as condições a seguir se aplicarem:
>
> - Você está configurando uma CRL.
> - Você solicita CA privada da AWS a criação automática de um bucket do S3.
> - Você está aplicando configurações de BPA no S3.
>
> Nessa situação, o console cria um bucket, mas tenta e não consegue torná-lo publicamente acessível. Verifique as configurações do Amazon S3 se isso ocorrer, desabilite o BPA conforme necessário e repita o procedimento para criar uma CA. Para obter mais informações, consulte Bloquear o acesso público ao seu armazenamento do Amazon S3.

3. Expanda Configurações de CRL para obter opções de configuração adicionais.

   - Adicione um Nome de CRL personalizado para criar um alias para o bucket do Amazon S3. Esse nome está contido em certificados emitidos pela CA na extensão “Pontos de distribuição de CRL” definida pela RFC 5280.
   - Digite a Validade em dias em que sua CRL permanecerá válida. O valor padrão é de 7 dias. Para CRLs online, um período de validade de dois a sete dias é comum. O CA privada da AWS tenta gerar novamente a CRL no ponto médio do período especificado.

4. Expanda Configurações do S3 para a configuração opcional de Versionamento de bucket e Registro em log de acesso ao bucket.

Para configurar o OCSP

1. Em Opções de revogação de certificados, escolha Ativar OCSP.
2. No campo Endpoint do OCSP personalizado - opcional, é possível fornecer um nome de domínio totalmente qualificado (FQDN) para um endpoint do OCSP que não seja da Amazon.

   Quando você fornece um FQDN nesse campo, CA privada da AWS insere o FQDN na extensão Authority Information Access de cada certificado emitido no lugar do URL padrão do respondente OCSP. AWS Quando um endpoint recebe um certificado contendo o FQDN

personalizado, ele consulta esse endereço para obter uma resposta do OCSP. Para que esse mecanismo funcione, você precisa fazer duas ações adicionais:

- Use um servidor proxy para encaminhar o tráfego que chega ao seu FQDN personalizado para o respondente AWS OCSP.
- Adicionar um registro CNAME correspondente ao seu banco de dados DNS.

> **Tip**
>
> Para obter mais informações sobre a implementação de uma solução OCSP completa usando um CNAME personalizado, consulte Configurar um URL personalizado para OCSP da CA privada da AWS.

Por exemplo, aqui está um registro CNAME para OCSP personalizado exibido no Amazon Route 53.

Nome de registro	Tipo	Política de roteamento	Diferenciador	Valor/Encaminhar tráfego para
alternative.example.com	CNAME	Simples	-	proxy.example.com

> **Note**
>
> O valor do CNAME não deve incluir um prefixo de protocolo como “http://” ou “https://”.

Adicionar tags

Na página Adicionar etiquetas, é possível marcar sua CA. As tags são pares chave-valor que servem como metadados para identificar e organizar recursos da AWS . Para obter uma lista de parâmetros de CA privada da AWS tags e instruções sobre como adicionar tags às CAs após a criação, consulteGerenciamento de etiquetas para sua CA privada.

> **Note**
>
> Para anexar etiquetas a uma CA privada durante o procedimento de criação, um administrador de CA deve primeiro associar uma política do IAM interna à ação `CreateCertificateAuthority` e permitir explicitamente a marcação. Para ter mais informações, consulte Tag-on-create: Anexando tags a uma CA no momento da criação.

Opções dde permissões de CA

Nas opções de permissões da CA, você pode, opcionalmente, delegar permissões de renovação automática ao responsável pelo AWS Certificate Manager serviço. O ACM só poderá renovar automaticamente os certificados de entidade final privada gerados por essa CA se essa permissão for concedida. Você pode atribuir permissões de renovação a qualquer momento com a CA privada da AWS CreatePermissionAPI ou o comando da CLI create-permission.

O padrão é habilitar essas permissões.

> **Note**
>
> AWS Certificate Manager não suporta a renovação automática de certificados de curta duração.

Definição de preço

Em Preços, confirme que você entende os preços de uma CA privada.

> **Note**
>
> Para obter as informações mais recentes sobre CA privada da AWS preços, consulte AWS Private Certificate Authority Preços. Você também pode usar a Calculadora de preços da AWS para estimar os custos.

Criar uma CA

Escolha Criar CA depois de verificar a precisão de todas as informações inseridas. A página de detalhes da CA é aberta e exibe seu status como Certificado pendente.

> **Note**
>
> Na página de detalhes, você pode concluir a configuração da CA escolhendo Ações, Instalar certificado de CA ou pode retornar posteriormente à lista Autoridades de certificação privadas e concluir o procedimento de instalação aplicável:
>
> - Instalar um certificado de CA raiz
> - Instalando um certificado CA subordinado hospedado pelo CA privada da AWS
> - Instalar um certificado de CA subordinada assinado por uma CA pai externa

Procedimento para criar uma CA (CLI)

Use o comando create-certificate-authority para criar uma CA privada. Você deve especificar a configuração da CA (contendo informações do algoritmo e do nome do requerente), a configuração de revogação (se planeja usar o OCSP e/ou uma CRL) e o tipo de CA (raiz ou subordinada). Os detalhes da configuração e da revogação estão contidos em dois arquivos que você fornece como argumentos ao comando. Opcionalmente, você também pode configurar o modo de uso da CA (para emitir certificados padrão ou de curta duração), anexar etiquetas e fornecer um token de idempotência.

Se estiver configurando uma CRL, você deverá ter um bucket Amazon S3 protegido antes de emitir o comando create-certificate-authority. Para ter mais informações, consulte Políticas de acesso para CRLs no Amazon S3 .

O arquivo de configuração da CA especifica as seguintes informações:

- O nome do algoritmo
- O tamanho da chave a ser usada para criar a chave privada da CA
- O tipo de algoritmo de assinatura que a CA usa para assinar
- Informações do assunto X.500

A configuração de revogação do OCSP define um objeto `OcspConfiguration` com as seguintes informações:

- O sinalizador `Enabled` é definido como “true”.
- (Opcional) Um CNAME personalizado declarado como um valor para `OcspCustomCname`.

A configuração de revogação de uma CRL define um objeto `CrlConfiguration` com as seguintes informações:

- O sinalizador `Enabled` é definido como “true”.
- O período de expiração da CRL em dias (o período de validade da CRL).
- O bucket do Amazon S3 que conterá a CRL.
- (Opcional) Um ObjectAcl valor S3 que determina se a CRL está acessível ao público. No exemplo apresentado, o acesso público está bloqueado. Para ter mais informações, consulte Habilitando o S3 Block Public Access (BPA) com CloudFront.
- (Opcional) Um alias CNAME para o bucket do S3 que é incluído em certificados emitidos pela CA. Se a CRL não estiver acessível ao público, isso apontará para um mecanismo de distribuição como a Amazon CloudFront.
- (Opcional) Um `CrlDistributionPointExtensionConfiguration` objeto com as seguintes informações:
  - O `OmitExtension` sinalizador definido como “verdadeiro” ou “falso”. Isso controla se o valor padrão da extensão CDP será gravado em um certificado emitido pela CA. Para obter mais informações sobre a extensão CDP, consulteDeterminando o URI do ponto de distribuição da CRL (CDP) . A CustomCname não pode ser definido se OmitExtension for “verdadeiro”.

> **Note**
>
> É possível habilitar os dois mecanismos de revogação na mesma CA, definindo um objeto `OcspConfiguration` e um objeto `CrlConfiguration` simultaneamente. Se você não fornecer um parâmetro --revocation-configuration, os dois mecanismos serão desabilitados por padrão. Se precisar de suporte para validação de revogação posteriormente, consulte Atualizar uma CA (CLI).

Os exemplos a seguir pressupõem que você tenha configurado seu diretório de configuração `.aws` com uma região, um endpoint e credenciais padrão válidos. Para obter informações sobre como configurar seu AWS CLI ambiente, consulte Configuração e configurações do arquivo de credenciais. Para facilitar a leitura, fornecemos a entrada de configuração e revogação da CA como arquivos JSON nos comandos de exemplo. Modifique os arquivos de exemplo conforme necessário para seu uso.

Todos os exemplos usam o arquivo de configuração `ca_config.txt` a seguir, salvo indicação em contrário.

Arquivo: ca_config.txt

```
{
   "KeyAlgorithm":"RSA_2048",
   "SigningAlgorithm":"SHA256WITHRSA",
   "Subject":{
      "Country":"US",
      "Organization":"Example Corp",
      "OrganizationalUnit":"Sales",
      "State":"WA",
      "Locality":"Seattle",
      "CommonName":"www.example.com"
   }
}
```

Exemplo 1: criar uma CA com o OCSP habilitado

Neste exemplo, o arquivo de revogação ativa o suporte padrão do OCSP, que usa o CA privada da AWS respondente para verificar o status do certificado.

Arquivo: 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
```

Se obtiver êxito, esse comando produz o nome de recurso da Amazon (ARN) da nova 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
```

Se obtiver êxito, esse comando produz o nome de recurso da Amazon (ARN) da CA.

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

Use o comando a seguir para inspecionar a configuração da 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
```

Essa descrição deve conter a seção a seguir.

```
"RevocationConfiguration": {
   ...
   "OcspConfiguration": {
      "Enabled": true
   }
   ...
}
```

Exemplo 2: criar uma CA com o OCSP e um CNAME personalizado habilitado

Neste exemplo, o arquivo de revogação habilita suporte personalizado ao OCSP. O parâmetro `OcspCustomCname` usa um nome de domínio totalmente qualificado (FQDN) como valor.

Quando você fornece um FQDN nesse campo, CA privada da AWS insere o FQDN na extensão Authority Information Access de cada certificado emitido no lugar do URL padrão do respondente OCSP. AWS Quando um endpoint recebe um certificado contendo o FQDN personalizado, ele consulta esse endereço para obter uma resposta do OCSP. Para que esse mecanismo funcione, você precisa fazer duas ações adicionais:

- Use um servidor proxy para encaminhar o tráfego que chega ao seu FQDN personalizado para o respondente AWS OCSP.
- Adicionar um registro CNAME correspondente ao seu banco de dados DNS.

> **Tip**
>
> Para obter mais informações sobre a implementação de uma solução OCSP completa usando um CNAME personalizado, consulte Configurar um URL personalizado para OCSP da CA privada da AWS.

Por exemplo, aqui está um registro CNAME para OCSP personalizado exibido no Amazon Route 53.

Nome de registro	Tipo	Política de roteamento	Diferenciador	Valor/Encaminhar tráfego para
alternative.example.com	CNAME	Simples	-	proxy.example.com

> **Note**
>
> O valor do CNAME não deve incluir um prefixo de protocolo como “http://” ou “https://”.

Arquivo: 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
```

Se obtiver êxito, esse comando produz o nome de recurso da Amazon (ARN) da CA.

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

Use o comando a seguir para inspecionar a configuração da 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
```

Essa descrição deve conter a seção a seguir.

```
"RevocationConfiguration": {
   ...
   "OcspConfiguration": {
      "Enabled": true,
      "OcspCustomCname": "alternative.example.com"
   }
   ...
}
```

Exemplo 3: criar uma CA com uma CRL anexada

Neste exemplo, a configuração de revogação define parâmetros da CRL.

Arquivo: 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
```

Se obtiver êxito, esse comando produz o nome de recurso da Amazon (ARN) da CA.

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

Use o comando a seguir para inspecionar a configuração da 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
```

Essa descrição deve conter a seção a seguir.

```
"RevocationConfiguration": {
```

```
   ...
   "CrlConfiguration": {
      "Enabled": true,
      "ExpirationInDays": 7,
      "S3BucketName": "DOC-EXAMPLE-BUCKET"
   },
   ...
}
```

Exemplo 4: criar uma CA com uma CRL anexada e um CNAME personalizado habilitado

Neste exemplo, a configuração de revogação define parâmetros de CRL que incluem um CNAME personalizado.

Arquivo: 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
```

Se obtiver êxito, esse comando produz o nome de recurso da Amazon (ARN) da CA.

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

Use o comando a seguir para inspecionar a configuração da 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
```

Essa descrição deve conter a seção a seguir.

```
"RevocationConfiguration": {
   ...
   "CrlConfiguration": {
      "Enabled": true,
      "ExpirationInDays": 7,
      "CustomCname": "alternative.example.com",
      "S3BucketName": "DOC-EXAMPLE-BUCKET",
   ...
   }
}
```

Exemplo 5: criar uma CA e especificar o modo de uso

Neste exemplo, o modo de uso da CA é especificado ao criar uma CA. Se não for especificado, o parâmetro de modo de uso assumirá GENERAL_PURPOSE como padrão. Neste exemplo, o parâmetro está definido como SHORT_LIVED_CERTIFICATE, o que significa que a CA emitirá certificados com um período máximo de validade de sete dias. Em situações em que é inconveniente configurar a revogação, um certificado de curta duração comprometido expira rapidamente como parte das operações normais. Consequentemente, esse exemplo de CA não tem um mecanismo de revogação.

> **Note**
>
> CA privada da AWS não executa verificações de validade em certificados de CA raiz.

```
$ 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
```

Use o describe-certificate-authoritycomando no AWS CLI para exibir detalhes sobre a CA resultante, conforme mostrado no comando a seguir:

```
$ 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
          }
       },
 ...
```

Exemplo 6: criar uma CA para login do Active Directory

É possível criar uma CA privada adequada para uso no repositório Enterprise NTAuth do Microsoft Active Directory (AD), onde ela pode emitir certificados de logon com cartão ou de controlador de

domínio. Para obter informações sobre a importação de um certificado CA para o AD, consulte Como importar certificados de autoridade de certificação (CA) de terceiros para o repositório Enterprise NTAuth.

A ferramenta certutil da Microsoft pode ser usada para publicar certificados CA no AD invocando a opção -dspublish. Um certificado publicado no AD com certutil é confiável em toda a floresta. Com o uso de uma política de grupo, também é possível limitar a confiança a um subconjunto de toda a floresta, por exemplo, um único domínio ou um grupo de computadores em um domínio. Para que o logon funcione, a CA emissora também deve ser publicada no repositório NTAuth. Para obter mais informações, consulte Distribuir certificados para computadores cliente usando a política de grupo.

Esse exemplo usa o arquivo de configuração `ca_config_AD.txt` a seguir.

Arquivo: 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
```

Se obtiver êxito, esse comando produz o nome de recurso da Amazon (ARN) da CA.

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

Use o comando a seguir para inspecionar a configuração da 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
```

Essa descrição deve conter a seção a seguir.

```
...

"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"
      }
   ]
}
...
```

Exemplo 7: Crie um Matter CA com uma CRL anexada e a extensão CDP omitida dos certificados emitidos

Você pode criar uma CA privada adequada para emitir certificados para o padrão doméstico inteligente Matter. Neste exemplo, a configuração da CA `ca_config_PAA.txt` define uma Autoridade de Atestado de Produto Matter (PAA) com a ID do fornecedor (VID) definida como FFF1.

Arquivo: 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"
      }
    ]
  }
}
```

A configuração de revogação ativa as CRLs e configura a CA para omitir a URL padrão do CDP de qualquer certificado emitido.

Arquivo: 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
```

Se obtiver êxito, esse comando produz o nome de recurso da Amazon (ARN) da CA.

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

Use o comando a seguir para inspecionar a configuração da 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
```

Essa descrição deve conter a seção a seguir.

```
"RevocationConfiguration": {
   ...
   "CrlConfiguration": {
      "Enabled": true,
      "ExpirationInDays": 7,
      "S3BucketName": "DOC-EXAMPLE-BUCKET",
   "CrlDistributionPointExtensionConfiguration":{
  "OmitExtension":true
   }
   },
   ...
}
...
```

Usando AWS CloudFormation para criar uma CA

Para obter informações sobre como criar uma CA privada usando AWS CloudFormation, consulte Referência do tipo de CA privada da AWS recurso no Guia AWS CloudFormation do usuário.

Criar e instalar o certificado de CA

Conclua os procedimentos a seguir para criar e instalar o certificado CA privado. Depois, a CA estará pronta para uso.

CA privada da AWS suporta três cenários para instalar um certificado CA:

- Instalando um certificado para uma CA raiz hospedada pelo CA privada da AWS
- Instalar um certificado CA subordinado cuja autoridade pai é hospedada pelo CA privada da AWS
- Instalar um certificado CA cuja autoridade pai é hospedada externamente

As seções a seguir descrevem os procedimentos para cada cenário. Os procedimentos do console começam na página do console CAs privadas.

Algoritmos de assinatura compatíveis

O suporte a algoritmos de assinatura para certificados de CA depende do algoritmo de assinatura da CA principal e da Região da AWS. As restrições a seguir se aplicam tanto ao console quanto AWS CLI às operações.

- Uma CA principal com o algoritmo de assinatura RSA pode emitir certificados com os seguintes algoritmos:
  - SHA256 RSA
  - SHA384 RSA
  - SHA512 RSA
- Em um legado Região da AWS, uma CA principal com o algoritmo de assinatura EDCSA pode emitir certificados com os seguintes algoritmos:
  - SHA256 ECDSA
  - SHA384 ECDSA
  - SHA512 ECDSA

  O legado Regiões da AWS inclui:

Nome da região	Localização geográfica
eu-north-1	Europa (Estocolmo)
me-south-1	Oriente Médio (Barém)
ap-south-1	Ásia-Pacífico (Mumbai)
eu-west-3	Europa (Paris)
us-east-2	Leste dos EUA (Ohio)
af-south-1	África (Cidade do Cabo)
eu-west-1	Europa (Irlanda)
eu-central-1	Europa (Frankfurt)
sa-east-1	América do Sul (São Paulo)
ap-east-1	Ásia-Pacífico (Hong Kong)
us-east-1	Leste dos EUA (Norte da Virgínia)
ap-northeast-2	Ásia-Pacífico (Seul)
eu-west-2	Europa (Londres)
ap-northeast-1	Ásia-Pacífico (Tóquio)

Nome da região	Localização geográfica
us-gov-east-1	AWS GovCloud (Leste dos EUA)
us-gov-west-1	AWS GovCloud (Oeste dos EUA)
us-west-2	Oeste dos EUA (Oregon)
us-west-1	Oeste dos EUA (N. da Califórnia)
ap-southeast-1	Ásia-Pacífico (Singapura)
ap-southeast-2	Ásia-Pacífico (Sydney)

- Em um sistema não legado Região da AWS, as seguintes regras se aplicam ao EDCSA:
  - Uma CA principal com o algoritmo de assinatura EC_prime256v1 pode emitir certificados com ECDSA P256.
  - Uma CA principal com o algoritmo de assinatura EC_secp384r1 pode emitir certificados com ECDSA P384.

Instalar um certificado de CA raiz

Você pode instalar um certificado CA raiz do AWS Management Console ou do AWS CLI.

Para criar e instalar um certificado para a CA raiz privada (console)

1. (Opcional) Se você ainda não estiver na página de detalhes da CA, abra o console do CA privada da AWS em https://console.aws.amazon.com/acm-pca/home. Na página Autoridades de certificação privadas, escolha uma CA raiz com o status Certificado pendente ou Ativo.
2. Escolha Ações, Instalar certificado de CA para abrir a página Instalar certificado de CA raiz.
3. Em Especifique os parâmetros de certificado CA raiz, especifique os seguintes parâmetros de certificado:

- Validade: especifica a data e a hora de expiração do certificado de CA. O período de validade CA privada da AWS padrão para um certificado CA raiz é de 10 anos.
- Algoritmo de assinatura: especifica o algoritmo de assinatura a ser usado quando a CA raiz emite novos certificados. As opções disponíveis variam de acordo com o Região da AWS local em que você está criando a CA. Para obter mais informaçõesAlgoritmos de assinatura compatíveis, consulteAlgoritmos criptográficos com suporte, e SigningAlgorithmem CertificateAuthorityConfiguração.
  - SHA256 RSA
  - SHA384 RSA
  - SHA512 RSA

Verifique se as configurações estão corretas e escolha Confirmar e instalar. CA privada da AWS exporta uma CSR para sua CA, gera um certificado usando um modelo de certificado de CA raiz e assina automaticamente o certificado. CA privada da AWS em seguida, importa o certificado CA raiz autoassinado.

4. A página de detalhes da CA exibe o status da instalação (sucesso ou falha) na parte superior. Se a instalação tiver sido bem-sucedida, a CA raiz recém-concluída exibirá um status Ativo no painel Geral.

Para criar e instalar um certificado para a CA raiz privada (AWS CLI)

1. Gere uma solicitação de assinatura 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
```

O arquivo resultante `ca.csr`, um arquivo PEM codificado no formato base64, tem a seguinte aparência.

```
-----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-----
```

É possível usar o OpenSSL para visualizar e verificar o conteúdo da CSR.

```
openssl req -text -noout -verify -in ca.csr
```

Isso gera uma saída semelhante à seguinte.

```
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. Usando o CSR da etapa anterior como argumento para o parâmetro `--csr`, emita o certificado raiz.

> **Note**
>
> Se você estiver usando a AWS CLI versão 1.6.3 ou posterior, use o prefixo `fileb://` ao especificar o arquivo de entrada necessário. Isso garante que os dados CA privada da AWS codificados em Base64 sejam analisados corretamente.

```
$ 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 o certificado raiz.

```
$ 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
```

O arquivo resultante `cert.pem`, um arquivo PEM codificado no formato base64, tem a seguinte aparência.

```
-----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-----
```

É possível usar o OpenSSL para visualizar e verificar o conteúdo do certificado.

```
openssl x509 -in cert.pem -text -noout
```

Isso gera uma saída semelhante à seguinte.

```
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 o certificado CA raiz para instalá-lo na CA.

> **Note**
>
> Se você estiver usando a AWS CLI versão 1.6.3 ou posterior, use o prefixo `fileb://` ao especificar o arquivo de entrada necessário. Isso garante que os dados CA privada da AWS codificados em Base64 sejam analisados corretamente.

```
$ aws acm-pca import-certificate-authority-certificate \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --certificate file://cert.pem
```

Inspecione o novo status da 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
```

O status agora aparece como ATIVO.

```
{
    "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
            }
        }
    }
}
```

Instalando um certificado CA subordinado hospedado pelo CA privada da AWS

Você pode usar o AWS Management Console para criar e instalar um certificado para sua CA subordinada CA privada da AWS hospedada.

Para criar e instalar um certificado para sua CA subordinada CA privada da AWS hospedada

1. (Opcional) Se você ainda não estiver na página de detalhes da CA, abra o console do CA privada da AWS em https://console.aws.amazon.com/acm-pca/home. Na página Autoridades de certificação privadas, escolha uma CA subordinada com o status Certificado pendente ou Ativo.
2. Escolha Ações, Instalar certificado de CA para abrir a página Instalar certificado de CA subordinada.
3. Na página Instalar certificado de CA subordinado, em Selecionar tipo de CA, escolha AWS Private CAinstalar um certificado gerenciado por CA privada da AWS.
4. Em Selecionar CA principal, escolha uma CA na lista CA privada principal. A lista é filtrada para exibir CAs que atendem aos seguintes critérios:

   - É preciso ter permissão para usar a CA.
   - A CA não é auto-assinável.
   - O CA está no estado `ACTIVE`.
   - O modo da CA é `GENERAL_PURPOSE`.
5. Em Especifique os parâmetros de certificado de CA subordinada, especifique os seguintes parâmetros de certificado:

   - Validade: especifica a data e a hora de expiração do certificado de CA.
   - Algoritmo de assinatura: especifica o algoritmo de assinatura a ser usado quando a CA raiz emite novos certificados. As opções são:
     - SHA256 RSA
     - SHA384 RSA
     - SHA512 RSA

   - Comprimento do caminho: o número de camadas de confiança que a CA subordinada pode adicionar ao assinar novos certificados. Um comprimento de caminho igual a zero (o padrão) significa que somente certificados de entidade final, e não certificados de CA podem ser

criados. Um comprimento de caminho de um ou mais significa que a CA subordinada pode emitir certificados para criar CAs adicionais subordinadas a ela.

- ARN do modelo: exibe o ARN do modelo de configuração para esse certificado CA. O modelo será alterado se você alterar o comprimento do caminho especificado. Se você criar um certificado usando o comando issue-certificate da CLI ou a IssueCertificateação da API, deverá especificar o ARN manualmente. Para obter informações sobre os modelos de certificado CA disponíveis, consulte Noções básicas sobre modelos de certificados.

6. Verifique se as configurações estão corretas e escolha Confirmar e instalar. CA privada da AWS exporta uma CSR, gera um certificado usando um modelo de certificado de CA subordinado e assina o certificado com a CA principal selecionada. CA privada da AWS em seguida, importa o certificado CA subordinado assinado.
7. A página de detalhes da CA exibe o status da instalação (sucesso ou falha) na parte superior. Se a instalação tiver sido bem-sucedida, a CA subordinada recém-concluída exibirá um status Ativo no painel Geral.

Instalar um certificado de CA subordinada assinado por uma CA pai externa

Depois de criar uma CA privada subordinada, conforme descrito em Procedimento para criar uma CA (console) ou Procedimento para criar uma CA (CLI) , você tem a opção de ativá-la instalando um certificado de CA assinado por uma autoridade de assinatura externa. Assinar seu certificado de CA subordinado com uma CA externa exige que você primeiro configure um provedor externo de serviços de confiança como sua autoridade de assinatura ou providencie o uso de um provedor terceirizado.

> **Note**
>
> Os procedimentos para criar ou obter um provedor de serviços de confiança externa CA externa estão fora do escopo deste guia.

Depois de criar uma CA subordinada e ter acesso a uma autoridade de assinatura externa, conclua as seguintes tarefas:

1. Obtenha uma solicitação de assinatura de certificado (CSR) de CA privada da AWS.
2. Envie a CSR à sua autoridade de assinatura externa e obtenha um certificado de CA assinada junto com todos os certificados da cadeia.

3. Importe o certificado CA e a cadeia CA privada da AWS para ativar sua CA subordinada.

Para ver os procedimentos detalhados, consulte Certificados CA privados externamente assinados .

Controlar o acesso a uma CA privada

Qualquer usuário com as permissões necessárias em uma CA privada CA privada da AWS pode usar essa CA para assinar outros certificados. O proprietário da CA pode emitir certificados ou delegar as permissões necessárias para a emissão de certificados a um usuário AWS Identity and Access Management (IAM) que resida na mesma. Conta da AWS Um usuário que reside em uma AWS conta diferente também pode emitir certificados se autorizado pelo proprietário da CA por meio de uma política baseada em recursos.

Usuários autorizados, sejam eles de conta única ou de contas cruzadas, podem usar nossos CA privada da AWS AWS Certificate Manager recursos ao emitir certificados. Os certificados emitidos pela CA privada da AWS IssueCertificateAPI ou pelo comando da CLI issue-certificate não são gerenciados. Ao expirarem, esses certificados exigem instalação manual nos dispositivos de destino e renovação manual. Os certificados emitidos pelo console do ACM, pela RequestCertificateAPI do ACM ou pelo comando da CLI request-certificate são gerenciados. Esses certificados podem ser facilmente instalados em serviços integrados com o ACM. Se o administrador da CA permitir e a conta do emissor tiver um perfil vinculado a serviço para o ACM, os certificados gerenciados serão renovados automaticamente quando expirarem.

Tópicos

- Criar permissões de conta única para um usuário do IAM
- Anexe uma política para acesso entre contas

Criar permissões de conta única para um usuário do IAM

Quando o administrador da CA (ou seja, o proprietário da CA) e o emissor do certificado residem em uma única AWS conta, a melhor prática é separar as funções de emissor e administrador criando um usuário AWS Identity and Access Management (IAM) com permissões limitadas. Para obter informações sobre como usar o IAM com CA privada da AWS, junto com exemplos de permissões, consulteIdentity and Access Management (IAM) para AWS Private Certificate Authority.

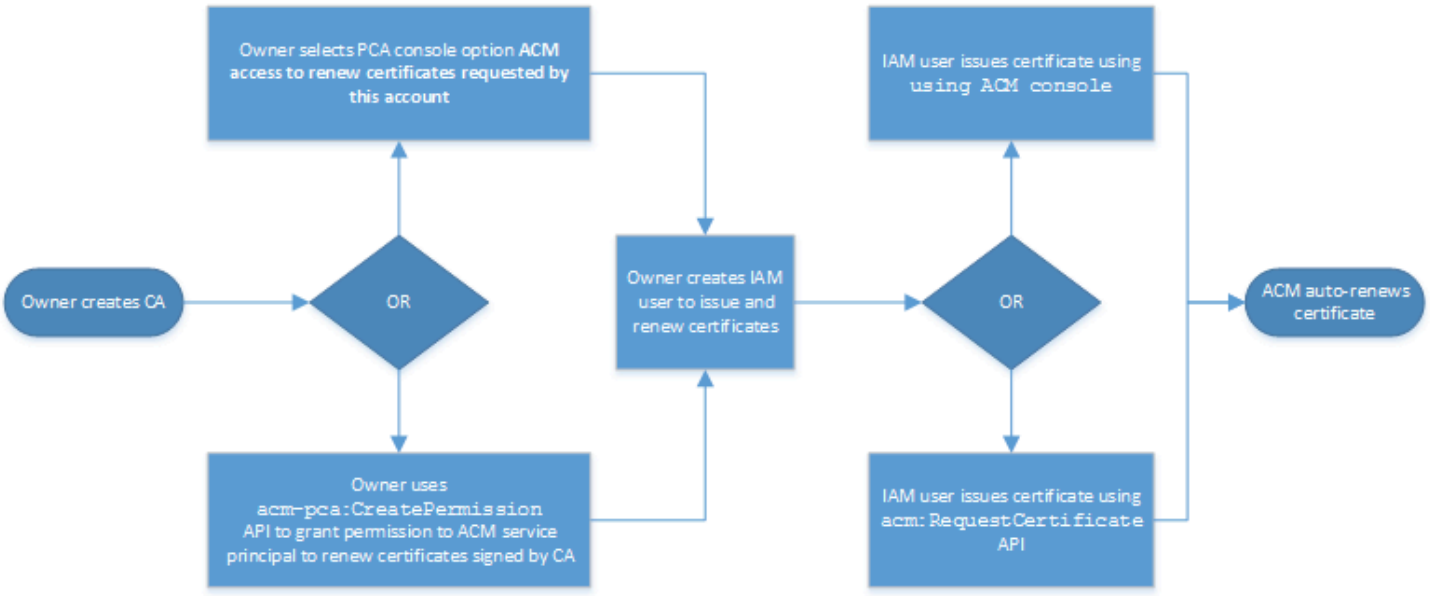
Caso 1 de conta única: emissão de certificado não gerenciado

Nesse caso, o proprietário da conta cria uma CA privada e, em seguida, cria um usuário do IAM com permissão para emitir certificados assinados pela CA privada. O usuário do IAM emite um certificado chamando a CA privada da AWS `IssueCertificate` API.

Os certificados emitidos dessa maneira não são gerenciados, o que significa que um administrador deve exportá-los e instalá-los nos dispositivos em que devem ser usados. Eles também devem ser renovados manualmente ao expirarem. A emissão de um certificado usando essa API requer uma solicitação de assinatura de certificado (CSR) e um par de chaves gerados externamente pelo CA privada da AWS OpenSSL ou por um programa similar. Para obter mais informações, consulte a documentação do `IssueCertificate`.

Caso 2 de conta única: emissão de certificado gerenciado por meio do ACM

Esse segundo caso envolve operações de API do ACM e do PCA. O proprietário da conta cria uma CA privada e um usuário do IAM como antes. Em seguida, o proprietário da conta concede permissão à entidade principal de serviço do ACM para renovar automaticamente todos os certificados assinados por essa CA. O usuário do IAM emite novamente o certificado, mas dessa vez chamando a API `RequestCertificate` do ACM, que lida com a geração de CSRs e de chaves. Quando o certificado expirar, o ACM automatizará o fluxo de trabalho de renovação.



O proprietário da conta tem a opção de conceder permissão de renovação por meio do console de gerenciamento durante ou após a criação da CA ou usando a API `CreatePermission` do PCA. Os certificados gerenciados criados a partir desse fluxo de trabalho estão disponíveis para uso com AWS serviços integrados ao ACM.

A seção a seguir inclui procedimentos para conceder permissões de renovação.

Atribuir permissões de renovação de certificado ao ACM

Com a renovação gerenciada no AWS Certificate Manager (ACM), você pode automatizar o processo de renovação de certificados para certificados públicos e privados. Para que o ACM renova automaticamente os certificados gerados por uma CA privada, a entidade principal de serviço do ACM deve receber toda a permissão possível pela própria CA. Se essas permissões de renovação não estiverem presentes para o ACM, o proprietário da CA (ou um representante autorizado) deverá reemitir manualmente cada certificado privado quando ele expirar.

> **Important**
>
> Esses procedimentos para atribuir permissões de renovação se aplicam somente quando o proprietário da CA e o emissor do certificado residem na mesma AWS conta. Para cenários entre contas, consulte Anexe uma política para acesso entre contas.

As permissões de renovação podem ser delegadas durante a criação da CA privada ou alteradas a qualquer momento depois, desde que a CA esteja no estado `ACTIVE`.

Você pode gerenciar permissões de CA privada no Console do CA privada da AWS, na AWS Command Line Interface (AWS CLI) ou na API do CA privada da AWS:

Para atribuir permissões de CA ao ACM (console)

1. Faça login na sua AWS conta e abra o CA privada da AWS console em https://console.aws.amazon.com/acm-pca/home.
2. Na página Autoridades de certificação privadas, escolha sua CA privada na lista.
3. Escolha Ações, Configurar permissões da CA.
4. Selecione Autorizar acesso ao ACM para renovar certificados solicitados por essa conta.
5. Selecione Save (Salvar).

Para gerenciar as permissões do ACM em CA privada da AWS ()AWS CLI

Use o comando create-permission para atribuir permissões ao ACM. Atribua todas as permissões necessárias (`IssueCertificate`, `GetCertificate` e `ListPermissions`) para que o ACM renove automaticamente seus 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
```

Use o comando list-permissions para listar as permissões delegadas por uma CA.

```
$ aws acm-pca list-permissions \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID
```

Use o comando delete-permission para revogar as permissões atribuídas por uma CA a um diretor de serviço. AWS

```
$ aws acm-pca delete-permission \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --principal acm.amazonaws.com
```

Anexe uma política para acesso entre contas

Quando o administrador da CA e o emissor do certificado residem em contas da AWS diferentes, o administrador deve compartilhar o acesso à CA. Isso é feito anexando uma política baseada em recurso a ela. A política concede permissões de emissão a um diretor específico, que pode ser o proprietário AWS da conta, um usuário do IAM, um AWS Organizations ID ou um ID de unidade organizacional.

Um administrador de CA pode anexar e gerenciar das seguintes maneiras:

- No console de gerenciamento, usando AWS Resource Access Manager (RAM), que é um método padrão para compartilhar AWS recursos entre contas. Quando você compartilha um recurso de CA AWS RAM com um diretor em outra conta, a política baseada em recursos necessária é anexada

à CA automaticamente. Para obter mais informações sobre o RAM, consulte o Guia do usuário do AWS RAM.

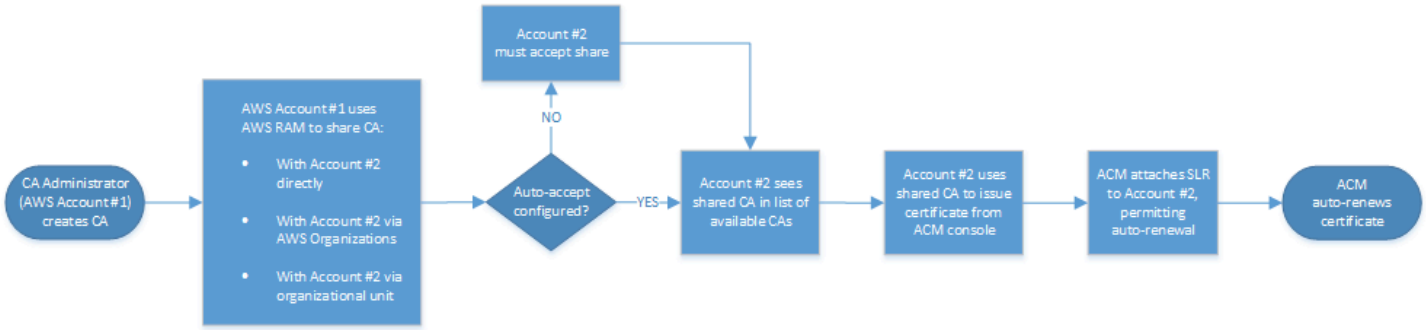
> **Note**
>
> É possível abrir facilmente o console do RAM escolhendo uma CA e depois selecionando Ações, Gerenciar compartilhamentos de recursos.

- Programaticamente, usando as APIs do PCA, e. PutPolicyGetPolicyDeletePolicy
- Manualmente, usando os comandos put-policy, get-policy e delete-policy do PCA no AWS CLI.

Somente o método do console requer acesso ao RAM.

Caso 1 entre contas: emissão de certificado gerenciado a partir do console

Nesse caso, o administrador da CA usa AWS Resource Access Manager (AWS RAM) para compartilhar o acesso da CA com outra AWS conta, o que permite que essa conta emita certificados ACM gerenciados. O diagrama mostra que é AWS RAM possível compartilhar a CA diretamente com a conta ou indiretamente por meio de uma AWS Organizations ID da qual a conta é membro.



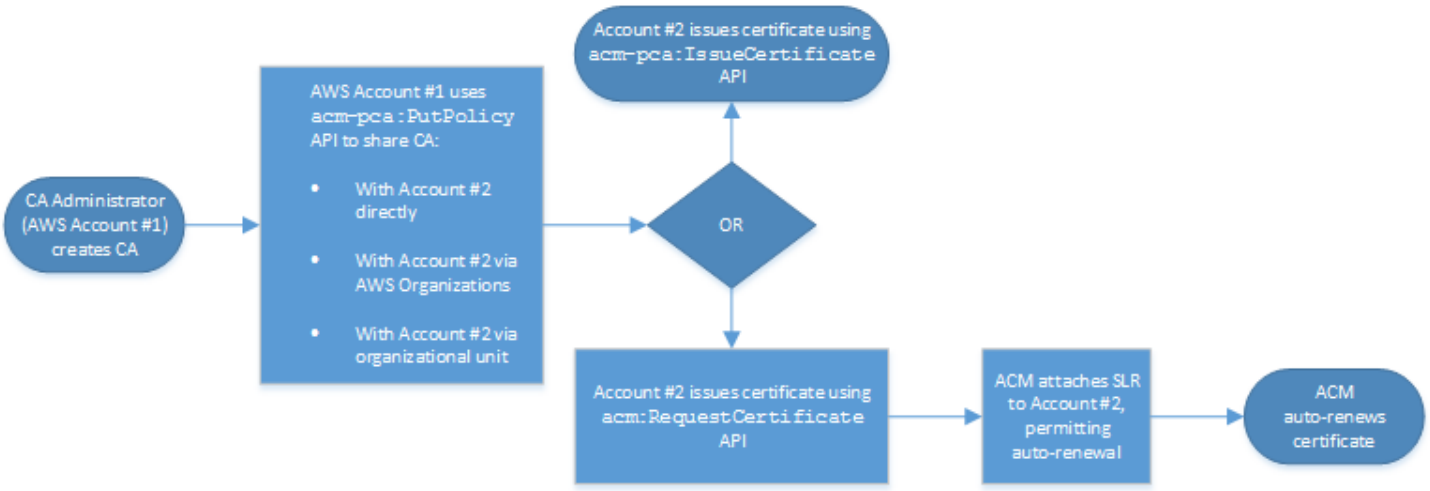
Depois que a RAM compartilha um recurso AWS Organizations, o principal destinatário deve aceitar o recurso para que ele entre em vigor. O destinatário pode configurar AWS Organizations para aceitar automaticamente os compartilhamentos oferecidos.

> **Note**
>
> A conta do destinatário é responsável por configurar a renovação automática no ACM. Normalmente, na primeira vez em que uma CA compartilhada é usada, o ACM instala um perfil vinculado a serviço que permite que ele faça chamadas de certificado autônomas no CA privada da AWS. Se isso falhar (geralmente devido à falta de uma permissão), os certificados da CA não serão renovados automaticamente. Somente o usuário do ACM pode

resolver o problema, e não o administrador da CA. Para obter mais informações, consulte Usar um perfil vinculado a serviço (SLR) com o ACM.

Caso 2 entre contas: emissão de certificados gerenciados e não gerenciados usando a API ou a CLI

Esse segundo caso demonstra as opções de compartilhamento e emissão que são possíveis usando a API AWS Certificate Manager e. CA privada da AWS Todas essas operações também podem ser realizadas usando os AWS CLI comandos correspondentes.



Como as operações de API estão sendo usadas diretamente neste exemplo, o emissor do certificado tem a opção de duas operações de API para emitir um certificado. A ação da API `IssueCertificate` do PCA resulta em um certificado não gerenciado que não será renovado automaticamente e deve ser exportado e instalado manualmente. A ação da API do ACM RequestCertificateresulta em um certificado gerenciado que pode ser facilmente instalado nos serviços integrados do ACM e renovado automaticamente.

> **Note**
>
> A conta do destinatário é responsável por configurar a renovação automática no ACM. Normalmente, na primeira vez em que uma CA compartilhada é usada, o ACM instala um perfil vinculado a serviço que permite que ele faça chamadas de certificado autônomas no CA privada da AWS. Se isso falhar (geralmente por falta de permissão), os certificados da CA não serão renovados automaticamente, e somente o usuário do ACM poderá resolver o problema, não o administrador da CA. Para obter mais informações, consulte Usar um perfil vinculado a serviço (SLR) com o ACM.

Listar CAs privadas

Você pode usar o CA privada da AWS console ou AWS CLI listar CAs privadas que você possui ou às quais tem acesso.

Para listar as CAs disponíveis usando o console

1. Faça login na sua AWS conta e abra o CA privada da AWS console em https://console.aws.amazon.com/acm-pca/home.
2. Revise as informações na lista Autoridades de certificado privadas. É possível navegar por várias páginas de CAs usando os números de página no canto superior direito. Cada CA ocupa uma linha com algumas ou todas as seguintes colunas exibidas para cada:

- Requerente: resumo das informações de nomes distintos da CA.
- Id: identificador exclusivo hexadecimal de 32 bytes da CA.
- Status: o status da CA. Os valores possíveis são Criando, Certificado pendente, Ativo, Excluído, Desabilitado, Expirado e Com falha.
- Tipo: o tipo de CA. Os valores possíveis são Raiz e Subordinada.
- Modo: o modo da CA. Os valores possíveis são Uso geral (emite certificados que podem ser configurados com qualquer data de expiração) e Certificado de curta duração (emite certificados com um período máximo de validade de sete dias). Em alguns casos, um curto período de validade pode substituir, um mecanismo de revogação. O padrão é Uso geral.
- Proprietário — A AWS conta que possui a CA. Pode ser sua conta ou uma conta que delegou permissões de gerenciamento da CA à você.
- Algoritmo de chave: o algoritmo de chave pública compatível pela CA. Os valores possíveis são RSA_2048, RSA_4096, EC_prime256v1 e EC_secp384r1.
- Algoritmo de assinatura: o algoritmo que a CA usa para assinar solicitações de certificado. (Não deve ser confundido com o parâmetro `SigningAlgorithm` usado para assinar certificados quando são emitidos.) Os valores possíveis são SHA256WITHECDSA, SHA384WITHECDSA, SHA512WITHECDSA, SHA256WITHRSA, SHA384WITHRSA e SHA512WITHRSA.

> **Note**
>
> Você pode personalizar as colunas que deseja exibir, bem como outras configurações, escolhendo o ícone de configurações no canto superior direito do console.

Para listar as CAs disponíveis usando o AWS CLI

Use o comando list-certificate-authorities para listar as CAs disponíveis, como mostrado no exemplo a seguir:

```
$ aws acm-pca list-certificate-authorities --max-items 10
```

O comando retorna informações semelhantes às seguintes:

```
{
   "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
            }
         }
      }
 ...
   ]
```

```
}
```

Visualizar uma CA privada

Você pode usar o console do ACM ou o AWS CLI para visualizar metadados detalhados sobre uma CA privada e alterar vários valores conforme necessário. Para obter informações detalhes sobre como atualizar CAs, consulte Atualizar a CA privada.

Para visualizar detalhes da CA no console

1. Faça login na sua AWS conta e abra o CA privada da AWS console em https://console.aws.amazon.com/acm-pca/home.
2. Analise a lista Autoridades de certificado privadas. É possível navegar por várias páginas de CAs usando os números de página no canto superior direito.
3. Para mostrar metadados detalhados para uma CA listada, escolha o botão de opção referente à CA que você deseja inspecionar. Isso abre um painel de detalhes com as seguintes visualizações em guias:

   - Guia Requerente: informações sobre o nome distinto da CA. Para ter mais informações, consulte Opções de nomes distintos de requerentes. Os campos exibidos incluem:
     - Requerente: resumo dos campos de informações de nome fornecidos
     - Organização (O): por exemplo, o nome de uma empresa
     - Unidade organizacional (UO): por exemplo, uma divisão dentro de uma empresa
     - Nome do país (C): código do país com duas letras
     - Nome do estado ou província: nome completo de um estado ou província
     - Nome da localidade: o nome de uma cidade
     - Nome comum (CN) — Uma string legível por humanos para identificar a CA.
   - Guia Certificado de CA: Informações sobre a validade do certificado da CA
     - Válido até: até que data e hora o certificado CA será válido
     - Expira em: o número de dias até a expiração
   - Guia Configuração de revogação: suas seleções atuais para opções de revogação de certificados. Escolha Editar para atualizar.
     - Distribuição da Lista de revogação de certificados (CRL): status Habilitado ou Desabilitado
     - Online Certificate Status Protocol (OCSP): status Habilitado ou Desabilitado

- Guia Permissões: sua seleção atual de permissões de renovação de certificados para essa CA por meio do AWS Certificate Manager (ACM). Escolha Editar para atualizar.
- Autorização do ACM para renovações: status Autorizado ou Não autorizado
- Guia Etiquetas: sua atribuição atual de etiquetas personalizáveis para essa CA. Selecione Gerenciar etiquetas.
- Guia Compartilhamentos de recursos — Sua atribuição atual de compartilhamentos de recursos para esta CA por meio de AWS Resource Access Manager (RAM). Escolha Gerenciar compartilhamentos de recursos para atualizar.
  - Nome: o nome do compartilhamento de recurso
  - Status: o status do compartilhamento de recurso

4. Escolha o campo ID da CA que você deseja inspecionar para abrir o painel Geral. O identificador exclusivo hexadecimal de 32 bytes dessa CA aparece na parte superior. O painel a seguir fornece as seguintes informações adicionais:

   - Status: o status da CA. Os valores possíveis são Criando, Certificado pendente, Ativo, Excluído, Desabilitado, Expirado e Com falha.
   - ARN: o Nome do recurso da Amazon para a CA.
   - Proprietário — A AWS conta que possui a CA. Pode ser sua conta (Própria) ou uma conta que delegou permissões de gerenciamento da CA à você.
   - Tipo de CA: o tipo da CA. Os valores possíveis são Raiz e Subordinada.
   - Criada em: a data e a hora em que a CA foi criada.
   - Data de expiração: a data e a hora de expiração do certificado da CA.
   - Modo: o modo da CA. Os valores possíveis são Uso geral (certificados que podem ser configurados com qualquer data de expiração) e Certificado de curta duração (certificados com um período máximo de validade de sete dias). Em alguns casos, um curto período de validade pode substituir, um mecanismo de revogação. O padrão é Uso geral.
   - Algoritmo de chave: o algoritmo de chave pública compatível pela CA. Os valores possíveis são RSA 2048, RSA 4096, ECDSA P2567 e ECDSA P384.
   - Algoritmo de assinatura: o algoritmo que a CA usa para assinar solicitações de certificado. (Não deve ser confundido com o parâmetro `SigningAlgorithm` usado para assinar certificados quando são emitidos.) Os valores possíveis são SHA256 ECDSA, SHA384 ECDSA, SHA512 ECDSA, SHA256 RSA, SHA384 RSA e SHA512 RSA
   - Padrão de segurança de armazenamento de chaves: nível de conformidade com os padrões federais de processamento de informações. Os valores possíveis são FIPS 140-2 nível 3 ou

superior e FIPS 140-2 nível 3 ou superior. Esse parâmetro varia de acordo com a região da AWS .

Para visualizar e modificar os detalhes da CA usando o AWS CLI

Use o comando describe-certificate-authority na AWS CLI para exibir detalhes sobre a CA, como mostra o comando a seguir:

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

O comando retorna informações semelhantes às seguintes:

```
{
   "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 obter informações sobre como atualizar uma CA privada pela linha de comando, consulte Atualizar uma CA (CLI).

Gerenciamento de etiquetas para sua CA privada

As tags são palavras ou frases que atuam como metadados para identificar e organizar os recursos da AWS . Cada tag consiste em uma chave e um valor. Você pode usar o CA privada da AWS console, AWS Command Line Interface (AWS CLI) ou a API PCA para adicionar, visualizar ou remover tags para CAs privadas.

É possível adicionar ou remover etiquetas personalizadas da sua CA privada a qualquer momento. Por exemplo, você pode atribuir uma etiqueta a CAs privadas com pares de chave-valor como `Environment=Prod` ou `Environment=Beta` para identificar para qual ambiente se destina a CA. Para obter mais informações, consulte Criar uma CA privada.

> **Note**
>
> Para anexar etiquetas a uma CA privada durante o procedimento de criação, um administrador de CA deve primeiro associar uma política do IAM interna à ação `CreateCertificateAuthority` e permitir explicitamente a marcação. Para ter mais informações, consulte Tag-on-create: Anexando tags a uma CA no momento da criação.

Outros AWS recursos também oferecem suporte à marcação. É possível atribuir a mesma tag a diferentes recursos para indicar se esses recursos estão relacionados. Por exemplo, você pode atribuir uma etiqueta como `Website=example.com` à sua CA, o balanceador de carga do Elastic Load Balancing e outros recursos relacionados. Para obter mais informações sobre a marcação de AWS recursos, consulte Como marcar seus recursos do Amazon EC2 no Guia do usuário do Amazon EC2.

As seguintes restrições básicas se aplicam às CA privada da AWS tags:

- O número máximo de tags por CA privada é 50.
- O tamanho máximo de uma chave de tag é 128 caracteres.
- O tamanho máximo de um valor de tag é 256 caracteres.
- A chave e o valor da tag podem conter os seguintes caracteres: A-Z, a-z e .:+=@_%-(hífen).
- As chaves e os valores de tags diferenciam maiúsculas de minúsculas.
- Os prefixos `aws:` e `rds:` são reservados para uso da AWS . Não é possível adicionar, editar nem excluir tags cujas chaves comecem com `aws:` ou `rds:`. Tags padrão que começam com `aws:` e `rds:` não contam na sua tags-per-resource cota.

- Se você planejar usar o esquema de tags em vários serviços e recursos, lembre-se de que outros serviços podem ter outras restrições para caracteres permitidos. Consulte a documentação desse serviço.
- CA privada da AWS as tags não estão disponíveis para uso nos Resource Groups e no Tag Editor no AWS Management Console.

Você pode atribuir uma tag a uma CA privada no Console do CA privada da AWS, na AWS Command Line Interface (AWS CLI) ou na API do CA privada da AWS.

Como atribuir uma tag a uma CA privada (console)

1. Faça login na sua AWS conta e abra o CA privada da AWS console em https://console.aws.amazon.com/acm-pca/home.
2. Na página Autoridades de certificação privadas, escolha sua CA privada na lista.
3. Na área de detalhes abaixo da lista, escolha a guia Tags. Uma lista de etiquetas existentes será exibida.
4. Selecione Gerenciar tags.
5. Selecione Adicionar nova tag.
6. Digite pares de chave e valor.
7. Selecione Save (Salvar).

Como atribuir uma tag a uma CA privada (AWS CLI)

Use o comando tag-certificate-authority para adicionar tags à sua 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
```

Use o comando list-tags para listar as tags de uma CA privada.

```
$ aws acm-pca list-tags \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --max-results 10
```

Use o comando untag-certificate-authority para remover tags de uma 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
```

Atualizar a CA privada

Você pode atualizar o status de uma CA privada ou alterar sua configuração de revogação depois de criá-la. Esse tópico fornece detalhes sobre o status da CA e o ciclo de vida da CA, junto com exemplos de atualizações do console e da CLI para CAs.

Atualizar o status da CA

O status de uma CA gerenciada pelos CA privada da AWS resultados de uma ação do usuário ou, em alguns casos, de uma ação de serviço. Por exemplo, o status de uma CA muda quando ela expira. As opções de status disponíveis para administradores de CA variam dependendo do status atual da CA.

CA privada da AWS pode relatar os seguintes valores de status. A tabela mostra os recursos de CA disponíveis em cada estado.

> **Note**
>
> Para todos os valores de status, exceto `DELETED` e `FAILED`, você será cobrado pela CA.

Status	Emitir certificados	Validar certificados com OCSP	Gerar CRLs	Gerar auditorias	É possível atualizar o certificado de CA	Certificados podem ser revogadc	Você é cobrado pela CA
`CREATING`: a CA está sendo criada.	Não	Não	Não	Não	Não	Não	Sim
	Não	Não	Não	Não	Não	Não	Sim

Status	Emitir certificados	Validar certificados com OCSP	Gerar CRLs	Gerar auditorias	É possível atualizar o certificado de CA	Certificados podem ser revogados	Você é cobrado pela CA
`PENDING_CERTIFICATE` : a CA foi criada e precisa de um certificado para ficar operacional.*							
`ACTIVE`	Sim	Sim	Sim	Sim	Sim	Sim	Sim
`DISABLED`: você desativou manualmente a CA.	Não	Sim	Sim	Sim	Não	Sim	Sim
`EXPIRED`: o certificado da CA expirou.**	Não	Não	Não	Não	Sim	Não	Sim
`FAILED`	A ação `CreateCertificateAuthority` falhou. Isso pode ocorrer devido a uma interrupção na rede, AWS falha no back-end ou outros erros. Uma CA com falha não pode ser recuperada. Exclua a CA e crie uma nova.						Não
`DELETED`	Sua CA está dentro do período de restauração, que pode durar de 7 a 30 dias. Após esse período, ela é excluída permanentemente. • Se você chamar a API `RestoreCertificateAuthority` em uma CA com o status `DELETED` e um certificado expirado, a CA será definida como `EXPIRED`. • Para obter mais informações sobre como excluir uma CA, consulte Excluir a CA privada.						Não

* Para concluir a ativação, você precisa gerar uma CSR, obter um certificado de CA assinado de uma CA e importar esse certificado para o CA privada da AWS. A CSR pode ser enviada à sua nova CA (para assinatura automática) ou a uma CA raiz ou subordinada on-premises. Para ter mais informações, consulte Criar e instalar o certificado de CA.

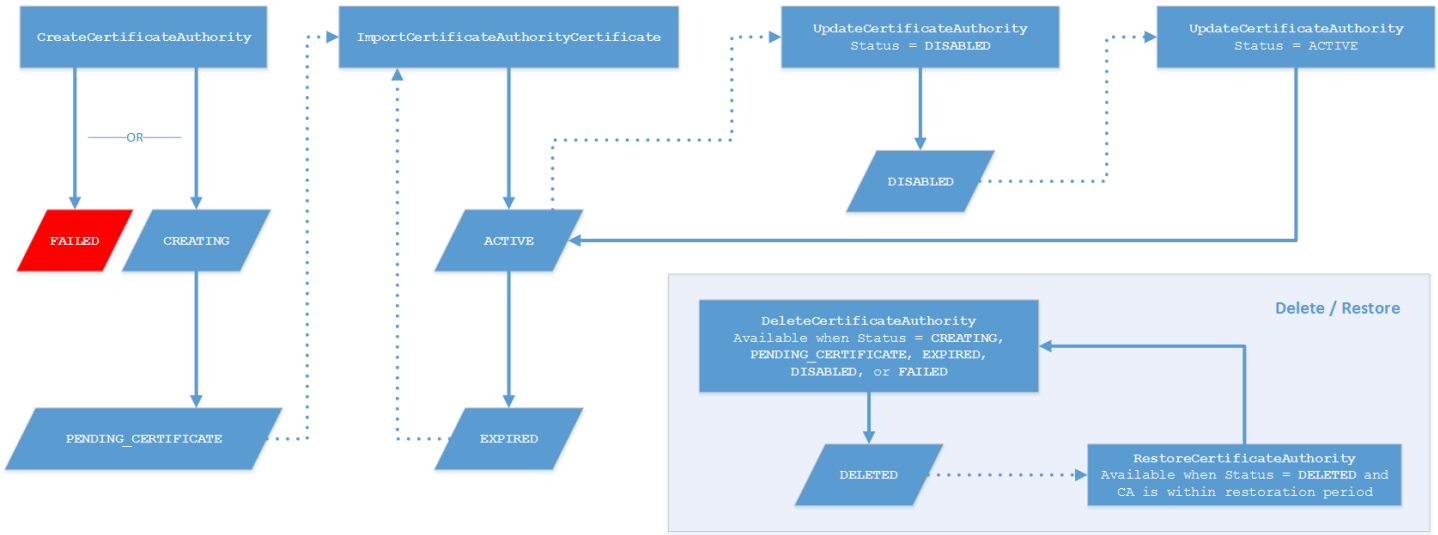
** Não é possível alterar diretamente o status de uma CA expirada. Se você importar um novo certificado para a CA, CA privada da AWS redefinirá o status para, a `ACTIVE` menos que tenha sido definido `DISABLED` antes da expiração do certificado.

Considerações adicionais sobre certificados de CA expirados:

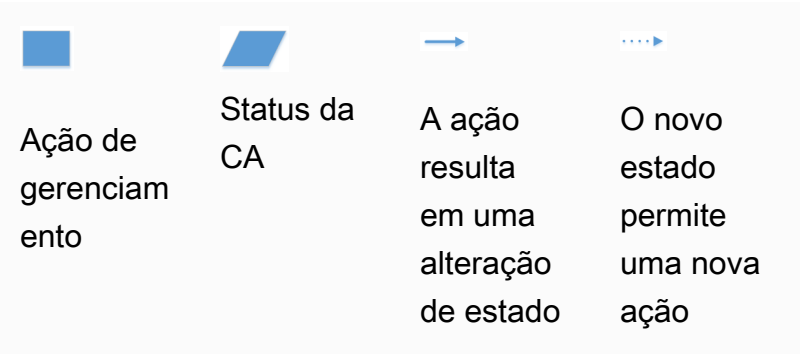
- Certificados de CA não são renovados automaticamente. Para obter informações sobre como automatizar a renovação AWS Certificate Manager, consulte. Atribuir permissões de renovação de certificado ao ACM
- Se você tentar emitir um novo certificado com uma CA expirada, a API `IssueCertificate` retornará `InvalidStateException`. Uma CA raiz expirada precisa autoassinar um novo certificado CA raiz para poder emitir novos certificados subordinados.
- As APIs `The ListCertificateAuthorities` e `DescribeCertificateAuthority` retornarão um status de `EXPIRED` se o certificado CA estiver expirado, independentemente de o status da CA estar definido como `ACTIVE` ou `DISABLED`. No entanto, se a CA expirada tiver sido definida como `DELETED`, o status retornado será `DELETED`.
- A API `UpdateCertificateAuthority` não pode atualizar o status de uma CA expirada.
- A API `RevokeCertificate` não pode ser usada para revogar qualquer certificado expirado, incluindo um certificado CA.

Status e ciclo de vida da CA

O diagrama a seguir ilustra o ciclo de vida da CA como uma interação das ações de gerenciamento com o status da CA.



Chave de diagrama



Na parte superior do diagrama, as ações de gerenciamento são aplicadas por meio do console, da CLI ou da API do CA privada da AWS . As ações usam a CA por meio da criação, ativação, expiração e renovação. O status da CA muda em resposta (conforme mostrado pelas linhas sólidas) às ações manuais ou atualizações automatizadas. Na maioria dos casos, um novo status leva a uma nova ação possível (mostrada por uma linha pontilhada) que o administrador da CA pode aplicar. A entrada inferior direita mostra os possíveis valores de status que permitem ações de exclusão e restauração.

Atualizar uma CA (console)

Os procedimentos a seguir mostram como atualizar configurações de CA existentes usando o AWS Management Console.

Atualizar o status de uma CA (console)

Neste exemplo, o status de uma CA habilitada é alterado para Desabilitada.

Para atualizar o status de uma CA

1. Faça login na sua AWS conta e abra o CA privada da AWS console em https://console.aws.amazon.com/acm-pca/home
2. Na página Autoridades de certificação privadas, escolha na lista uma CA privada que esteja ativa.
3. No menu Ações, escolha Desabilitar para desabilitar a CA privada.

Atualizar a configuração de revogação de uma CA (console)

É possível atualizar a configuração de revogação de uma CA privada, por exemplo, adicionando ou removendo o suporte para OCSP ou CRL ou modificando suas configurações.

> **Note**
>
> As alterações na configuração de revogação de uma CA não afetam certificados que já foram emitidos. Para que a revogação gerenciada funcione, os certificados mais antigos devem ser reemitidos.

Para OCSP, é possível alterar as seguintes configurações:

- Habilite ou desabilite o OCSP.
- Habilite ou desabilite um nome de domínio totalmente qualificado (FQDN) personalizado.
- Altere o FQDN.

Para uma CRL, é possível alterar uma das seguintes configurações:

- Se a CA privada gerar uma lista de revogação de certificados (CRL)
- O número de dias antes de uma CRL expirar. Observe que CA privada da AWS começa a tentar regenerar a CRL na metade do número de dias que você especificar.
- O nome do bucket do Amazon S3 no qual sua CRL é salva.
- Um alias para ocultar o nome do bucket do Amazon S3 da visualização pública.

> **Important**
>
> Alterar qualquer um dos parâmetros anteriores pode gerar efeitos negativos. Alguns exemplos incluem desabilitar a geração de CRLs, alterar o período de validade ou alterar o bucket do S3 depois de colocar a CA privada em produção. Essas alterações podem violar certificados existentes dependentes da CRL e da configuração atual dessa CRL. A alteração do alias pode ser feita com segurança, desde que o antigo alias permaneça vinculado ao bucket correto.

Para atualizar as configurações de revogação

1. Faça login na sua AWS conta e abra o CA privada da AWS console em https://console.aws.amazon.com/acm-pca/home.
2. Na página Autoridades de certificação privadas, escolha sua CA privada na lista. Isso abre o painel de detalhes referente à CA.
3. Escolha a guia Configuração de revogação e escolha Editar.
4. Em Opções de revogação de certificado, duas opções são exibidas:

   - Ativar distribuição de CRL
   - Ativar OCSP

   É possível configurar uma, nenhuma ou ambas as opções de revogação para sua CA. Embora opcional, a revogação gerenciada é recomendada como melhor prática. Antes de concluir essa etapa, consulte Configurar um método de revogação de certificado para obter informações sobre as vantagens de cada método, a configuração preliminar que talvez seja necessária e recursos adicionais de revogação.

Para configurar uma CRL

1. Selecione Ativar distribuição de CRL.
2. Para criar um bucket do Amazon S3 para as entradas da sua CRL, selecione Criar um novo bucket do S3. Dê um nome exclusivo para o bucket. (Você não precisa incluir o caminho para o bucket.) Caso contrário, deixe essa opção desmarcada e escolha um bucket existente na lista Nomes de buckets do S3.

Se você criar um novo bucket, CA privada da AWS cria e anexa a política de acesso necessária a ele. Se decidir usar um bucket existente, deverá anexar uma política de acesso a ele antes de começar a gerar CRLs. Use um dos padrões de política descritos em Políticas de acesso para CRLs no Amazon S3 . Para obter informações sobre como anexar uma política, consulte Adicionar uma política de bucket usando o console do Amazon S3.

> **Note**
>
> Quando você estiver usando o CA privada da AWS console, uma tentativa de criar uma CA falhará se as duas condições a seguir se aplicarem:
>
> - Você está aplicando configurações de Bloqueeio do acesso público ao seu bucket ou conta do Amazon S3.
> - Você solicitou CA privada da AWS a criação automática de um bucket do Amazon S3.
>
> Nessa situação, o console tenta por padrão criar um bucket acessível ao público, e o Amazon S3 rejeita essa ação. Verifique as configurações do Amazon S3 se isso ocorrer. Para obter mais informações, consulte Bloquear o acesso público ao seu armazenamento do Amazon S3.

3. Expanda Avançado para obter opções de configuração adicionais.

   - Adicione um Nome de CRL personalizado para criar um alias para o bucket do Amazon S3. Esse nome está contido em certificados emitidos pela CA na extensão “Pontos de distribuição de CRL” definida pela RFC 5280.
   - Digite o número de dias em que a CRL permanecerá válida. O valor padrão é de 7 dias. Para CRLs on-line, um período de validade de 2 a 7 dias é comum. CA privada da AWS tenta regenerar a CRL no ponto médio do período especificado.

4. Quando terminar, escolha Salvar alterações.

Para configurar o OCSP

1. Na página Revogação do certificado, escolha Ativar OCSP.
2. (Opcional) No campo Endpoint do OCSP personalizado, forneça um nome de domínio totalmente qualificado (FQDN) para o endpoint do OCSP.

Quando você fornece um FQDN nesse campo, CA privada da AWS insere o FQDN na extensão Authority Information Access de cada certificado emitido no lugar do URL padrão do respondente OCSP. AWS Quando um endpoint recebe um certificado contendo o FQDN personalizado, ele consulta esse endereço para obter uma resposta do OCSP. Para que esse mecanismo funcione, você precisa fazer duas ações adicionais:

- Use um servidor proxy para encaminhar o tráfego que chega ao seu FQDN personalizado para o respondente AWS OCSP.
- Adicionar um registro CNAME correspondente ao seu banco de dados DNS.

> **Tip**
>
> Para obter mais informações sobre a implementação de uma solução OCSP completa usando um CNAME personalizado, consulte Configurar um URL personalizado para OCSP da CA privada da AWS.

Por exemplo, aqui está um registro CNAME para OCSP personalizado exibido no Amazon Route 53.

Nome de registro	Tipo	Política de roteamento	Diferenciador	Valor/Encaminhar tráfego para
alternative.example.com	CNAME	Simples	-	proxy.example.com

> **Note**
>
> O valor do CNAME não deve incluir um prefixo de protocolo como “http://” ou “https://”.

3. Quando terminar, escolha Salvar alterações.

Atualizar uma CA (CLI)

Os procedimentos a seguir mostram como atualizar o status e a configuração de revogação de uma CA existente usando a AWS CLI.

> **Note**
>
> As alterações na configuração de revogação de uma CA não afetam certificados que já foram emitidos. Para que a revogação gerenciada funcione, os certificados mais antigos devem ser reemitidos.

Para atualizar o status da sua CA privada (AWS CLI)

Use o comando update-certificate-authority.

Isso é útil quando há uma CA existente com o status `DISABLED` que você deseja definir como `ACTIVE`. Para começar, confirme o status inicial da CA com o seguinte 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
```

Isso resulta em uma saída semelhante à seguinte:

```
{
    "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
            }
        }
    }
}
```

O seguinte comando define o status da CA privada como `ACTIVE`. Isso apenas é possível quando um certificado válido está instalado na 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"
```

Inspecione o novo status da 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
```

O status agora 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
            }
        }
    }
}
```

Em alguns casos, você pode ter uma CA ativa sem um mecanismo de revogação configurado. Se quiser começar a usar uma lista de revogação de certificados (CRL), use o procedimento a seguir.

Para adicionar uma CRL a uma CA existente (AWS CLI)

1. Use o comando a seguir para inspecionar o status atual da 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
```

A saída confirma que a CA tem o status `ACTIVE`, mas não está configurada para usar uma 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. Crie e salve um arquivo com um nome, como `revoke_config.txt`, para definir seus parâmetros de configuração de CRL.

```
{
   "CrlConfiguration":{
      "Enabled": true,
      "ExpirationInDays": 7,
      "S3BucketName": "bucket-name"
   }
}
```

> **Note**
>
> Ao atualizar uma CA de atestado de dispositivo Matter para habilitar CRLs, você deve configurá-la para omitir a extensão CDP dos certificados emitidos para ajudar a se adequar ao padrão Matter atual. Para fazer isso, defina seus parâmetros de configuração de CRL conforme ilustrado abaixo:
>
> ```
> {
>    "CrlConfiguration":{
>       "Enabled": true,
>       "ExpirationInDays": 7,
>       "S3BucketName": "bucket-name"
>       "CrlDistributionPointExtensionConfiguration":{
>          "OmitExtension": true
>       }
>    }
> }
> ```

3. Use o comando update-certificate-authority para e o arquivo de configuração de revogação para atualizar a 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. Inspecione novamente o status da 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
```

A saída confirma que a CA está configurada para usar uma 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
            }
        }
    }
}
```

Em alguns casos, talvez você queira adicionar suporte à revogação do OCSP em vez de habilitar uma CRL, como no procedimento anterior. Nesse caso, siga as seguintes etapas.

Para adicionar suporte ao OCSP a uma CA existente (AWS CLI)

1. Crie e salve um arquivo com um nome, como `revoke_config.txt`, para definir os parâmetros do OCSP.

```
{
   "OcspConfiguration":{
      "Enabled":true
   }
}
```

2. Use o comando update-certificate-authority para e o arquivo de configuração de revogação para atualizar a 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. Inspecione novamente o status da 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
```

A saída confirma que a CA está configurada para usar o 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**
>
> Você também pode configurar o suporte simultâneo para CRL e OCSP em uma CA.

Excluir a CA privada

Você pode excluir uma CA privada do AWS Management Console ou AWS CLI permanentemente. Você pode excluir uma, por exemplo, para substituí-la por uma nova CA que tem uma nova chave privada. Para excluir uma CA com segurança, siga estas etapas:

1. Crie a CA de substituição.
2. Assim que a nova CA privada estiver em produção, desative a antiga, mas não a exclua imediatamente.
3. Mantenha a CA antiga desabilitada até que todos os certificados emitidos por ela tenham expirado.
4. Exclua a CA antiga.

CA privada da AWS não verifica se todos os certificados emitidos expiraram antes de processar uma solicitação de exclusão. É possível gerar um relatório de auditoria para determinar quais certificados expiraram. Embora a CA esteja desativada, você pode revogar certificados, mas você não pode emitir novos.

Se for preciso excluir uma CA privada antes que todos os certificados emitidos por ela expirarem, recomendamos que você também revogue o certificado de CA. O certificado de CA será listado na CRL da CA pai, e a CA privada não será confiável para os clientes.

> **Important**
>
> Uma CA privada poderá ser excluída se estiver no estado `PENDING_CERTIFICATE`, `CREATING`, `EXPIRED`, `DISABLED` ou `FAILED`. Para excluir uma CA no estado `ACTIVE`, primeiro é necessário desabilitá-la ou a solicitação de exclusão resultará em uma exceção. Se você estiver excluindo uma CA privada no estado `PENDING_CERTIFICATE` ou `DISABLED`, poderá definir o tamanho do período de restauração de 7 a 30 dias, com 30 sendo o padrão. Durante esse período, o status é definido como `DELETED` e a CA é restaurável. Uma CA privada que é excluída enquanto no estado `CREATING` ou `FAILED` não tem nenhum período de restauração atribuído e não pode ser restaurada. Para ter mais informações, consulte Restaurar uma CA privada.
>
> Você não será cobrado por uma CA privada depois que ela for excluída. No entanto, se uma CA excluída for restaurada, você será cobrado pelo tempo entre a exclusão e a restauração. Para ter mais informações, consulte Definição de preço.

Como excluir uma CA privada (console)

1. Faça login na sua AWS conta e abra o CA privada da AWS console em https://console.aws.amazon.com/acm-pca/home.
2. Na página Autoridades de certificação privadas, escolha sua CA privada na lista.
3. Se a CA estiver no estado `ACTIVE`, você deverá desabilitá-la primeiro. No menu Actions (Ações), selecione Disable (Desabilitar). Quando solicitado, escolha Estou ciente do risco, continuar.
4. Para uma CA que não esteja no estado `ACTIVE`, escolha Ações, Excluir.
5. Se a CA estiver no estado `DISABLED`, `EXPIRED` ou `PENDING_CERTIFICATE`, a página Excluir CA permitirá especificar um período de restauração de 7 a 30 dias. Se a CA privada não estiver em um desses estados, ela não poderá ser restaurada, e a exclusão será permanente.

6. Escolha Excluir.
7. Se você tiver certeza de que deseja excluir a CA privada, selecione Permanently delete (Excluir permanentemente) quando for solicitado. O status da CA privada será alterado para `DELETED`. No entanto, é possível restaurar a CA privada antes do fim do período de restauração. Para verificar o período de restauração de uma CA privada no `DELETED` estado, ligue para a operação da API DescribeCerticateListCertificateAuthority ou Authority.

Como excluir uma CA privada (AWS CLI)

Use o comando delete-certificate-authority para excluir uma 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
```

Restaurar uma CA privada

É possível restaurar uma CA privada que foi excluída, desde que a CA permaneça dentro do período de restauração especificado durante a exclusão. O período de restauração é de 7 a 30 dias. Ao final desse período, a CA privada será excluída permanentemente. Para ter mais informações, consulte Excluir a CA privada. Não é possível restaurar uma CA privada que foi excluída permanentemente.

> **Note**
>
> Você não será cobrado por uma CA privada depois que ela for excluída. No entanto, se uma CA excluída for restaurada, você será cobrado pelo tempo entre a exclusão e a restauração. Para ter mais informações, consulte Definição de preço.

Restaurar uma CA privada (console)

Você pode usar o AWS Management Console para restaurar uma CA privada.

Para restaurar uma CA privada (console)

1. Faça login na sua AWS conta e abra o CA privada da AWS console em https://console.aws.amazon.com/acm-pca/home.

2. Na página Autoridades de certificação privadas, escolha sua CA privada excluída na lista.
3. No menu Ações, selecione Restaurar.
4. Na página Restaurar CA, escolha Restaurar novamente.
5. Se for bem-sucedido, o status da CA privada será definido para seu estado de pré-exclusão. Escolha Ações, Habilitar e Habilitar novamente para alterar o status para `ACTIVE`. Se a CA privada estava no estado `PENDING_CERTIFICATE` no momento da exclusão, você deverá importar um certificado CA para a CA privada para poder ativá-la.

Restaurar uma CA privada (AWS CLI)

Use o comando restore-certificate-authority para restaurar uma CA privada excluída que está no estado `DELETED`. As etapas a seguir discutem todo o processo necessário para excluir, restaurar e, depois, reativar uma CA privada.

Para excluir, restaurar e reativar uma CA privada (AWS CLI)

1. Exclua a CA privada.

   Execute o comando delete-certificate-authority para excluir a CA privada. Se o status da CA privada for `DISABLED` ou `PENDING_CERTIFICATE`, você poderá definir o parâmetro `--permanent-deletion-time-in-days` para especificar o período de restauração da CA privada de 7 a 30 dias. Se você não especificar um período de restauração, o padrão será de 30 dias. Se for bem-sucedido, esse comando definirá o status da CA privada como `DELETED`.

   > **Note**
   >
   > Para ser restaurável, o status da CA privada no momento da exclusão deverá ser `DISABLED` ou `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 a CA privada.

Execute o comando restore-certificate-authority para restaurar a CA privada. É necessário executar o comando antes que o período de restauração definido com o comando delete-certificate-authority expire. Se for bem-sucedido, o comando definirá o status da CA privada como seu status de pré-exclusão.

```
$ aws acm-pca restore-certificate-authority \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID
```

3. Torne a CA privada `ACTIVE`.

   Execute o comando update-certificate-authority para alterar o status da CA privada para `ACTIVE`.

```
$ aws acm-pca update-certificate-authority \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --status ACTIVE
```

Administração de certificados

Após criar e ativar uma autoridade de certificação (CA) privada e configurar o acesso a ela, você ou seus usuários autorizados poderão concluir as tarefas discutidas nesta seção. Se ainda não configurou políticas do AWS Identity and Access Management (IAM) para a CA, saiba mais sobre como fazer isso na seção Identity and Access Management deste guia. Para obter informações sobre como configurar o acesso à CA em cenários de conta única e entre contas, consulte Controlar o acesso a uma CA privada.

Tópicos

Emitir certificados de entidade final privada

Com uma CA privada em vigor, você pode solicitar certificados de entidade final privada do AWS Certificate Manager (ACM) ou do CA privada da AWS. Os recursos de ambos os serviços são comparados na tabela a seguir.

Recurso	ACM	CA privada da AWS
Emitir certificados de entidade final	✓ (usando RequestCertificate ou o console)	✓ (usando IssueCertificate)
Associação com balanceadores de carga e serviços da AWS voltados para a Internet.	✓	Não suportado
Renovação gerenciada de certificados	✓	Com suporte indireto por meio do ACM

Recurso	ACM	CA privada da AWS
Suporte a consoles	✓	Não suportado
Suporte à API	✓	✓
Compatibilidade da CLI	✓	✓

Quando o CA privada da AWS cria um certificado, ele segue um modelo que especifica o tipo do certificado e o comprimento do caminho. Se nenhum ARN de modelo for fornecido à instrução de API ou CLI que cria o certificado, o modelo EndEntityCertificate/V1 é aplicado por padrão. Para obter mais informações sobre modelos de certificado disponíveis, consulte Noções básicas sobre modelos de certificados.

Embora os certificados do ACM sejam projetados com base na confiança pública, o CA privada da AWS atende às necessidades da sua PKI privada. Consequentemente, é possível configurar certificados usando a API do CA privada da AWS e a CLI de maneiras não permitidas pelo ACM. Incluindo o seguinte:

- Criar um certificado com qualquer nome de requerente.
- Usar qualquer um dos algoritmos de chave privada e tamanhos de chave compatíveis.
- Usar qualquer um dos algoritmos de assinatura compatíveis.
- Especificar qualquer período de validade para a CA privada e os certificados privados.

Depois de criar um certificado TLS privado usando o CA privada da AWS, você pode importá-lo para o ACM e usá-lo com um serviço da AWS compatível.

> **Note**
>
> Os certificados criados com o procedimento abaixo, usando o issue-certificate comando ou com a ação da IssueCertificateAPI, não podem ser exportados diretamente para uso externoAWS. Porém, você pode usar sua CA privada para assinar certificados emitidos pelo ACM, e esses certificados podem ser exportados juntamente com suas chaves secretas. Para obter mais informações, consulte Solicitar um certificado prvado e Exportar certificado privado, no Guia do Usuário do ACM.

Emitir um certificado padrão (AWS CLI)

Você pode usar o comando da CA privada da AWS CLI issue-certificate ou a ação da API IssueCertificatepara solicitar um certificado de entidade final. Esse comando requer o nome de recurso da Amazon (ARN) da CA privada que você deseja usar para emitir o certificado. Você também deve gerar uma solicitação de assinatura de certificado (CSR) usando um programa como o OpenSSL.

Se você usar a API do CA privada da AWS ou a AWS CLI para emitir um certificado privado, este não será gerenciado, o que significa que não será possível usar o console do ACM, a CLI do ACM ou a API do ACM para visualizá-lo ou exportá-lo, e ele não será renovado automaticamente. Porém, você pode usar o comando get-certificate do PCA para recuperar os detalhes do certificado e, se for proprietário da CA, poderá criar um relatório de auditoria.

Considerações ao criar certificados

- Em conformidade com o RFC 5280, o tamanho do nome de domínio (tecnicamente, o Nome Comum) fornecido não pode exceder 64 octetos (caracteres), incluindo pontos. Para adicionar um nome de domínio mais longo, especifique-o no campo Nome alternativo do requerente, que oferece suporte a nomes de até 253 octetos de comprimento.
- Se você estiver usando a AWS CLI versão 1.6.3 ou posterior, use o prefixo `fileb://` ao especificar arquivos de entrada codificados em base64, como CSRs. Isso garante que o CA privada da AWS analise os dados corretamente.

O comando OpenSSL a seguir gera uma CSR e uma chave privada para um certificado:

```
$ openssl req -out csr.pem -new -newkey rsa:2048 -nodes -keyout private-key.pem
```

É possível inspecionar o conteúdo do CSR da seguinte maneira:

```
$ openssl req -in csr.pem -text -noout
```

A saída resultante deve ser semelhante a este exemplo 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
```

O comando abaixo cria um certificado. Como nenhum modelo foi especificado, um certificado básico de entidade final é emitido por padrão.

```
$ 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"
```

O ARN do certificado emitido é retornado:

```
{
   "CertificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
}
```

> **Note**
>
> O CA privada da AWS retorna imediatamente um ARN com um número de série ao receber o comando issue-certificate. Porém, o processamento do certificado ocorre de maneira assíncrona e ainda pode falhar. Se isso acontecer, um comando get-certificate usando o novo ARN também falhará.

Emita um certificado com um nome de requerente personalizado usando um modelo APIPassthrough

Neste exemplo, é emitido um certificado contendo elementos personalizados do nome do requerente. Além de fornecer uma CSR como a fornecida em Emitir um certificado padrão (AWS CLI), você passa dois argumentos adicionais para o comando issue-certificate: o ARN de um modelo APIPassthrough e um arquivo de configuração JSON que especifique os atributos personalizados e seus identificadores de objeto (OIDs). Não é possível usar `StandardAttributes` junto com `CustomAttributes`. Porém, você pode transmitir OIDs padrão como parte do `CustomAttributes`. Os OIDs de nome de requerente padrão estão listados na tabela a seguir (informações da RFC 4519 e do banco de dados de referência global de OIDs):

Nome do requerente	Abreviação	ID de objeto
countryName	c	2.5.4.6
commonName	cn	2.5.4.3
dnQualifier [qualificador de nome distinto]		2.5.4.46
generationQualifier		2.5.4.44
givenName		2.5.4.42
Initials		2.5.4.43
Locality	l	2.5.4.7
organizationName	o	2.5.4.10

Nome do requerente	Abreviação	ID de objeto
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	0.9.2342.19200300.100.1.25
userid		0.9.2342.19200300.100.1.1

O arquivo de configuração de amostra `api_passthrough_config.txt` contém o código a seguir:

```
{
  "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"
      }
    ]
  }
}
```

Use o seguinte comando para emitir o 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
```

O ARN do certificado emitido é retornado:

```
{
   "CertificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
}
```

Recupere o certificado localmente, da seguinte maneira:

```
$ 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
```

É possível inspecionar o conteúdo do certificado usando OpenSSL:

```
$ openssl x509 -in cert.pem -text -noout
```

> **Note**
>
> Também é possível criar uma CA privada que transmita atributos personalizados a cada certificado emitido.

Emita um certificado com extensões personalizadas usando um modelo APIPassthrough

Nesse exemplo, é emitido um certificado que contém extensões personalizadas. Para isso, você precisa transmitir três argumentos ao comando issue-certificate: o ARN de um modelo APIPassthrough e um arquivo de configuração JSON que especifique as extensões personalizadas e uma CSR, como a mostrada em Emitir um certificado padrão (AWS CLI).

O arquivo de configuração de amostra `api_passthrough_config.txt` contém o código a seguir:

```
{
  "Extensions": {
    "CustomExtensions": [
      {
        "ObjectIdentifier": "2.5.29.30",
        "Value": "MBWgEzARgg8ucGVybWl0dGVkLnRlc3Q=",
        "Critical": true
      }
    ]
  }
}
```

O certificado personalizado é emitido da seguinte maneira:

```
$ 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
```

O ARN do certificado emitido é retornado:

```
{
   "CertificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
}
```

Recupere o certificado localmente, da seguinte maneira:

```
$ 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
```

É possível inspecionar o conteúdo do certificado usando OpenSSL:

```
$ openssl x509 -in cert.pem -text -noout
```

Recuperar um certificado privado

Você pode usar a API CA privada da AWS e a AWS CLI para emitir um certificado privado. Se o fizer, poderá usar a AWS CLI ou a API do CA privada da AWS para recuperar esse certificado. Se você tiver usado o ACM para criar a sua CA privada para solicitar certificados, deverá usar o ACM para exportar o certificado e a chave privada criptografada. Para obter mais informações, consulte Exportar um certificado privado.

Para recuperar um certificado de entidade final

Use o comando get-certificate da AWS CLI para recuperar um certificado privado. Você também pode usar a operação GetCertificateda API. Convém formatar a saída com jq, um analisador semelhante a sed.

> **Note**
>
> Se você quiser revogar um certificado, poderá usar o comando get-certificate para recuperar o número de série em formato hexadecimal. Você também pode criar um relatório de auditoria para recuperar o número de série hexadecimal. Para ter mais informações, consulte Usar relatórios de auditoria com sua 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'
```

Esse comando gera o certificado e a cadeia de certificado no seguinte formato padrão.

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

Para recuperar um certificado de CA

É possível usar a API do CA privada da AWS e a AWS CLI para recuperar o certificado da autoridade de certificação (CA) para a sua CA privada. Execute o comando get-certificate-authority-certificate. Você também pode chamar a operação GetCertificateAuthorityCertificate . Convém formatar a saída com jq, um analisador semelhante a sed.

```
$ 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'
```

Esse comando gera o certificado de CA no seguinte formato padrão.

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

Listar certificados privados

Para listar seus certificados privados, gere um relatório de auditoria, recupere-o no bucket do S3 e analise seu conteúdo conforme necessário. Para obter mais informações sobre como criar relatórios de auditoria do CA privada da AWS, consulte Usar relatórios de auditoria com sua CA privada. Para

obter informações sobre como recuperar um objeto de um bucket do S3, consulte Baixar um objeto, no Guia do usuário do Amazon Simple Storage Service.

Os exemplos a seguir ilustram abordagens à criação de relatórios de auditoria e sua análise em busca de dados úteis. Os resultados são formatados em JSON, e os dados são filtrados usando o jq, um analisador ao estilo do sed.

1. Crie um relatório de auditoria.

O comando a seguir gera um relatório de auditoria para uma CA especificada.

```
$ 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
```

Se for bem-sucedido, ele retornará a ID e a localização do novo relatório de auditoria.

```
{
   "AuditReportId":"audit_report_ID",
   "S3Key":"audit-report/CA_ID/audit_report_ID.json"
}
```

2. Recupere e formate um relatório de auditoria.

Esse comando recupera um relatório de auditoria, exibe seu conteúdo na saída padrão e filtra os resultados para mostrar somente certificados emitidos em ou depois de 01/12/2020.

```
$ 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")'
```

Os itens retornados são semelhantes a:

```
{
   "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. Salve um relatório de auditoria localmente.

Se quiser fazer várias consultas, é conveniente salvar um relatório de auditoria em um arquivo local.

```
$ aws s3api get-object \
     --region region \
     --bucket bucket_name \
     --key audit-report/CA_ID/audit_report_ID.json > my_local_audit_report.json
```

O mesmo filtro anterior produz a mesma saída:

```
$ 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 um intervalo de datas

É possível consultar certificados emitidos dentro de um intervalo de datas, da seguinte maneira:

```
$ cat my_local_audit_report.json | jq '.[] | select(.issuedAt >= "2020-11-01"
 and .issuedAt <= "2020-11-10")'
```

O conteúdo filtrado é exibido na saída padrão:

```
{
   "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. Procure certificados seguindo um modelo especificado.

O comando a seguir filtra o conteúdo do relatório usando um ARN de modelo:

```
$ cat my_local_audit_report.json | jq '.[] | select(.templateArn == "arn:aws:acm-
pca:::template/RootCACertificate/V1")'
```

A saída mostra registros de certificado correspondentes:

```
{
   "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. Filtro de certificados revogados

Para localizar todos os certificados revogados, use o seguinte comando:

```
$ cat my_local_audit_report.json | jq '.[] | select(.revokedAt != null)'
```

Um certificado revogado é exibido da seguinte maneira:

```
{
   "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. Filtre usando uma expressão regular.

O comando a seguir procura nomes de requerentes que contenham a string “leaf”:

```
$ cat my_local_audit_report.json | jq '.[] | select(.subject|test("leaf"))'
```

Os registros de certificado correspondentes são retornados da seguinte maneira:

```
{
   "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"
}
```

Exportar um certificado privado e sua chave secreta

O CA privada da AWS não pode exportar diretamente um certificado privado que ele tenha assinado e emitido. Porém, você pode usar o AWS Certificate Manager para exportar esse certificado junto com sua chave secreta criptografada. O certificado é então totalmente portável para implantação em qualquer parte da sua PKI privada. Para obter mais informações, consulte Exportar um certificado privado, no o do usuário do AWS Certificate Manager.

Como benefício adicional, o AWS Certificate Manager fornece renovação gerenciada para certificados privados emitidos usando o console do ACM, a ação `RequestCertificate` da API do ACM ou o comando request-certificate na seção ACM do AWS CLI. Para obter mais informações sobre renovações, consulte Renovar certificados em uma PKI privada.

Revogar um certificado privado

Você pode revogar um CA privada da AWS certificado usando o AWS CLI comando revoke-certificate ou a ação da API. RevokeCertificate Um certificado talvez precise ser revogado antes de sua expiração programada se, por exemplo, sua chave secreta for comprometida ou o domínio associado se tornar inválido. Para que a revogação seja efetiva, o cliente que usa esse certificado precisa de uma maneira de verificar o status da revogação sempre que tentar criar uma conexão de rede segura.

O CA privada da AWS fornece dois mecanismos totalmente gerenciados para oferecer suporte à verificação do status de revogação: o Online Certificate Status Protocol (OCSP) e listas de revogação de certificados (CRLs). Com o OCSP, o cliente consulta um banco de dados de revogação autoritativo que retorna um status em tempo real. Com uma CRL, o cliente compara o certificado com uma lista de certificados revogados que ele baixa e armazena periodicamente. Clientes se recusam a aceitar certificados que foram revogados.

Tanto o OCSP quanto as CRLs dependem de informações de validação incorporadas em certificados. Por isso, uma CA emissora deve ser configurada para oferecer suporte a um ou a ambos os mecanismos antes da emissão. Para obter informações sobre como selecionar e implementar a revogação gerenciada por meio do CA privada da AWS, consulte Configurar um método de revogação de certificado.

Certificados revogados são sempre registrados em relatórios de auditoria do CA privada da AWS.

> **Note**
>
> Emissores de certificados entre contas precisam de permissões adicionais para revogar os certificados emitidos; caso contrário, o proprietário da CA deverá fazer a revogação. Para permitir a revogação por emissores de várias contas, o administrador da CA deve criar dois compartilhamentos de RAM, ambos apontando para a mesma CA:
>
> 1. Um compartilhamento com a permissão `AWSRAMRevokeCertificateCertificateAuthority`.

2. Um compartilhamento com a permissão `AWSRAMDefaultPermissionCertificateAuthority`.

Para revogar um certificado

Use a ação da RevokeCertificateAPI ou o comando revoke-certificate para revogar um certificado PKI privado. O número de série deve estar no formato hexadecimal. É possível recuperar o número de série chamando o comando get-certificate. O comando `revoke-certificate` não retorna uma resposta.

```
$ 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 revogados e o OCSP

As respostas do OCSP podem levar até 60 minutos para refletir o novo status quando um certificado é revogado. Em geral, o OCSP tende a oferecer suporte à distribuição mais rápida de informações de revogação pois, ao contrário das CRLs, que podem ser armazenadas em cache pelos clientes por vários dias, as respostas do OCSP normalmente não são armazenadas em cache pelos clientes.

Certificados revogados em uma CRL

Uma CRL normalmente é atualizada aproximadamente 30 minutos depois que um certificado é revogado. Se, por algum motivo, uma atualização da CRL falhar, o CA privada da AWS realizará novas tentativas a cada 15 minutos.

Com a Amazon CloudWatch, você pode criar alarmes para as métricas `CRLGenerated` e. `MisconfiguredCRLBucket` Para obter mais informações, consulte CloudWatchMétricas suportadas. Para obter mais informações sobre a criação e a configuração de CRLs, consulte Planejar uma lista de revogação de certificados (CRL).

O exemplo a seguir mostra um certificado revogado em uma lista de revogação 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 revogados em um relatório de auditoria

Todos os certificados, incluindo certificados revogados, são incluídos no relatório de auditoria de uma CA privada. O exemplo a seguir mostra um relatório de auditoria com um certificado revogado e um emitido. Para ter mais informações, consulte Usar relatórios de auditoria com sua 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"
   }
]
```

Automatizar a exportação de um certificado renovado

Ao usar o CA privada da AWS para criar uma CA, você pode importar essa CA no AWS Certificate Manager e deixar o ACM gerenciar a emissão e a renovação do certificado. Se um certificado que está sendo renovado estiver associado a um serviço integrado, o serviço aplicará o novo certificado perfeitamente. Porém, se o certificado foi originalmente exportado para uso em outro local do seu ambiente de PKI (por exemplo, em um servidor ou equipamento on-premises), você precisará exportá-lo novamente após a renovação.

Para ver um exemplo de solução que automatiza o processo de exportação do ACM usando Amazon e EventBridge AWS Lambda, consulte Automatização da exportação de certificados renovados.

Noções básicas sobre modelos de certificados

O CA privada da AWS usa modelos de configuração para emitir certificados de CA e certificados de entidade final. Quando você emite um certificado de CA do console do PCA, o modelo de certificado de CA raiz ou subordinado apropriado é aplicado automaticamente.

Se você usar a CLI ou a API para emitir um certificado, poderá fornecer um ARN de modelo como parâmetro para a ação `IssueCertificate`. Se você não fornecer um ARN, o modelo `EndEntityCertificate/V1` será aplicado por padrão. Para obter mais informações, consulte a documentação do comando IssueCertificateAPI e issue-certificate.

> **Note**
>
> Usuários do AWS Certificate Manager (ACM) com acesso compartilhado entre contas a uma CA privada podem emitir certificados gerenciados que são assinados pela CA. Emissores entre contas estão limitados por uma política baseada em recursos e têm acesso somente aos seguintes modelos de certificado de entidade final:
>
> - EndEntityCertificate/V1
> - EndEntityClientAuthCertificate/V1
> - EndEntityServerAuthCertificate/V1
> - BlankEndEntityCertificate_Passagem de API/v1
> - BlankEndEntityCertificate_APICSRPassthrough/v1
> - Certificado CA subordinado_ 0/V1 PathLen
>
> Para ter mais informações, consulte Políticas baseadas em recurso.

Tópicos

Variedades de modelos

O CA privada da AWS oferece suporte a quatro variedades de modelos.

- Modelos base

  Modelos predefinidos nos quais nenhum parâmetro de passagem é permitido.

- Modelos CSRPassthrough

  Modelos que estendem suas versões de modelo base correspondentes, permitindo a passagem da CSR. As extensões na CSR que é usada para emitir o certificado são copiadas para o certificado emitido. Nos casos em que a CSR contém valores de extensão em conflito com a definição do modelo, esta sempre tem a prioridade mais alta. Para obter mais detalhes sobre prioridade, consulte Orden de operações dos modelos.

- Modelos APIPassthrough

  Modelos que estendem suas versões de modelo base correspondentes, permitindo a passagem da API. Valores dinâmicos conhecidos pelo administrador ou por outros sistemas intermediários podem não ser conhecidos pela entidade que solicita o certificado, podem ser impossíveis de definir em um modelo e talvez não estejam disponíveis na CSR. Porém, o administrador da CA pode recuperar informações adicionais de outra fonte de dados, como um Active Directory, para concluir a solicitação. Por exemplo, se uma máquina não souber a qual unidade organizacional ela pertence, o administrador poderá pesquisar as informações no Active Directory e adicioná-las à solicitação de certificado, incluindo as informações em uma estrutura JSON.

  Os valores no parâmetro `ApiPassthrough` da ação `IssueCertificate` são copiados para o certificado emitido. Nos casos em que o parâmetro `ApiPassthrough` contém informações em conflito com a definição do modelo, esta sempre tem a prioridade mais alta. Para obter mais detalhes sobre prioridade, consulte Orden de operações dos modelos.

- Modelos APICSRPassthrough

  Modelos que estendem suas versões de modelo base correspondentes, permitindo a passagem da API e da CSR. As extensões na CSR usadas para emitir o certificado são copiadas para o certificado emitido, e os valores no parâmetro `ApiPassthrough` da ação `IssueCertificate` também são copiados. Nos casos em que a definição de modelo, os valores de passagem da API e as extensões de passagem da CSR exibem conflito, a definição de modelo tem a prioridade mais alta, seguida pelos valores de passagem da API e depois pelas extensões de passagem da CSR. Para obter mais detalhes sobre prioridade, consulte Orden de operações dos modelos.

As tabelas abaixo listam os tipos de modelo compatíveis com o CA privada da AWS, com links para suas definições.

> **Note**
>
> Para obter informações sobre modelos de ARNs em GovCloud regiões, consulte AWS Private Certificate Authorityo Guia do AWS GovCloud (US) usuário.

Modelos base

Nome do modelo	ARN do modelo	Tipo de certificado
CodeSigningCertificate/V1	`arn:aws:acm-pca:::template/CodeSigningCertificate/V1`	Assinatura de código
EndEntityCertificate/V1	`arn:aws:acm-pca:::template/EndEntityCertificate/V1`	Entidade final
EndEntityClientAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate/V1`	Entidade final
EndEntityServerAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate/V1`	Entidade final
SigningCertificateOCSP/V1	`arn:aws:acm-pca:::template/OCSPSigningCertificate/V1`	Assinatura OCSP
RootCACertificate/V1	`arn:aws:acm-pca:::template/RootCACertificate/V1`	CA
Certificado CA subordinado_ 0/V1 PathLen	`arn:aws:acm-pca:::template/Subordina`	CA

Nome do modelo	ARN do modelo	Tipo de certificado
	teCACertificate_Pathlen0/V1	
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

Modelos CSRPassthrough

Nome do modelo	ARN do modelo	Tipo de certificado
BlankEndEntityCertificate_CSRPassthrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CSRPassthrough/V1	Entidade final
BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassthrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassthrough/V1	Entidade final
BlankSubordinateCertificado CA _ 0_CSR Passthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubo	CA

Nome do modelo	ARN do modelo	Tipo de certificado
	rdinateCACertificate_PathLen0_CSRPassthrough/V1	
BlankSubordinateCertificado CA _1_CSR Passthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
BlankSubordinateCertificado CA _2_CSR Passthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
BlankSubordinateCertificado CA _3_CSR Passthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA
CodeSigningCertificate_CSRPassthrough/v1	arn:aws:acm-pca:::template/CodeSigningCertificate_CSRPassthrough/V1	Assinatura de código
EndEntityCertificate_CSRPassthrough/v1	arn:aws:acm-pca:::template/EndEntityCertificate_CSRPassthrough/V1	Entidade final
EndEntityClientAuthCertificate_CSRPassthrough/v1	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_CSRPassthrough/V1	Entidade final

Nome do modelo	ARN do modelo	Tipo de certificado
EndEntityServerAuthCertificate_CSRPassthrough/v1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_CSRPassthrough/V1`	Entidade final
OCSP SigningCertificate_CSRPassthrough/v1	`arn:aws:acm-pca:::template/OCSPSigningCertificate_CSRPassthrough/V1`	Assinatura OCSP
Certificado CA subordinado_0_CSR Passthrough/v1 PathLen	`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
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

Modelos APIPassthrough

Nome do modelo	ARN do modelo	Tipo de certificado
BlankEndEntityCertificate_Passagem de API/v1	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_APIPassthrough/V1`	Entidade final
BlankEndEntityCertificate_CriticalBasicConstraints _APIPassthrough/v1	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough/V1`	Entidade final
CodeSigningCertificate_Passagem de API/v1	`arn:aws:acm-pca:::template/CodeSigningCertificate_APIPassthrough/V1`	Assinatura de código
EndEntityCertificate_Passagem de API/v1	`arn:aws:acm-pca:::template/EndEntityCertificate_APIPassthrough/V1`	Entidade final
EndEntityClientAuthCertificate_Passagem de API/v1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_APIPassthrough/V1`	Entidade final
EndEntityServerAuthCertificate_Passagem de API/v1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APIPassthrough/V1`	Entidade final
OCSP SigningCertificate _APIPassthrough/v1	`arn:aws:acm-pca:::template/OCSPSigni`	Assinatura OCSP

Nome do modelo	ARN do modelo	Tipo de certificado
	ngCertificate_APIP assthrough/V1	
RootCACertificate_APIPassth rough/V1	arn:aws:acm-pca::: template/RootCACer tificate_APIPassth rough/V1	CA
BlankRootCAcertificate_apip assthrough/v1	arn:aws:acm-pca::: template/BlankRoot CACertificate_APIP assthrough/V1	CA
BlankRootCertificado CA _ 0_APIPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len0_APIPassthrough/V1	CA
BlankRootCertificado CA_ 1_APIPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len1_APIPassthrough/V1	CA
BlankRootCertificado CA _ 2_APIPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len2_APIPassthrough/V1	CA
BlankRootCertificado CA _ 3_APIPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len3_APIPassthrough/V1	CA

Nome do modelo	ARN do modelo	Tipo de certificado
Certificado CA subordinado _ PathLen 0_APIPassthrough/v1	arn:aws:acm-pca::: template/SubordinateCACertificate_PathLen0_APIPassthrough/V1	CA
BlankSubordinateCertificado CA _ 0_APIPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1	CA
Certificado CA subordinado _ PathLen 1_APIPassthrough/v1	arn:aws:acm-pca::: template/SubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
BlankSubordinateCertificado CA_ 1_APIPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
Certificado CA subordinado _ PathLen 2_APIPassthrough/v1	arn:aws:acm-pca::: template/SubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
BlankSubordinateCertificado CA _ 2_APIPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	CA

Nome do modelo	ARN do modelo	Tipo de certificado
Certificado CA subordinado _ PathLen 3_APIPassthrough/v1	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen3_APIPassthrough/ V1	CA
BlankSubordinateCertificado CA _ 3_APIPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen3_APIPas sthrough/V1	CA

Modelos APICSRPassthrough

Nome do modelo	ARN do modelo	Tipo de certificado
BlankEndEntityCertificate_A PICSRPassthrough/v1	arn:aws:acm-pca::: template/BlankEndE ntityCertificate_A PICSRPassthrough/V1	Entidade final
BlankEndEntityCertificate_ CriticalBasicConstraints _APICSRPassthrough/v1	arn:aws:acm-pca::: template/BlankEndE ntityCertificate_C riticalBasicConstr aints_APICSRPassth rough/V1	Entidade final
CodeSigningCertificate_APIC SRPassthrough/v1	arn:aws:acm-pca::: template/CodeSigni ngCertificate_APIC SRPassthrough/V1	Assinatura de código

Nome do modelo	ARN do modelo	Tipo de certificado
EndEntityCertificate_APICSR Passthrough/v1	`arn:aws:acm-pca:::template/EndEntityCertificate_APICSRPassthrough/V1`	Entidade final
EndEntityClientAuthCertificate_APICSRPassthrough/v1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_APICSRPassthrough/V1`	Entidade final
EndEntityServerAuthCertificate_APICSRPassthrough/v1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APICSRPassthrough/V1`	Entidade final
OCSP SigningCertificate _APICSRPassthrough/v1	`arn:aws:acm-pca:::template/OCSPSigningCertificate_APICSRPassthrough/V1`	Assinatura OCSP
Certificado CA subordinado_ 0_APICSR PathLen Passthrough/ 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

Nome do modelo	ARN do modelo	Tipo de certificado
Certificado CA subordinado_ 1_APICSR PathLen Passthrough/ v1	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen1_APICSRPasst hrough/V1	CA
BlankSubordinateCertificado CA _ 1_APICSRPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen1_APICSR Passthrough/V1	CA
Certificado CA subordinado _ 2_APICSRPassThrough/ PathLen 3_APIPassthroughV1 PathLen	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen2_APICSRPasst hrough/V1	CA
BlankSubordinateCertificado CA _ 2_APICSRPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen2_APICSR Passthrough/V1	CA
Certificado CA subordinado _ 3_APICSR PathLen Passthrough/ v1	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen3_APICSRPasst hrough/V1	CA
BlankSubordinateCertificado CA _ 3_APICSRPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen3_APICSR Passthrough/V1	CA

Orden de operações dos modelos

As informações contidas em um certificado emitido podem ser provenientes de quatro fontes: definição de modelo, passagem de API, passagem de CSR e configuração da CA.

Os valores de passagem de API apenas são respeitados quando você usa um modelo de passagem de API ou de passagem de APICSR. A passagem de CSR apenas é respeitada quando você usa um modelo de passagem de CSR ou de APICSR. Quando essas fontes de informação estão em conflito, uma regra geral geralmente é aplicável: para cada valor de extensão, a definição de modelo tem a prioridade mais alta, seguida por valores de passagem de API e depois por extensões de passagem de CSR.

Exemplos

1. A definição do modelo para EndEntityClientAuthCertificate_apiPassthrough define a ExtendedKeyUsage extensão com um valor de “autenticação de servidor web TLS, autenticação de cliente web TLS”. Se ExtendedKeyUsage for definido na CSR ou no `IssueCertificate ApiPassthrough` parâmetro, o `ApiPassthrough` valor para ExtendedKeyUsage será ignorado porque a definição do modelo tem prioridade, e o valor CSR para o ExtendedKeyUsage valor será ignorado porque o modelo não é uma variedade de passagem de CSR.

   > **Note**
   >
   > Mesmo assim, a definição de modelo copia outros valores da CSR, como o requerente e o nome alternativo do requerente. Esses valores ainda são retirados da CSR, mesmo que o modelo não seja uma variedade de passagem da CSR, pois a definição de modelo sempre tem a prioridade mais alta.

2. A definição do modelo para EndEntityClientAuthCertificate_APICSrPassthrough define a extensão Subject Alternative Name (SAN) como sendo copiada da API ou CSR. Se a extensão de SAN for definida na CSR e fornecida no parâmetro `IssueCertificate ApiPassthrough`, o valor de passagem de API terá prioridade, pois estes têm prioridade sobre valores de passagem de CSR.

Definições de modelos

As seções a seguir fornecem detalhes de configuração sobre os modelos de certificado do CA privada da AWS com suporte.

BlankEndEntityCertificateDefinição de _APIPassthrough/v1

Com modelos de certificado de entidade final em branco, você pode emitir certificados de entidade final com apenas restrições básica X.509 presentes. Esse é o certificado de entidade final mais simples que o CA privada da AWS pode emitir, mas pode ser personalizado usando a estrutura da API. A extensão de restrições básicas define se o certificado é ou não um certificado de CA. Um modelo de certificado de entidade final em branco impõe um valor de FALSE a restrições básicas, para garantir que um certificado de entidade final seja emitido, e não um certificado de CA.

Você pode usar modelos de passagem em branco para emitir certificados de cartão inteligente que exigem valores específicos para uso da chave (KU) e uso estendido da chave (EKU). Por exemplo, Uso estendido de chave pode exigir Autenticação de cliente e Login com cartão inteligente, enquanto Uso de chave pode exigir Assinatura digital, Não repúdio e Codificação da chave. Ao contrário de outros modelos de passagem, os modelos de certificado de entidade final em branco permitem a configuração de extensões KU e EKU, onde KU pode ser qualquer um dos nove valores suportados (DigitalSignature, NonRepudiation, KeyEncipherment, DataEncipherment, KeyAgreement, CrlSign, EncipherOnly e DecipherOnly) e EKU pode ser qualquer um dos valores suportados (ServerAuth keyCertSign, ClientAuth, codesign, Proteção de e-mail, registro de data e hora e assinatura OCSP), além de extensões personalizadas.

BlankEndEntityCertificate_Passagem de API/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankEndEntityCertificateDefinição de _APICSRPassthrough/v1

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankEndEntityCertificate_APICSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankEndEntityCertificate_ CriticalBasicConstraints _definição de APICSRPassthrough/v1

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _APICSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítico, CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]

Parâmetro X509v3	Valor
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagegem da configuração de CA, API ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankEndEntityCertificate_ CriticalBasicConstraints _definição de APIPassthrough/v1

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _APIPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítico, CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagegem da configuração de CA ou API]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankEndEntityCertificate_ CriticalBasicConstraints _definição de CSRPassthrough/v1

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _CSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítico, CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankEndEntityCertificateDefinição de _CSRPassthrough/v1

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankEndEntityCertificate_CSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 0_CSRPassthrough/v1 definição PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 0_CSR Passthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 0`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 0_definição de APICSRPassthrough/v1 PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 0_APICSRPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 0`

Parâmetro X509v3	Valor
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 0_definição de APIPassthrough/v1 PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 0_APIPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 0`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

BlankSubordinateCAcertificate_ 1_definição de APIPassthrough/v1 PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA_ 1_APIPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 1`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 1_definição CSRPassthrough/v1 PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 1_CSR Passthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 1`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 1_definição de APICSRPassthrough/v1 PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 1_APICSRPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 1`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 2_definição de APIPassthrough/v1 PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 2_APIPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 2`

Parâmetro X509v3	Valor
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 2_definição CSRPassthrough/v1 PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 2_CSR Passthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 2`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 2_definição de APICSRPassthrough/v1 PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 2_APICSRPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 2`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 3_definição de APIPassthrough/v1 PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 3_APIPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 3`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 3_CSRPassthrough/v1 definição PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 3_CSR Passthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 3`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

BlankSubordinateCAcertificate_ 3_definição de APICSRPassthrough/v1 PathLen

Para obter informações gerais sobre modelos em branco, consulte BlankEndEntityCertificateDefinição de _APIPassthrough/v1.

BlankSubordinateCertificado CA _ 3_API CSR Passthrough PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 3`

Parâmetro X509v3	Valor
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

CodeSigningCertificateDefinição de /V1

Esse modelo é usado para criar certificados para assinatura de código. Você pode usar certificados de assinatura de código no CA privada da AWS com qualquer solução de assinatura de código baseada em uma infraestrutura de CA privada. Por exemplo, os clientes que usam a assinatura de código para AWS IoT podem gerar um certificado de assinatura de código com o CA privada da AWS e importá-lo no AWS Certificate Manager. Para obter mais informações, consulte Para que serve a assinatura de códigoAWS IoT? e obtenha e importe um certificado de assinatura de código.

CodeSigningCertificate/V1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	`CA:FALSE`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital
Uso estendido de chave	Crítica, assinatura de código
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

*Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

CodeSigningCertificateDefinição de _APICSRPassthrough/v1

Esse modelo estende CodeSigningCertificate /V1 para oferecer suporte a valores de passagem de API e CSR.

CodeSigningCertificate_APICSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	`CA:FALSE`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital
Uso estendido de chave	Crítica, assinatura de código
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

CodeSigningCertificateDefinição de _APIPassthrough/v1

Este modelo é idêntico ao modelo `CodeSigningCertificate` com uma diferença: nele, o CA privada da AWS transmitirá extensões adicionais da solicitação por meio da API ao certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na API.

CodeSigningCertificate_Passagem de API/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	`CA:FALSE`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital
Uso estendido de chave	Crítica, assinatura de código
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

CodeSigningCertificateDefinição de _CSRPassthrough/v1

Este modelo é idêntico ao modelo `CodeSigningCertificate` com uma diferença: neste modelo, o CA privada da AWS passará extensões adicionais da solicitação de assinatura de certificado (CSR) para o certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na CSR.

CodeSigningCertificate_CSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	`CA:FALSE`
Identificador de chave da autoridade	[SKI do certificado de CA]

Parâmetro X509v3	Valor
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital
Uso estendido de chave	Crítica, assinatura de código
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

*Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityCertificateDefinição de /V1

Este modelo é usado para criar certificados para entidades finais, como sistemas operacionais ou servidores Web.

EndEntityCertificate/V1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de servidor Web TLS, autentica ção de cliente Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

*Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityCertificateDefinição de _APICSRPassthrough/v1

Esse modelo estende EndEntityCertificate /V1 para oferecer suporte a valores de passagem de API e CSR.

EndEntityCertificate_APICSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de servidor Web TLS, autentica ção de cliente Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityCertificateDefinição de _APIPassthrough/v1

Este modelo é idêntico ao modelo `EndEntityCertificate` com uma diferença: nele, o CA privada da AWS transmitirá extensões adicionais da solicitação por meio da API ao certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na API.

EndEntityCertificate_Passagem de API/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de servidor Web TLS, autentica ção de cliente Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityCertificateDefinição de _CSRPassthrough/v1

Este modelo é idêntico ao modelo `EndEntityCertificate` com uma diferença: neste modelo, o CA privada da AWS passará extensões adicionais da solicitação de assinatura de certificado (CSR) para o certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na CSR.

EndEntityCertificate_CSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	CA:FALSE

Parâmetro X509v3	Valor
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de servidor Web TLS, autentica ção de cliente Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

*Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityClientAuthCertificateDefinição de /V1

Este modelo difere do `EndEntityCertificate` apenas no valor de uso de chave estendida, que o restringe à autenticação de cliente Web TLS.

EndEntityClientAuthCertificate/V1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	CA:`FALSE`
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de cliente Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

*Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityClientAuthCertificateDefinição de _APICSRPassthrough/v1

Esse modelo estende EndEntityClientAuthCertificate /V1 para oferecer suporte a valores de passagem de API e CSR.

EndEntityClientAuthCertificate_APICSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de cliente Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityClientAuthCertificateDefinição de _APIPassthrough/v1

Este modelo é idêntico ao modelo `EndEntityClientAuthCertificate` com uma diferença. Nesse modelo, o CA privada da AWS transmitirá extensões adicionais por meio da API ao certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na API.

EndEntityClientAuthCertificate_Passagem de API/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de cliente Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityClientAuthCertificateDefinição de _CSRPassthrough/v1

Este modelo é idêntico ao modelo `EndEntityClientAuthCertificate` com uma diferença. Neste modelo, o CA privada da AWS passará extensões adicionais da solicitação de assinatura de certificado (CSR) para o certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na CSR.

EndEntityClientAuthCertificate_CSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]

Parâmetro X509v3	Valor
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de cliente Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

*Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityServerAuthCertificateDefinição de /V1

Este modelo difere do `EndEntityCertificate` apenas no valor de uso de chave estendida, que o restringe à autenticação do servidor Web TLS.

EndEntityServerAuthCertificate/V1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	CA:`FALSE`
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de servidor Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

*Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityServerAuthCertificateDefinição de _APICSRPassthrough/v1

Esse modelo estende EndEntityServerAuthCertificate /V1 para oferecer suporte a valores de passagem de API e CSR.

EndEntityServerAuthCertificate_APICSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de servidor Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityServerAuthCertificateDefinição de _APIPassthrough/v1

Este modelo é idêntico ao modelo `EndEntityServerAuthCertificate` com uma diferença. Nesse modelo, o CA privada da AWS transmitirá extensões adicionais por meio da API ao certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na API.

EndEntityServerAuthCertificate_Passagem de API/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]

Parâmetro X509v3	Valor
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de servidor Web TLS
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

EndEntityServerAuthCertificateDefinição de _CSRPassthrough/v1

Este modelo é idêntico ao modelo `EndEntityServerAuthCertificate` com uma diferença. Neste modelo, o CA privada da AWS passará extensões adicionais da solicitação de assinatura de certificado (CSR) para o certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na CSR.

EndEntityServerAuthCertificate_CSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	CA:FALSE
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, criptografia de chave
Uso estendido de chave	Autenticação de servidor Web TLS

Parâmetro X509v3	Valor
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

*Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Definição de SigningCertificate OCSP/V1

Este modelo é usado para criar certificados para assinar respostas OCSP. O modelo é idêntico ao modelo `CodeSigningCertificate`, exceto que o valor de uso de chave estendida especifica a assinatura OCSP em vez da assinatura de código.

SigningCertificateOCSP/V1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	`CA:FALSE`
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital
Uso estendido de chave	Crítica, assinatura OCSP
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

*Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Definição de OCSP SigningCertificate _APICSRPassthrough/v1

Esse modelo estende o SigningCertificate OCSP/V1 para suportar valores de passagem de API e CSR.

OCSP SigningCertificate _APICSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	`CA:FALSE`
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital
Uso estendido de chave	Crítica, assinatura OCSP
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Definição de OCSP SigningCertificate _APIPassthrough/v1

Este modelo é idêntico ao modelo `0CSPSigningCertificate` com uma diferença. Nesse modelo, o CA privada da AWS transmitirá extensões adicionais por meio da API ao certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na API.

OCSP SigningCertificate _APIPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	`CA:FALSE`
Identificador de chave da autoridade	[SKI do certificado CA]

Parâmetro X509v3	Valor
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital
Uso estendido de chave	Crítica, assinatura OCSP
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Definição de OCSP SigningCertificate _CSRPassthrough/v1

Este modelo é idêntico ao modelo `0CSPSigningCertificate` com uma diferença. Neste modelo, o CA privada da AWS passará extensões adicionais da solicitação de assinatura de certificado (CSR) para o certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na CSR.

OCSP SigningCertificate _CSRPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	`CA:FALSE`
Identificador de chave da autoridade	[SKI do certificado CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital
Uso estendido de chave	Crítica, assinatura OCSP
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

*Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Definição de RootCACertificate/V1

Esse modelo é usado para emitir certificados CA raiz autoassinados. Os certificados CA incluem uma extensão de restrições básicas críticas com o campo CA definido como TRUE para designar que o certificado pode ser utilizado para emitir certificados CA. O modelo não especifica um comprimento de caminho (pathLenConstraint) porque isso pode inibir a expansão futura da hierarquia. O uso estendido de chave é excluído para impedir o uso do certificado CA como um certificado de cliente ou de servidor TLS. Nenhuma informação de CRL é especificada porque um certificado autoassinado não pode ser revogado.

RootCACertificate/V1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Assinatura digital crítica keyCertSign, sinal CRL
Pontos de distribuição de CRL	N/D

Definição de RootCACertificate_APIPassthrough/V1

Esse modelo estende RootCACertificate/V1 para oferecer suporte a valores de passagem da API.

RootCACertificate_APIPassthrough/V1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]

Parâmetro X509v3	Valor
Restrições básicas	Crítica, `CA:TRUE`
Identificador de chave da autoridade	[Passagem de API]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Assinatura digital crítica keyCertSign, sinal CRL
Pontos de distribuição de CRL*	N/D

BlankRootDefinição de CAcertificate_apipassthrough/v1

Com modelos de certificado raiz em branco, você pode emitir certificados raiz com apenas restrições básicas X.509 presentes. Esse é o certificado raiz mais simples que CA privada da AWS pode ser emitido, mas pode ser personalizado usando a estrutura da API. A extensão de restrições básicas define se o certificado é ou não um certificado CA. Um modelo de certificado raiz em branco impõe um valor de `TRUE` quatro restrições básicas para garantir que um certificado CA raiz seja emitido.

Você pode usar modelos raiz de passagem em branco para emitir certificados raiz que exigem valores específicos para o uso da chave (KU). Por exemplo, o uso da chave pode exigir `keyCertSign` `ecRLSign`, mas não`digitalSignature`. Diferentemente do outro modelo de certificado de passagem raiz que não está em branco, os modelos de certificado raiz em branco permitem a configuração da extensão KU, em que KU pode ser qualquer um dos nove valores suportados (`digitalSignaturenonRepudiation`,`keyEncipherment`,`dataEncipherment`,`keyAgreement`,`keyCe` `cRLSignencipherOnly`, `edecipherOnly`).

BlankRootCAcertificate_apipassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`
Identificador de chave do requerente	[Derivado da CSR]

BlankRootCAcertificate_ 0_definição de APIPassthrough/v1 PathLen

Para obter informações gerais sobre modelos de CA raiz em branco, consulte???.

BlankRootCertificado CA _ 0_APIPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 0`
Identificador de chave do requerente	[Derivado da CSR]

BlankRootCAcertificate_ 1_definição de APIPassthrough/v1 PathLen

Para obter informações gerais sobre modelos de CA raiz em branco, consulte???.

BlankRootCertificado CA_ 1_APIPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 1`
Identificador de chave do requerente	[Derivado da CSR]

BlankRootCAcertificate_ 2_definição de APIPassthrough/v1 PathLen

Para obter informações gerais sobre modelos de CA raiz em branco, consulte???.

BlankRootCertificado CA _ 2_APIPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]

Parâmetro X509v3	Valor
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 2`
Identificador de chave do requerente	[Derivado da CSR]

BlankRootCAcertificate_ 3_definição de APIPassthrough/v1 PathLen

Para obter informações gerais sobre modelos de CA raiz em branco, consulte???.

BlankRootCertificado CA _ 3_APIPassthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 3`
Identificador de chave do requerente	[Derivado da CSR]

Definição de Certificado Subordinado_ PathLen 0/V1

Esse modelo é usado para emitir certificados CA subordinados com um comprimento de caminho de`0`. Os certificados CA incluem uma extensão de restrições básicas críticas com o campo CA definido como `TRUE` para designar que o certificado pode ser utilizado para emitir certificados CA. O uso estendido da chave não está incluído, o que impede que o certificado CA seja usado como um certificado de cliente ou de servidor TLS.

Para obter mais informações sobre caminhos de certificação, consulte Configurar restrições de comprimento no caminho de certificação.

Certificado CA subordinado_ 0/V1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]

Parâmetro X509v3	Valor
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 0`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

*Os pontos de distribuição de CRL são incluídos nos certificados emitidos com esse modelo somente se a CA estiver configurada com a geração de CRL ativada.

Certificação CA subordinada_ 0_definição de PathLen APICSRPassthrough/v1

Esse modelo estende SubordinateCacertificate_ PathLen 0/V1 para suportar valores de passagem de API e CSR.

Certificado CA subordinado_ 0_APICSR PathLen Passthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 0`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Certificado subordinado CA_ 0_definição de API passthrough/v1 PathLen

Esse modelo estende SubordinateCacertificate_ PathLen 0/V1 para oferecer suporte aos valores de passagem da API.

Certificado CA subordinado _ PathLen 0_APIPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 0`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Certificação CA subordinada_ 0_definição PathLen CSRPassthrough/v1

Este modelo é idêntico ao modelo `SubordinateCACertificate_PathLen0` com uma diferença: neste modelo, o CA privada da AWS passará extensões adicionais da solicitação de assinatura de certificado (CSR) para o certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na CSR.

> **Note**
>
> Um CSR que contém extensões adicionais personalizadas deve ser criado fora do CA privada da AWS.

Certificado CA subordinado _ 0_CSR Passthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 0`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

*Os pontos de distribuição da CRL serão incluídos nos certificados emitidos com este modelo apenas se a CA estiver configurada com a geração de CRL habilitada.

Certificação Subordinate_ 1/V1 definição PathLen

Esse modelo é usado para emitir certificados CA subordinados com um comprimento de caminho de1. Os certificados CA incluem uma extensão de restrições básicas críticas com o campo CA definido como `TRUE` para designar que o certificado pode ser utilizado para emitir certificados CA. O uso estendido da chave não está incluído, o que impede que o certificado CA seja usado como um certificado de cliente ou de servidor TLS.

Para obter mais informações sobre caminhos de certificação, consulte Configurar restrições de comprimento no caminho de certificação.

Certificado CA subordinado_ 1/V1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 1`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

*Os pontos de distribuição da CRL serão incluídos nos certificados emitidos com este modelo apenas se a CA estiver configurada com a geração de CRL habilitada.

Certificação CA subordinada_ 1_definição de PathLen APICSRPassthrough/v1

Esse modelo estende SubordinateCacertificate_ PathLen 1/V1 para suportar valores de passagem de API e CSR.

Certificado CA subordinado_ 1_APICSR PathLen Passthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 1`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]

Parâmetro X509v3	Valor
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Certificado subordinado CA_ 1_definição de API passthrough/v1 PathLen

Esse modelo estende SubordinateCacertificate_ PathLen 0/V1 para oferecer suporte aos valores de passagem da API.

Certificado CA subordinado _ PathLen 1_APIPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 1`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Certificação CA subordinada_ 1_definição CSR Passthrough/v1 PathLen

Este modelo é idêntico ao modelo `SubordinateCACertificate_PathLen1` com uma diferença: neste modelo, o CA privada da AWS passará extensões adicionais da solicitação de assinatura de certificado (CSR) para o certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na CSR.

> **Note**
>
> Um CSR que contém extensões adicionais personalizadas deve ser criado fora do CA privada da AWS.

Certificado CA subordinado_ 1_CSR Passthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 1`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

*Os pontos de distribuição da CRL serão incluídos nos certificados emitidos com este modelo apenas se a CA estiver configurada com a geração de CRL habilitada.

Certificação Subordinate_ Definição 2/V1 PathLen

Esse modelo é usado para emitir certificados CA subordinados com um comprimento de caminho igual a 2. Os certificados CA incluem uma extensão de restrições básicas críticas com o campo CA definido como TRUE para designar que o certificado pode ser utilizado para emitir certificados CA. O

uso estendido da chave não está incluído, o que impede que o certificado CA seja usado como um certificado de cliente ou de servidor TLS.

Para obter mais informações sobre caminhos de certificação, consulte Configurar restrições de comprimento no caminho de certificação.

Certificado CA subordinado _ 2/V1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 2`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

*Os pontos de distribuição da CRL serão incluídos nos certificados emitidos com este modelo apenas se a CA estiver configurada com a geração de CRL habilitada.

Certificação CA subordinada_ 2_definição de PathLen APICSRPassthrough/v1

Esse modelo estende SubordinateCacertificate_ PathLen 2/V1 para suportar valores de passagem de API e CSR.

Certificado CA subordinado _ 2_APICSR PathLen Passthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 2`

Parâmetro X509v3	Valor
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Certificado Subordinate_ 2_definição de PathLen APIPassthrough/v1

Esse modelo estende SubordinateCacertificate_ PathLen 2/V1 para oferecer suporte aos valores de passagem da API.

Certificado CA subordinado _ PathLen 2_APIPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 2`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Certificação CA subordinada_ 2_definição PathLen CSRPassthrough/v1

Este modelo é idêntico ao modelo `SubordinateCACertificate_PathLen2` com uma diferença: neste modelo, o CA privada da AWS passará extensões adicionais da solicitação de assinatura de certificado (CSR) para o certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na CSR.

> **Note**
>
> Um CSR que contém extensões adicionais personalizadas deve ser criado fora do CA privada da AWS.

Certificado CA subordinado _ 2_CSR Passthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 2`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

*Os pontos de distribuição da CRL serão incluídos nos certificados emitidos com este modelo apenas se a CA estiver configurada com a geração de CRL habilitada.

Definição de Certificado Subordinate_ PathLen 3/V1

Esse modelo é usado para emitir certificados CA subordinados com um comprimento de caminho igual a 3. Os certificados CA incluem uma extensão de restrições básicas críticas com o campo CA definido como TRUE para designar que o certificado pode ser utilizado para emitir certificados CA. O

uso estendido da chave não está incluído, o que impede que o certificado CA seja usado como um certificado de cliente ou de servidor TLS.

Para obter mais informações sobre caminhos de certificação, consulte Configurar restrições de comprimento no caminho de certificação.

Certificado CA subordinado _ 3/V1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 3`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

*Os pontos de distribuição da CRL serão incluídos nos certificados emitidos com este modelo apenas se a CA estiver configurada com a geração de CRL habilitada.

Certificação CA subordinada_ 3_definição de PathLen APICSRPassthrough/v1

Esse modelo estende SubordinateCacertificate_ PathLen 3/V1 para suportar valores de passagem de API e CSR.

Certificado CA subordinado _ 3_APICSR PathLen Passthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 3`

Parâmetro X509v3	Valor
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

SubordinateCcertificate_ 3_definição de PathLen APIPassthrough/v1

Esse modelo estende SubordinateCacertificate_ PathLen 3/V1 para oferecer suporte aos valores de passagem da API.

Certificado CA subordinado _ PathLen 3_APIPassthrough/v1

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem de API ou CSR]
Sujeito	[Passagem de API ou CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 3`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA]

Os pontos de distribuição de CRL serão incluídos no modelo somente se a CA estiver configurada com a geração de CRL habilitada.

Certificado Subordinate_ 3_CSR Passthrough/v1 definição PathLen

Este modelo é idêntico ao modelo `SubordinateCACertificate_PathLen3` com uma diferença: neste modelo, o CA privada da AWS passará extensões adicionais da solicitação de assinatura de certificado (CSR) para o certificado se as extensões não forem especificadas no modelo. As extensões especificadas no modelo sempre substituem extensões na CSR.

> **Note**
>
> Um CSR que contém extensões adicionais personalizadas deve ser criado fora do CA privada da AWS.

Certificado CA subordinado _ 3_CSR Passthrough/v1 PathLen

Parâmetro X509v3	Valor
Nome alternativo do requerente	[Passagem da CSR]
Sujeito	[Passagem da CSR]
Restrições básicas	Crítica, `CA:TRUE`, `pathlen: 3`
Identificador de chave da autoridade	[SKI do certificado de CA]
Identificador de chave do requerente	[Derivado da CSR]
Uso da chave	Crítica, assinatura digital, `keyCertSign` , sinal de CRL
Pontos de distribuição de CRL*	[Passagem da configuração de CA ou CSR]

*Os pontos de distribuição da CRL serão incluídos nos certificados emitidos com este modelo apenas se a CA estiver configurada com a geração de CRL habilitada.

Usar a API do CA privada da AWS (exemplos Java)

Você pode usar a API AWS Private Certificate Authority para interagir com o serviço de forma programática, enviando solicitações HTTP. O serviço retorna respostas HTTP. Para obter mais informações, consulte Referência da API do AWS Private Certificate Authority.

Além da API HTTP, é possível usar os AWS SDKs e as ferramentas de linha de comando para interagir com o CA privada da AWS. Isso é recomendado em vez da API HTTP. Para obter mais informações, consulte Ferramentas para a Amazon Web Services. Os tópicos a seguir mostram como usar o AWS SDK for Java para programar a API do CA privada da AWS.

O GetCertificateAuthorityCsr, GetCertificate, e DescribeCertificateAuthorityAuditReportas operações apoiam os garçons. Você pode usar agentes de espera para controlar a progressão de seu código com base na presença ou no estado de certos recursos. Para obter mais informações, consulte os tópicos a seguir, bem como os garçons AWS SDK for Java no blog do AWSdesenvolvedor.

Tópicos

- Criar e ativar uma CA raiz programaticamente
- Criar e ativar uma CA subordinada programaticamente
- CreateCertificateAuthority
- Usando CreateCertificateAuthority para dar suporte ao Active Directory
- CreateCertificateAuthorityAuditReport
- CreatePermission
- DeleteCertificateAuthority
- DeletePermission
- DeletePolicy
- DescribeCertificateAuthority
- DescribeCertificateAuthorityAuditReport
- GetCertificate
- GetCertificateAuthorityCertificate
- GetCertificateAuthorityCsr
- GetPolicy
- ImportCertificateAuthorityCertificate
- IssueCertificate

- ListCertificateAuthorities
- ListPermissions
- ListTags
- PutPolicy
- RestoreCertificateAuthority
- RevokeCertificate
- TagCertificateAuthorities
- UntagCertificateAuthority
- UpdateCertificateAuthority
- Criar CAs e certificados com nomes de requerente personalizados
- Criar certificados com extensões personalizadas

Criar e ativar uma CA raiz programaticamente

Este exemplo Java mostra como ativar uma CA raiz usando as seguintes ações da API do CA privada da 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);
    }
}
```

Criar e ativar uma CA subordinada programaticamente

Este exemplo Java mostra como ativar uma CA subordinada usando as seguintes ações da API do CA privada da 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

O exemplo de Java a seguir mostra como usar a CreateCerticateAuthorityoperação.

A operação cria uma autoridade de certificação subordinada privada (CA). Você deve especificar a configuração da CA, a configuração de revogação, o tipo de CA e um token de idempotência opcional.

A configuração da CA especifica:

- O nome do algoritmo e o tamanho da chave a serem usados para criar a chave privada da CA
- O tipo de algoritmo de assinatura que a CA usa para assinar
- Informações do assunto X.500

A configuração da CRL especifica:

- O período de expiração da CRL em dias (o período de validade da CRL)
- O bucket do Amazon S3 que conterá a CRL
- Um alias CNAME para o bucket do S3 que é incluído em certificados emitidos pela CA

Se for bem-sucedida, a função retornará o nome de recurso da Amazon (ARN) da 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);
    }
}
```

Sua saída deve ser similar à seguinte:

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

Usando CreateCertificateAuthority para dar suporte ao Active Directory

O exemplo de Java a seguir mostra como usar a CreateCerticateAuthorityoperação para criar uma CA que possa ser instalada no repositório Enterprise NTAuth do Microsoft Active Directory (AD).

A operação cria uma autoridade de certificação (CA) raiz privada usando identificadores de objetos personalizados (OIDs). Para obter mais informações e um exemplo em AWS CLI de uma operação equivalente, consulte Criar uma CA para login no Active Directory.

Se for bem-sucedida, a função retornará o nome de recurso da Amazon (ARN) da 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);
    }
}
```

Sua saída deve ser similar à seguinte:

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

CreateCertificateAuthorityAuditReport

O exemplo de Java a seguir mostra como usar a CreateCertificateAuthorityAuditReportoperação.

A operação cria um relatório de auditoria que lista toda vez que um certificado é emitido ou revogado. O relatório é salvo no bucket do Amazon S3 que você especifica na entrada. Você pode gerar um novo relatório uma vez a 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);

   }
}
```

Sua saída deve ser similar à seguinte:

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

CreatePermission

O exemplo de Java a seguir mostra como usar a CreatePermissionoperação.

A operação atribui permissões de acesso de uma CA privada a um principal de serviço da AWS designado. Os serviços podem ter permissão para criar e recuperar certificados de uma CA privada, bem como listar as permissões ativas que a CA privada concedeu. Para renovar automaticamente os certificados por meio do ACM, você deve atribuir todas as permissões possíveis (`IssueCertificate`,`GetCertificate`, e`ListPermissions`) da CA ao responsável pelo serviço do ACM (). `acm.amazonaws.com` Você pode encontrar o ARN de uma CA chamando a ListCertificateAuthoritiesfunção.

Depois que uma permissão é criada, você pode inspecioná-la com a ListPermissionsfunção ou excluí-la com a DeletePermissionfunção.

```
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

O exemplo de Java a seguir mostra como usar a DeleteCertificateAuthorityoperação.

Essa operação exclui a autoridade de certificação (CA) privada que você criou usando a CreateCertificateAuthorityoperação. A operação `DeleteCertificateAuthority` requer que você forneça um ARN para a CA a ser excluída. Você pode encontrar o ARN chamando a ListCertificateAuthoritiesoperação. Será possível excluir a CA privada imediatamente se o status for `CREATING` ou `PENDING_CERTIFICATE`. No entanto, se você já tiver importado o certificado, não será possível excluí-lo imediatamente. Primeiro, você deve desativar a CA chamando a UpdateCertificateAuthorityoperação e definindo o `Status` parâmetro como`DISABLED`. Depois, você pode usar o parâmetro `PermanentDeletionTimeInDays` na operação `DeleteCertificateAuthority` para especificar o número de dias, de 7 a 30. Durante esse período, a CA privada pode ser restaurada para o status `disabled`. Por padrão, se você não definir o parâmetro `PermanentDeletionTimeInDays`, o período de restauração será de 30 dias. Depois que esse período expirar, a CA privada será excluída permanentemente e não poderá ser restaurada. Para ter mais informações, consulte Restaurar uma CA.

Para obter um exemplo de Java que mostra como usar a RestoreCertificateAuthorityoperação, 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

O exemplo de Java a seguir mostra como usar a DeletePermissionoperação.

A operação exclui as permissões que uma CA privada delegou a um responsável pelo AWS serviço usando a CreatePermissionsoperação. Você pode encontrar o ARN de uma CA chamando a ListCertificateAuthoritiesfunção. Você pode inspecionar as permissões concedidas por uma CA chamando a ListPermissionsfunção.

```
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

O exemplo de Java a seguir mostra como usar a DeletePolicyoperação.

A operação exclui a política baseada em recurso anexada a uma CA privada. Uma política baseada em recurso é usada para permitir o compartilhamento de uma CA entre contas. Você pode encontrar o ARN de uma CA privada chamando a ListCertificateAuthoritiesação.

As ações de API relacionadas incluem PutPolicyGetPolicye.

```
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

O exemplo de Java a seguir mostra como usar a DescribeCertificateAuthorityoperação.

A operação lista informações sobre a autoridade de certificação (CA) privada. Você deve especificar o nome de recurso da Amazon (ARN) da CA privada. A saída contém o status da sua CA. Pode ser qualquer uma destas:

- `CREATING` - CA privada da AWS está criando sua autoridade de certificação privada.
- `PENDING_CERTIFICATE`: o certificado está pendente. Você deve usar a CA raiz on-premises ou subordinada para assinar a CSR da CA privada e, em seguida, importá-la para o PCA.
- `ACTIVE`: sua CA privada está ativa.
- `DISABLED`: sua CA privada foi desabilitada.
- `EXPIRED`: seu certificado de CA privada expirou.
- `FAILED`: sua CA privada não pode ser criada.
- `DELETED`: sua CA privada está dentro do período de restauração, após o qual será excluída 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

O exemplo de Java a seguir mostra como usar a DescribeCertificateAuthorityAuditReportoperação.

A operação lista informações sobre um relatório de auditoria específico que você criou ao chamar a CreateCertificateAuthorityAuditReportoperação. As informações de auditoria são criadas sempre que a autoridade de certificação (CA) da chave privada é usada. A chave privada é usada quando você emite um certificado, assina uma CRL ou revoga um 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);
    }
}
```

Sua saída deve ser similar à seguinte:

```
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

O exemplo de Java a seguir mostra como usar a GetCertificateoperação.

A operação recupera um certificado do CA privado. O ARN do certificado é retornado quando você chama a IssueCertificateoperação. Você deve especificar o ARN da CA privada e o ARN do certificado emitido ao chamar a operação `GetCertificate`. Você pode recuperar o certificado se ele estiver no estado `ISSUED`. Você pode chamar a CreateCertificateAuthorityAuditReportoperação para criar um relatório que contenha informações sobre todos os certificados emitidos e revogados pela sua CA 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);
    }
}
```

Sua saída deve ser uma cadeia de certificados semelhante à seguir para a autoridade de certificação (CA) e o certificado que você especificou.

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

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

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

GetCertificateAuthorityCertificate

O exemplo de Java a seguir mostra como usar a GetCertificateAuthorityCertificateoperação.

Essa operação recupera o certificado e a cadeia de certificados da sua autoridade de certificação (CA) privada. O certificado e a cadeia são strings codificadas em base64 no formato PEM. A cadeia não inclui o certificado de CA. Cada certificado na cadeia assina o anterior a ele.

```
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);
    }
}
```

Sua saída deve ser um certificado e uma cadeia semelhante à seguir para a autoridade de certificação (CA) que você especificou.

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

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

GetCertificateAuthorityCsr

O exemplo de Java a seguir mostra como usar a GetCertificateAuthorityCsroperação.

Essa operação recupera a solicitação de assinatura de certificado (CSR) para recuperar a sua autoridade de certificação (CA) privada. A CSR é criada quando você chama a CreateCertificateAuthorityoperação. Leve a CSR para sua infraestrutura X.509 on-premises e assine-a usando sua CA raiz ou uma subordinada. Em seguida, importe o certificado assinado de volta para o ACM PCA chamando a ImportCertificateAuthorityCertificateoperação. A CSR é retornada como uma string codificada em base64 no 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);
   }
}
```

Sua saída deve ser similar à seguir para a autoridade de certificação (CA) que você especificar. A solicitação de assinatura de certificado (CSR) é codificada no formato PEM codificado em base64. Salve-a em um arquivo local, use-a em sua infraestrutura X.509 on-premises e assine-a usando sua CA raiz ou uma subordinada.

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

GetPolicy

O exemplo de Java a seguir mostra como usar a GetPolicyoperação.

A operação recupera a política baseada em recurso anexada a uma CA privada. Uma política baseada em recurso é usada para permitir o compartilhamento de uma CA entre contas. Você pode encontrar o ARN de uma CA privada chamando a ListCertificateAuthoritiesação.

As ações de API relacionadas incluem PutPolicyDeletePolicye.

```
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

O exemplo de Java a seguir mostra como usar a ImportCertificateAuthorityCertificateoperação.

Essa operação importa o certificado CA privado assinado para o CA privada da AWS. Antes de chamar essa operação, você deve criar a autoridade de certificação privada chamando a CreateCertificateAuthorityoperação. Em seguida, você deve gerar uma solicitação de assinatura de certificado (CSR) chamando a GetCertificateAuthorityCsroperação. Leve a CSR para sua CA on-premises e use seu certificado raiz ou um certificado subordinado para assiná-la. Crie uma cadeia de certificados e copie o certificado assinado e a cadeia de certificados no seu diretório de trabalho.

```
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

O exemplo de Java a seguir mostra como usar a IssueCertificateoperação.

Essa operação usa a autoridade de certificação (CA) privada para emitir um certificado de entidade final. Essa operação retorna o nome de recurso da Amazon (ARN) do certificado. Você pode recuperar o certificado chamando GetCertificatee especificando o ARN.

> **Note**
>
> A IssueCertificateoperação exige que você especifique um modelo de certificado. Este exemplo usa o `EndEntityCertificate/V1` modelo. Para obter informações sobre todos os modelos disponíveis, consulte Noções básicas sobre modelos 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);
    }
}
```

Sua saída deve ser similar à seguinte:

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

ListCertificateAuthorities

O exemplo de Java a seguir mostra como usar a operação ListCertificateAuthorities.

Essa operação lista as autoridades de certificação (CAs) privadas que você criou usando a operação 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());
    }
}
```

Se você tiver autoridades de certificação a serem listadas, a saída deverá ser semelhante à seguinte:

```
[{
 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

O exemplo de Java a seguir mostra como usar a ListPermissionsoperação.

Essa operação lista as permissões, se houver, atribuídas por sua CA privada. As permissões, incluindo `IssueCertificateGetCertificate`, e`ListPermissions`, podem ser atribuídas a um responsável pelo AWS serviço com a CreatePermissionoperação e revogadas com a DeletePermissionsoperação.

```
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);
   }
}
```

Se a CA privada designada tiver atribuído permissões a um principal de serviço, a saída deverá ser semelhante à seguinte:

```
[{
       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

O exemplo de Java a seguir mostra como usar a ListTagsoperação.

Essa operação lista as tags, se houver, que estão associadas à sua CA privada. Tags são rótulos que você pode usar para identificar e organizar suas CAs. Cada tag consiste em uma chave e um

valor opcional. Chame a TagCertificateAuthorityoperação para adicionar uma ou mais tags à sua CA. Chame a UntagCertificateAuthorityoperação para remover as tags.

```
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);
    }
}
```

Se você tiver tags a serem listadas, sua saída deverá ser semelhante à seguinte:

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

PutPolicy

O exemplo de Java a seguir mostra como usar a PutPolicyoperação.

A operação anexa uma política baseada em recurso a uma CA privada, permitindo o compartilhamento entre contas. Quando autorizado por uma política, uma entidade principal que reside em outra conta da AWS pode emitir e renovar certificados privados de entidade final usando uma CA privada que não seja de sua propriedade. Você pode encontrar o ARN de uma CA privada chamando a ListCertificateAuthoritiesação. Para obter exemplos de políticas, consulte a orientação do CA privada da AWS sobre Políticas baseadas em recurso.

Depois que uma política é anexada a uma CA, você pode inspecioná-la com a GetPolicyação ou excluí-la com a DeletePolicyação.

```
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

O exemplo de Java a seguir mostra como usar a RestoreCertificateAuthorityoperação. A CA privada pode ser restaurada a qualquer momento durante o período de restauração. No momento, esse período pode durar de 7 a 30 dias após a data de exclusão e poderá ser definido quando a CA for excluída. Para ter mais informações, consulte Restaurar uma CA. Consulte também o exemplo 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

O exemplo de Java a seguir mostra como usar a RevokeCertificateoperação.

Essa operação revoga um certificado que você emitiu ao chamar a IssueCertificateoperação. Se você tiver habilitado uma lista de revogação de certificados (CRL) ao criar ou atualizar a sua CA privada, as informações sobre os certificados revogados serão incluídas na CRL. O CA privada da AWS grava a CRL em um bucket do Amazon S3 que você especifica. Para obter mais informações, consulte a CrlConfigurationestrutura.

```
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

O exemplo de Java a seguir mostra como usar a TagCertificateAuthorityoperação.

Essa operação adiciona uma ou mais tags à CA privada. Tags são rótulos que você pode usar para identificar e organizar os recursos da AWS. Cada tag consiste em uma chave e um valor opcional. Ao chamar essa operação, você especifica a CA privada pelo seu nome de recurso da Amazon (ARN). Você especifica a tag usando um par de chave-valor. Para identificar uma característica específica dessa CA, você pode aplicar uma tag a apenas uma CA privada. Ou, para filtrar uma relação comum entre essas CAs, você pode aplicar a mesma tag a várias CAs privadas. Para remover uma ou mais tags, use a UntagCertificateAuthorityoperação. Ligue para a ListTagsoperação para ver quais tags estão associadas à sua 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

O exemplo de Java a seguir mostra como usar a UntagCertificateAuthorityoperação.

Essa operação remove uma ou mais tags da sua CA privada. Um tag consiste em uma chave e um valor. Se você não especificar a parte do valor da tag ao chamar essa operação, a tag será removida, independentemente do valor. Se você especificar um valor, a tag será removida somente se ela estiver associada ao valor especificado. Para adicionar tags a uma CA privada, use a TagCertificateAuthorityoperação. Ligue para a ListTagsoperação para ver quais tags estão associadas à sua 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

O exemplo de Java a seguir mostra como usar a UpdateCertificateAuthorityoperação.

A operação atualiza o status ou a configuração de uma autoridade de certificação (CA) privada. A sua CA privada deve estar no estado `ACTIVE` ou `DISABLED` para que você possa atualizá-la. Você pode desativar uma CA privada que está no estado `ACTIVE` ou ativar novamente uma CA que está no 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;
        }
    }
}
```

Criar CAs e certificados com nomes de requerente personalizados

O objeto CustomAttribute permite que os administradores transmitam identificadores de objetos personalizados (OIDs) para CAs e certificados privados. OIDs personalizados podem ser usados para criar hierarquias especializadas de nomes de requerentes que refletem a estrutura e as necessidades da sua organização. Certificados personalizados devem ser criados usando um dos modelos `ApiPassthrough`. Para obter mais informações sobre os modelos, consulte Variedades de modelos. Para obter mais informações sobre o uso de atributos personalizados, consulte Emitir certificados de entidade final privada e Procedimento para criar uma CA (CLI) .

Não é possível usar `StandardAttributes` em conjunto com `CustomAttributes`. Entretanto, você pode transmitir OIDs padrão como parte de um `CustomAttributes`. Os OIDs de nome de requerente padrão estão listados na tabela a seguir:

Nome do requerente	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
Initials	2.5.4.43
Locality	2.5.4.7
Organização	2.5.4.10
OrganizationalUnit	2.5.4.11
Pseudonym	2,5.4.65
SerialNumber	2.5.4.5
Estado	2.5.4.8

Nome do requerente	ID de objeto
Surname (Sobrenome)	2.5.4.4
Cargo	2.5.4.12

Tópicos

- Criar uma CA com CustomAttribute
- Emitir um certificado com CustomAttribute

Criar uma CA com 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);
    }
}
```

Emitir um certificado com 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);
    }
}
```

Criar certificados com extensões personalizadas

O objeto CustomExtension permite que os administradores definam extensões X.509 personalizadas em certificados privados. Certificados personalizados devem ser criados usando um dos modelos ApiPassthrough. Para obter mais informações sobre os modelos, consulte Variedades de modelos. Para obter mais informações sobre o uso de extensões personalizadas, consulte Emitir certificados de entidade final privada.

Tópicos

- Ativar uma CA subordinada com a extensão NameConstraints
- Emitir um certificado com a extensão da declaração QC

Ativar uma CA subordinada com a extensão 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);
    }
}
```

Emitir um certificado com a extensão da declaração 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);
    }
}
```

Usando a API do CA privada da AWS para implementar o padrão Matter (exemplos Java)

É possível usar a API do AWS Private Certificate Authority para criar certificados que estejam em conformidade com o padrão de conectividade Matter. O Matter especifica configurações de certificado que reforçam a segurança e a consistência dos dispositivos com Internet das Coisas (IoT) em diversas plataformas de engenharia. Para obter mais informações sobre o Matter, consulte buildwithmatter.com.

Os exemplos Java nesta seção interagem com o serviço enviando solicitações HTTP. O serviço retorna respostas HTTP. Para obter mais informações, consulte Referência da API do AWS Private Certificate Authority.

Além da API HTTP, é possível usar os AWS SDKs e as ferramentas de linha de comando para interagir com o CA privada da AWS. Isso é recomendado em vez da API HTTP. Para obter mais informações, consulte Ferramentas para a Amazon Web Services. Os tópicos a seguir mostram como usar o AWS SDK for Java para programar a API do CA privada da AWS.

O GetCertificateAuthorityCsr, GetCertificate, e DescribeCertificateAuthorityAuditReportas operações apoiam os garçons. Você pode usar agentes de espera para controlar a progressão de seu código com base na presença ou no estado de certos recursos. Para obter mais informações, consulte os tópicos a seguir, bem como os garçons AWS SDK for Java no blog do AWSdesenvolvedor.

O Matter 1.2, lançado em outubro de 2023, suporta a revogação de DAC usando listas de revogação de certificados (CRLs). Para ajudá-lo a se adequar ao padrão Matter atual, ao habilitar a revogação de CRL para CAs que emitem certificados Matter, no `CrlConfiguration` objeto, na `CrlDistributionPointExtensionConfiguration` estrutura, defina como. `OmitExtension` `true`

Normalmente, as CAs incorporam o Ponto de Distribuição da CRL (CDP) nos certificados que emitem para que as partes confiáveis que realizam a validação da cadeia de certificados possam buscar a CRL e verificar o status do certificado. No Matter, o URI do CDP não é gravado em certificados. Em vez disso, os usuários buscam CDPs no Matter Distributed Compliance Ledger (DCL), o armazenamento de dados confiável da Matter. Você deve carregar o URI do CDP no Matter DCL para que ele possa ser descoberto ao validar DACs. Para obter mais informações sobre como determinar o URI do CDP, consulteDeterminando o URI do ponto de distribuição da CRL (CDP) . Para obter mais informações sobre o Matter, consulte a documentação do Matter DCL.

Tópicos

Ative uma Autoridade de Atestado de Produto (PAA)

Este exemplo de Java mostra como usar o Definição de RootCACertificate_APIPassthrough/V1 modelo para criar e instalar um certificado Matter Root CA (PAA) para certificação do produto. A extensão AuthorityKeyIdentifier (AKI) é opcional para PaaS. Para definir uma AKI, você deve gerar um valor de AKI codificado em Base64 e passá-lo por um. CustomExtension

O exemplo executa as seguintes ações da API do CA privada da AWS:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Se encontrar problemas, consulte Usar o padrão Matter na seção sobre solução 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);
    }
}
```

Ativar um Product Attestation Intermediate (PAI)

Este exemplo de Java mostra como usar o BlankSubordinateCAcertificate_ 0_definição de APIPassthrough/v1 PathLen modelo para criar e instalar um certificado Matter Subordinate CA (PAI) para certificação do produto. Você deve gerar um KeyUsage valor codificado em Base64 e passá-lo por um. CustomExtension

O exemplo executa as seguintes ações da API do CA privada da AWS:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

- GetCertificateAuthorityCertificate

Se encontrar problemas, consulte Usar o padrão Matter na seção sobre solução 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);
    }
}
```

Crie um Certificado de Atestado de Dispositivo (DAC)

Este exemplo de Java mostra como usar o modelo BlankEndEntityCertificate_ CriticalBasicConstraints _APIPassthrough/v1 para criar um Certificado de Atestado de Dispositivo Matter. Você deve gerar um KeyUsage valor codificado em Base64 e passá-lo por um. CustomExtension

O exemplo executa a seguinte ação da API do CA privada da AWS:

- IssueCertificate

Se encontrar problemas, consulte Usar o padrão Matter na seção sobre solução 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);
    }
}
```

Ative uma CA raiz para certificados operacionais de nós (NOC).

Este exemplo de Java mostra como usar o Definição de RootCACertificate_APIPassthrough/V1 modelo para criar e instalar um certificado Matter Root CA para emitir NOCs. A extensão AuthorityKeyIdentifier (AKI) é opcional para certificados NOC Root CA. Para definir uma AKI, você deve gerar um valor de AKI codificado em Base64 e passá-lo por um. CustomExtension

O exemplo executa as seguintes ações da API do CA privada da AWS:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Se encontrar problemas, consulte Usar o padrão Matter na seção sobre solução 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);
    }
}
```

Ativar uma CA subordinada para certificados operacionais de nós (NOC)

Este exemplo de Java mostra como usar o BlankSubordinateCAcertificate_ 0_definição de APIPassthrough/v1 PathLen modelo para emitir e instalar um certificado Matter Subordinate CA para emitir NOCs. Você deve gerar um KeyUsage valor codificado em Base64 e passá-lo por um. CustomExtension

O exemplo executa as seguintes ações da API do CA privada da AWS:

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

Se ocorrerem problemas, consulte a Usar o padrão Matter seção Solução 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);
    }
}
```

Crie um certificado operacional de nó (NOC)

Este exemplo de Java mostra como usar o modelo BlankEndEntityCertificate_ CriticalBasicConstraints _APIPassthrough/v1 para criar um certificado operacional Matter Node. Você deve gerar um KeyUsage valor codificado em Base64 e passá-lo por um. CustomExtension

O exemplo executa a seguinte ação da API do CA privada da AWS:

- IssueCertificate

Se encontrar problemas, consulte Usar o padrão Matter na seção sobre solução 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);
   }
}
```

Usando a CA privada da AWS API para implementar o padrão de carteira de motorista móvel (mDL) (exemplos em Java)

Você pode usar a AWS Private Certificate Authority API para criar certificados que estejam em conformidade com o padrão ISO/IEC para carteira de motorista móvel (mDL). Esse padrão estabelece especificações de interface para a implementação de uma carteira de habilitação em associação com um dispositivo móvel, incluindo configurações de certificado.

Os exemplos Java nesta seção interagem com o serviço enviando solicitações HTTP. O serviço retorna respostas HTTP. Para obter mais informações, consulte a AWS Private Certificate Authority Referência da API.

Além da API HTTP, você também pode usar os AWS SDKs e as AWS CLI ferramentas para gerenciarCA privada da AWS. Recomendamos usar o SDK ou AWS CLI a API HTTP. Para obter mais informações, consulte Ferramentas para a Amazon Web Services. Os tópicos a seguir mostram como usar o AWS SDK for Java para programar a API do CA privada da AWS.

O `GetCertificateAuthorityCsr`, `GetCertificate`, e `DescribeCertificateAuthorityAuditReport`as operações apoiam os garçons. Você pode usar agentes de espera para controlar a progressão de seu código com base na presença ou no estado de certos recursos. Para obter mais informações, consulte os tópicos a seguir e Garçons no AWS SDK for Java no blog do desenvolvedor. AWS

Tópicos

- Ativar um certificado de autoridade certificadora da autoridade emissora (IACA)
- Crie um certificado de signatário do documento

Ativar um certificado de autoridade certificadora da autoridade emissora (IACA)

Este exemplo de Java mostra como usar o BlankRootCAcertificate_ 0_definição de APIPassthrough/v1 PathLen modelo para criar e instalar um certificado de autoridade certificadora de autoridade emissora (IACA) compatível com o padrão ISO/IEC mDL. Você deve gerar valores codificados em

base64 para`KeyUsage`,, e `IssuerAlternativeName`CRLDistributionPoint, e transmiti-los. `CustomExtensions`

> **Note**
>
> O certificado de link IACA estabelece um caminho confiável do certificado raiz IACA antigo para o novo certificado raiz IACA. A autoridade emissora pode gerar e distribuir um certificado de link IACA durante o processo de rechave da IACA. Você não pode emitir um certificado de link IACA usando um certificado raiz IACA com `pathLen=0` set.

O exemplo executa as seguintes ações da API do CA privada da 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);
    }
}
```

Crie um certificado de signatário do documento

Este exemplo de Java mostra como usar o modelo BlankEndEntityCertificate_APIPassthrough/v1 para criar um certificado de assinante de documentos compatível com o padrão ISO/IEC mDL. Você deve gerar valores codificados em base64 para `KeyUsageIssuerAlternativeName`, e `CRLDistributionPoint` e transmiti-los. `CustomExtensions`

O exemplo executa a seguinte ação da API do CA privada da 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 CA privados externamente assinados

Se a raiz de confiança da sua hierarquia de CA privada precisar ser uma CA fora do CA privada da AWS, crie e autoassine a sua própria CA raiz. Como alternativa, é possível obter um certificado CA privado assinado por uma CA privada externa operada por sua organização. Seja qual for a origem, você pode usar essa CA obtida externamente para assinar um certificado CA subordinado privado que é gerenciado pelo CA privada da AWS.

> **Note**
>
> Os procedimentos para criar ou obter um provedor de serviços de confiança externa CA externa estão fora do escopo deste guia.

O uso de uma CA primária externa com o CA privada da AWS permite que você imponha restrições de nomes de CA, conforme definido na seção Restrições de nomes da RFC 5280. Restrições de nome fornecem uma maneira para os administradores de CA restringirem nomes de requerente em certificados.

Se você planeja assinar um certificado de CA subordinado privado com uma CA externa, existem três tarefas que devem ser concluídas antes de ter uma CA ativa no CA privada da AWS:

1. Gere uma solicitação de assinatura de certificado (CSR).
2. Envie a CSR para sua autoridade de assinatura externa e retorne com um certificado assinado e uma cadeia de certificados.
3. Instale um certificado assinado no CA privada da AWS.

Os procedimentos a seguir descrevem como concluir essas tarefas usando o AWS Management Console ou a AWS CLI.

Para obter e instalar um certificado de CA externamente assinado (console)

1. (Opcional) Se você ainda não estiver na página de detalhes da CA, abra o console do CA privada da AWS em https://console.aws.amazon.com/acm-pca/home. Na página Autoridades de certificação privadas, escolha uma CA subordinada com o status Certificado pendente, Ativa, Desabilitada ou Expirada.

2. Escolha Ações, Instalar certificado de CA para abrir a página Instalar certificado de CA subordinada.
3. Na página Instalar certificado de CA subordinada, em Selecionar tipo de CA, escolha CA privada externa.
4. Em CSR para essa CA, o console exibe o texto ASCII codificado em Base64 da CSR. É possível copiar o texto usando o botão Copiar ou escolher Exportar CSR para um arquivo e salvá-lo localmente.

   > **Note**
   >
   > O formato exato do texto da CSR deve ser preservado ao copiar e colar.

5. Se você não puder executar imediatamente as etapas offline para obter um certificado assinado da sua autoridade de assinatura externa, feche a página e retorne a ela depois de possuir um certificado assinado e uma cadeia de certificados.

   Caso contrário, se estiver pronto, execute uma das seguintes ações:

   - Cole o texto ASCII codificado em Base64 do corpo do certificado e da cadeia de certificados nas respectivas caixas de texto.
   - Escolha Upload para carregar o corpo do certificado e a cadeia de certificados dos arquivos locais nas respectivas caixas de texto.
6. Escolha Confirmar e instalar.

Para obter e instalar um certificado de CA externamente assinado (CLI)

1. Use o get-certificate-authority-csrcomando para recuperar a solicitação de assinatura de certificado (CSR) para sua CA privada. Se você deseja enviar a CSR para sua exibição, use a `--output text` opção para eliminar os caracteres CR/LF do final de cada linha. Para enviar a CSR para um arquivo, use a opção de redirecionamento (**>**) seguido por um nome de arquivo.

```
$ 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
```

Depois de salvar uma CSR como arquivo local, você pode inspecioná-la usando o seguinte comando OpenSSL:

```
openssl req -in path_to_CSR_file -text -noout
```

Esse comando gera uma saída semelhante à seguinte. Observe que a extensão da CA é TRUE, o que indica que a CSR é para um certificado CA.

```
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. Envie a CSR à sua autoridade de assinatura externa e obtenha arquivos contendo o certificado assinado codificado em PEM Base64 e a cadeia de certificados.
3. Use o import-certificate-authority-certificatecomando para importar o arquivo de certificado CA privado e o arquivo em cadeia paraCA privada da AWS.

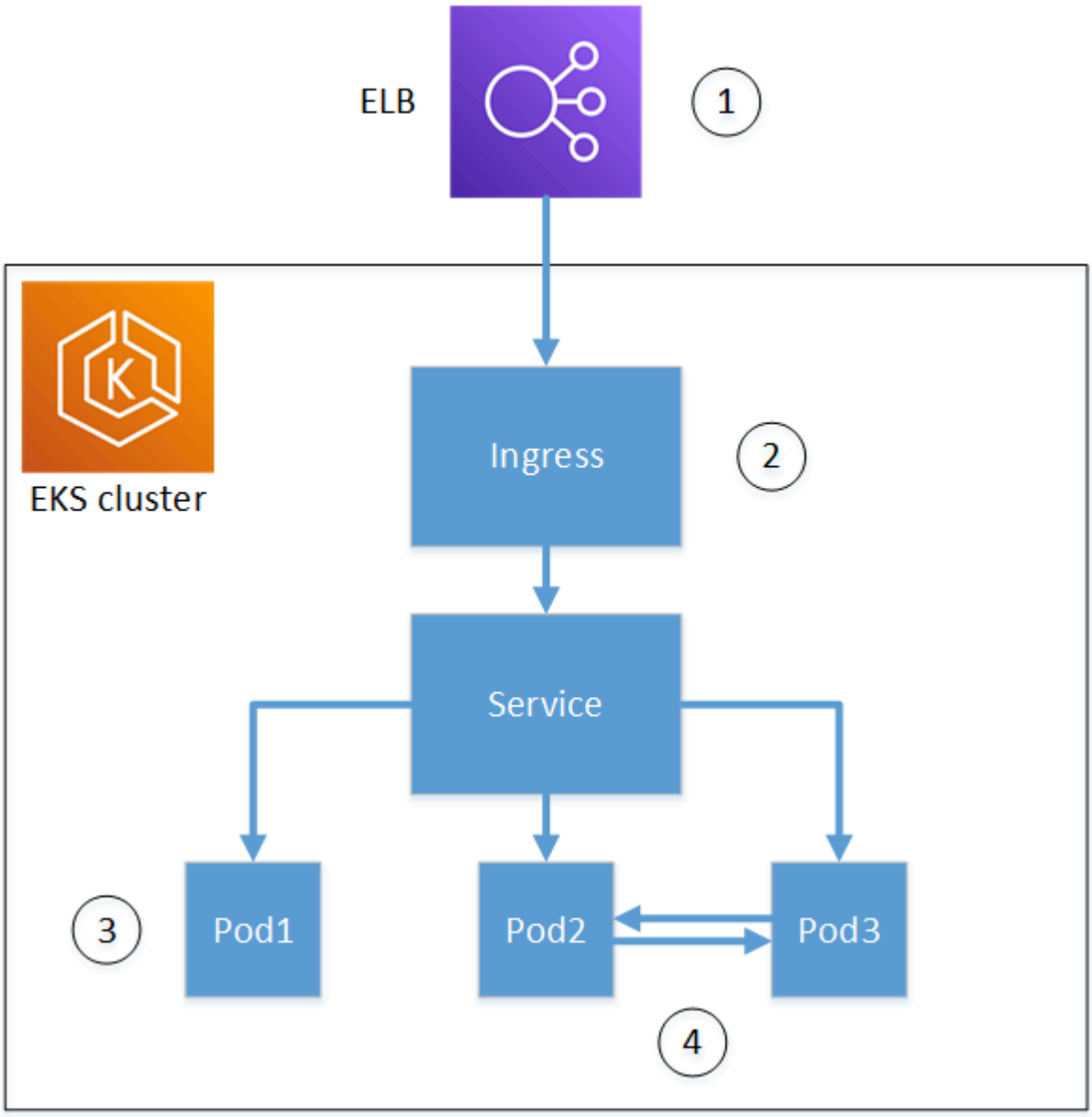
```
$ aws acm-pca import-certificate-authority-certificate \
--certificate-authority-arn arn:aws:acm-pca:region:account:\
certificate-authority/12345678-1234-1234-1234-123456789012 \
--certificate file://C:\example_ca_cert.pem \
--certificate-chain file://C:\example_ca_cert_chain.pem
```

Proteger o Kubernetes com o CA privada da AWS

Contêineres e aplicações Kubernetes usam certificados digitais para fornecer autenticação e criptografia seguras via TLS. Uma solução amplamente adotada para gerenciar o ciclo de vida de certificados TLS no Kubernetes é o cert-manager, um complemento do Kubernetes que solicita certificados, os distribui para contêineres Kubernetes e automatiza a renovação desses certificados.

CA privada da AWSfornece um plug-in de código aberto para cert-manager, aws-privateca-issuer, para usuários do cert-manager que desejam configurar uma CA sem armazenar chaves privadas no cluster. Usuários com exigências normativas para controlar o acesso e auditar operações de CA podem usar essa solução para melhorar a capacidade de auditoria e apoiar a conformidade. É possível usar o plug-in AWS Private CA Issuer com o Amazon Elastic Kubernetes Service (Amazon EKS), um Kubernetes autogerenciado na AWS, ou no Kubernetes on-premises.

O diagrama abaixo mostra algumas das opções disponíveis para usar o TLS em um cluster do Amazon EKS. Esse cluster de exemplo, contendo vários recursos, está atrás de um balanceador de carga. Os números identificam possíveis pontos finais para comunicações protegidas por TLS, incluindo o balanceador de carga externo, o controlador de entrada, um pod individual dentro de um serviço e um par de pods que se comunicam com segurança entre si.



1. Término no balanceador de carga.

   O Elastic Load Balancing (ELB) é um serviço integrado do AWS Certificate Manager, o que significa que você pode provisionar o ACM com uma CA privada, assinar um certificado com ela e instalá-la usando o console do ELB. Essa solução fornece comunicação criptografada entre um cliente remoto e o balanceador de carga. Os dados são transmitidos sem criptografia ao cluster do EKS. Como alternativa, é possível fornecer um certificado privado a um balanceador de carga que não seja da AWS para encerrar o TLS.
2. Término no controlador de entrada Kubernetes.

O controlador de entrada reside no cluster do EKS como um balanceador de carga e roteador nativos. Se você instalou o cert-manager e aws-privateca-issuerprovisionou o cluster com uma CA privada, o Kubernetes pode instalar um certificado TLS assinado no controlador, permitindo que ele sirva como ponto final do cluster para comunicações externas. As comunicações entre o balanceador de carga e o controlador de entrada são criptografadas e, após a entrada, os dados são transmitidos sem criptografia aos recursos do cluster.

3. Término em um pod.

   Cada pod é um grupo de um ou mais contêineres que compartilham recursos de armazenamento e rede. Se você instalou o cert-manager e provisionou o cluster com uma CA privada aws-privateca-issuer, o Kubernetes pode instalar certificados TLS assinados em pods conforme necessário. Uma conexão TLS terminando em um pod não está disponível por padrão para outros pods no cluster.

4. Comunicações seguras entre pods.

   Você também pode provisionar vários pods com certificados que permitem a comunicação deles entre si. Os cenários a seguir são possíveis:

   - Provisionamento com certificados autoassinados e gerados pelo Kubernetes. Isso protege as comunicações entre os pods, mas os certificados autoassinados não atendem aos requisitos da HIPAA ou do FIPS.
   - Provisionamento com certificados assinados por uma CA privada. Como nos números 2 e 3 acima, isso requer a instalação do cert-manager e aws-privateca-issuero provisionamento do cluster com uma CA privada. O Kubernetes pode então instalar certificados TLS assinados nos pods conforme necessário.

Uso entre contas do cert-manager

Administradores com acesso entre contas a uma CA podem usar o cert-manager para provisionar um cluster do Kubernetes. Para ter mais informações, consulte Práticas de segurança recomendadas para acesso entre contas a CAs privadas.

> **Note**
>
> Somente determinados modelos de certificados do CA privada da AWS podem ser usados em cenários entre contas. Consulte the section called “Modelos de certificados com suporte ” para obter uma lista de modelos disponíveis.

Modelos de certificados com suporte

A tabela a seguir lista os modelos do CA privada da AWS que podem ser usados com o cert-manager para provisionar um cluster do Kubernetes.

Modelos compatíveis com o Kubernetes	Suporte para uso entre contas
BlankEndEntityCertificateDefinição de _CSRPassthrough/v1	
CodeSigningCertificateDefinição de /V1	
EndEntityCertificateDefinição de /V1	✓
EndEntityClientAuthCertificateDefinição de /V1	✓
EndEntityServerAuthCertificateDefinição de /V1	✓
Definição de SigningCertificate OCSP/V1	

Soluções de exemplo

As soluções de integração a seguir mostram como configurar o acesso ao CA privada da AWS em um cluster do Amazon EKS.

- Clusters Kubernetes habilitados para TLS com o CA privada da AWS e o Amazon EKS
- Configurando a criptografia end-to-end TLS no Amazon EKS com o novo AWS Load Balancer Controller

CA privada da AWS Connector para Active Directory

O que é o AWS Private CA Connector para Active Directory

O AWS Private CA pode emitir e gerenciar certificados exigidos pelo AWS Managed Microsoft AD. Usando o CA privada da AWS Connector para Active Directory (Connector para AD), você pode substituir CAs corporativas on-premises ou outras CAs terceirizadas por uma CA privada gerenciada de sua propriedade, fornecendo inscrição de certificados para usuários, grupos e máquinas gerenciados pelo AD.

É possível usar o Connector para AD com o AWS Managed Microsoft AD para eliminar a infraestrutura on-premises, migrando sua infraestrutura do AD e de chave pública para a nuvem. Para clientes que desejam usar o AWS Private CA com o AD on-premises, esse recurso também se integra ao AWS Managed Microsoft AD Connector.

Tópicos

Você é um usuário iniciante do Connector para AD?

Se estiver usando o Connector para AD pela primeira vez, recomendamos que você leia as seguintes seções para começar:

- O que CA privada da AWSé
- O que é AWS Directory Service?

Acessar o Connector para AD

Você pode acessar o Connector for AD por meio do console e das APIs. AWS CLI Você pode obter acesso ao conector no console a partir do AWS Private CA console, do seu AWS Directory Service console ou pesquisando por Connector for AD na barra de AWS Management Console pesquisa.

Preços do Connector para AD

O Connector para AD é oferecido como um recurso do CA privada da AWS sem custo adicional. Você paga apenas pelas autoridades de certificação privadas e pelos certificados emitidos por meio delas.

Para obter as informações de preço do CA privada da AWS mais recentes, a Preços do AWS Private Certificate Authority. Você também pode usar a Calculadora de preços da AWS para estimar os custos.

Introdução ao AWS Private CA Connector para Active Directory

Com o AWS Private CA Connector para Active Directory, você pode emitir certificados da sua CA privada para seus objetos do Active Directory para autenticação e criptografia. Quando você cria um conector, o AWS Private Certificate Authority cria um endpoint na sua VPC, para que seus objetos de diretório solicitem certificados.

Para emitir certificados, você cria um conector e modelos compatíveis com AD para o conector. Ao criar um modelo, você pode definir permissões de inscrição para seus grupos do AD.

Tópicos

Pré-requisitos do Connector para AD

O seguinte é necessário para o Connector para AD:

Para criar um conector do serviço Connector para AD, você precisa configurar uma AWS Private Certificate Authority (CA) e um diretório. Em seguida, precisa compartilhar a CA privada e o diretório com o serviço Connector para AD. Você também precisa definir corretamente grupos de segurança e políticas do IAM para criar um conector.

CA privada da AWS

Configure uma CA privada da AWS para emitir certificados para seus objetos de diretório. Para ter mais informações, consulte Administração de CAs privadas.

A CA privada da AWS deve estar no estado `Active` para criar um conector do serviço Connector para AD. O nome do requerente da CA privada deve incluir um nome comum. A criação do conector falhará se você tentar criar um conector usando uma CA privada sem um nome comum.

Active Directory

Além de uma CA privada da AWS, você precisa de um Active Directory em uma nuvem privada virtual (VPC). O Connector para AD oferece suporte aos seguintes tipos de diretório oferecidos pelo AWS Directory Service:

- AWSMicrosoft Active Directory gerenciado: Com AWS Directory Service você pode executar o Microsoft Active Directory (AD) como um serviço gerenciado. AWS Directory Service for Microsoft Active Directorytambém conhecido comoAWS Managed Microsoft AD, é alimentado pelo Windows Server 2019. Com o AWS Managed Microsoft AD, você pode executar workloads com reconhecimento de diretório na Nuvem AWS, incluindo Microsoft SharePoint e aplicações .NET personalizadas e baseadas em SQL Server.
- Active Directory Connector: o AD Connector é um gateway de diretório que pode redirecionar solicitações de diretório para seu Microsoft Active Directory on-premises sem armazenar nenhuma informação em cache na nuvem. O AD Connector oferece suporte à conexão com um domínio hospedado no Amazon EC2.

> **Note**
>
> Não há suporte para a inscrição de controladores de domínio ao usar o Connector para AD com o AWS Managed Microsoft AD.

Conta de serviço

Ao usar o Directory Service AD Connector, você precisa delegar permissões adicionais à conta de serviço. Defina uma lista de controle de acesso (ACL) na conta de serviço para poder:

- Adicionar e remover um nome de entidade principal do serviço (SPN) a si mesmo

- Criar e atualizar autoridades de certificação nos seguintes contêineres:

```
#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
```

- Crie e atualize um objeto da Autoridade de AuthCertificates Certificação (CA) do NT. Observação: se o objeto NT AuthCertificates CA existir, você deverá delegar permissões para ele. Se o objeto não existir, você deverá delegar a capacidade de criar objetos secundários no contêiner de serviços de chave pública.

```
#objects
CN=NTAuthCertificates,CN=Public Key Services,CN=Services,CN=Configuration
```

> **Note**
>
> Se estiver usando o AWS Managed Microsoft AD, as permissões adicionais serão delegadas automaticamente quando você autorizar o serviço Connector para AD com seu diretório. É possível ignorar essa etapa de pré-requisito.

Você pode usar esse PowerShell script para delegar as permissões adicionais. Isso criará o objeto da autoridade de AuthCertifiates certificação NT. Substitua "myconnectoraccount" pelo nome da conta de serviço.

```
$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 do IAM

atributPara criar um conector do Connector para AD, você precisa de uma política do IAM que permita criar recursos de conectores, compartilhar a CA privada com o serviço Connector para AD e autorizar esse serviço com seu diretório.

Este é um exemplo de política gerenciada pelo usuário:

```
{
    "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/*"
        }
    ]
}
```

O Connector para AD requer permissões adicionais do AWS RAM, tanto para uso no console quanto na linha de comando.

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

Compartilhar a CA privada da AWS com o Connector para AD

Você precisará compartilhar sua AWS Private CA com o serviço de conectores usando o compartilhamento de entidade principal do serviço AWS Resource Access Manager.

Quando você cria um conector no AWS console, o compartilhamento de recursos é criado automaticamente para você.

Ao criar um compartilhamento de recursos usando o AWS CLI, você usará o comando create-resource-share do AWS RAM.

O comando a seguir cria um compartilhamento de recurso:

```
$  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
```

O responsável pelo serviço que liga CreateConnector tem permissões de emissão de certificados no PCA. Para evitar que as entidades principais de serviços que usam o Connector para AD tenham acesso geral aos recursos da sua CA privada da AWS, restrinja suas permissões usando `CalledVia`.

Autorizar o Connector para AD no seu diretório

Você autoriza o serviço Connector para AD no seu diretório para ele possa se comunicar com esse diretório. Para autorizar o serviço Connector para AD, crie uma inscrição de diretório. Para obter mais informações sobre como criar uma inscrição de diretório, consulte Gerenciar inscrições de diretório

Grupos de segurança

A comunicação entre a VPC e o conector do Connector para AD é feita por meio do AWS PrivateLink, o que requer grupos de segurança com regras de entrada que abram as portas 443 TCP e UDP na sua VPC. Esse grupo de segurança será solicitado quando você criar um conector. É

possível especificar a origem como personalizada e selecionar o bloco CIDR da sua VPC. É possível optar por restringir ainda mais isso (ou seja, IP, CIDR e ID do grupo de segurança).

Criar um conector

Para obter instruções, consulte o procedimento Criar um conector.

Configurar políticas do AD

O Connector para AD não consegue visualizar ou gerenciar a configuração do objeto de política de grupo (GPO) do cliente. O GPO controla o roteamento das solicitações do AD à CA privada da AWS do cliente ou a outros servidores de autenticação ou fornecimento automático de certificados. Uma configuração de GPO inválida pode fazer com que suas solicitações sejam roteadas incorretamente. Cabe aos clientes configurar e testar a configuração do Connector para AD.

Políticas de grupo são associadas a um conector, e você pode optar por criar vários conectores para um único AD. Cabe a você gerenciar o controle de acesso a cada conector se as suas configurações de política de grupo forem diferentes.

A segurança das chamadas de plano de dados depende do Kerberos e da sua configuração de VPC. Qualquer pessoa com acesso à VPC poderá fazer chamadas de plano de dados, desde que esteja autenticada no AD correspondente. Isso existe fora dos limites AWSAuth e o gerenciamento da autorização e autenticação depende de você, o cliente.

No Active Directory, siga as etapas abaixo para criar um GPO que aponte para o URI gerado quando você criou um conector. Essa etapa é necessária para usar o Connector para AD a partir do console ou da linha de comando.

Configure GPOs.

1. Abra o Gerenciador de Servidores no DC
2. Acesse Ferramentas e escolha Gerenciamento de políticas de grupo no canto superior direito do console.
3. Acesse Floresta > Domínios. Selecione seu nome de domínio e clique com o botão direito no domínio. Selecione Criar um GPO neste domínio, vinculá-lo aqui... e insira `PCA GP0` para o nome.
4. O GPO recém-criado será listado abaixo do seu nome de domínio.

5. Escolha GPO do PCA e selecione Editar. Se uma caixa de diálogo for aberta com a mensagem de alerta Este é um link e as alterações serão propagadas globalmente, confirme a mensagem para continuar. O Editor de gerenciamento de políticas de grupo será aberto.
6. No Editor de gerenciamento de políticas de grupo, cesse Configuração do computador > Políticas > Configurações do Windows > Configurações de segurança > Políticas de chave pública (escolha a pasta).
7. Selecione o tipo de objeto e escolha Cliente de serviços de certificado - Política de inscrição de certificado
8. Nas opções, altere o Modelo de configuração para Habilitado.
9. Confirme se a Política de inscrição do Active Directory está marcada e Habilitada. Escolha Adicionar.
10. A janela Servidor de políticas de inscrição de certificado deve ser aberta.
11. Insira o endpoint do servidor de política de inscrição de certificados que foi gerado quando você criou seu conector no campo Inserir URI da política do servidor de inscrição.
12. Deixe Tipo de autenticação como Windows integrado.
13. Escolha Validar. Depois que a validação for bem-sucedida, selecione Adicionar. A caixa de diálogo é fechada.
14. Volte para Cliente de serviços de certificado - Política de inscrição de certificado e marque a caixa ao lado do conector recém-criado para garantir que ele seja a política de inscrição padrão
15. Escolha Política de inscrição do Active Directory e selecione Remover.
16. Na caixa de diálogo de confirmação, escolha Sim para excluir a autenticação baseada em LDAP.
17. Escolha Aplicar e OK na janela Cliente de serviços de certificados > Política de inscrição de certificado e feche essa janela.
18. Acesse a pasta Política de chave pública e escolha Cliente de serviços de certificado - Inscrição automática.
19. Altere a opção Modelo de configuração para Habilitado.
20. Confirme se as opções Renovar certificados expirados e Atualizar certificados estão ambas marcadas. Deixe as outras configurações como estão.
21. Escolha Aplicar, depois OK e feche a caixa de diálogo.

Em seguida, configure as políticas de chave pública para configuração de usuários. Acesse Configuração de usuários > Políticas > Configurações do Windows > Configurações de segurança >

Políticas de chave pública. Siga os procedimentos descritos da etapa 6 à etapa 21 para configurar as Políticas de chave pública para configuração de usuários.

Depois que você concluir a configuração dos GPOs e das políticas de chave pública, os objetos no domínio solicitarão certificados do CA privada da AWS Connector para AD e receberão certificados emitidos pelo CA privada da AWS.

Criar um modelo

Para obter instruções, consulte o procedimento Criar um modelo de conector.

Gerenciar permissões de grupos do AD

Para obter instruções, consulte o procedimento Gerenciar grupos e permissões do AD para modelos.

Procedimentos do CA privada da AWS Connector para Active Directory

Os procedimentos nesta seção descrevem como criar conectores, configurar modelos e fazer a integração com o CA privada da AWS e o Active Directory. É possível realizar essas operações no console do CA privada da AWS Connector para AD, usando a seção Connector para AD da AWS CLI ou usando a API do CA privada da AWS Connector para AD.

> **Note**
>
> Embora o CA privada da AWS Connector para AD esteja estreitamente integrado ao CA privada da AWS, os dois serviços têm APIs separadas. Para obter mais informações, consulte a Referência da AWS Private Certificate AuthorityAPI e a Referência da API do CA privada da AWS Connector for Active Directory.

Procedimentos

- Criar um conector
- Criar um modelo de conector
- Listar conectores para o Active Directory
- Listar modelos de conectores
- Visualizar detalhes do conector

- Visualizar detalhes de um modelo de conector
- Gerenciar inscrições de diretório
- Gerenciar grupos e permissões do AD para modelos
- Configurar o nome da entidade principal do serviço
- Marcar recursos do Connector para AD

Criar um conector

Use os procedimentos a seguir para visualizar os detalhes de configuração de um conector no console, na linha de comando ou na API do AWS Private CA Connector para Active Directory.

Criar um conector (console)

Conclua os procedimentos a seguir para criar e configurar um conector usando o console do AWS.

Tarefas

- Abrir console
- Abra Criar um conector
- Escolha ou crie um diretório
- Escolher uma CA privada
- Configurar a marcação
- Examinar e criar

Abrir console

Faça login na sua conta da AWS e abra o console do AWS Private CA Connector para Active Directory em **https://console.aws.amazon.com/pca-connector-ad/home**.

Abra Criar um conector

Na página inicial do serviço ou na página Connector para Active Directory, escolha Criar conector.

Escolha ou crie um diretório

Na página Criar conector de CA privado para o Active Directory, forneça informações na seção Active Directory.

- Em Selecione seu tipo do Active Directory, escolha um dos dois tipos disponíveis:
  - AWS Directory Service for Microsoft Active Directory— Especifica um Active Directory gerenciado porAWS Directory Service.
  - Active Directory local com AWS AD Connector: usa o AD Connector para acessar um Active Directory que você hospeda on-premises.
- Em Selecionar seu diretório, escolha seu diretório na lista.

  Como alternativa, você pode escolher Criar diretório, o que abre o console do AWS Directory Service em uma nova janela. Ao terminar de criar um novo diretório, retorne ao console do AWS Private CA Connector para Active Directory e atualize a lista de diretórios. Seu novo diretório estará disponível para seleção.

  > **Note**
  >
  > Ao criar um diretório, observe que o Connector para AD oferece suporte somente aos seguintes tipos de diretório oferecidos no console do AWS Directory Service:
  >
  > - AWS Managed Microsoft AD
  > - AD Connector

- Em Selecionar grupos de segurança para o endpoint de VPC, escolha um grupo de segurança na lista.

  Como alternativa, você pode escolher Criar grupo de segurança, que abre o console do Amazon EC2 na página Criar grupo de segurança em uma nova janela. Ao terminar de criar um grupo de segurança, retorne ao console do AWS Private CA Connector para Active Directory e atualize a lista de grupos de segurança. Seu novo grupo de segurança estará disponível para seleção.

Escolher uma CA privada

Na página Autoridade de certificação privada, escolha sua CA privada na lista.

Como alternativa, você pode escolher Criar CA privada, que abre o console do CA privada da AWS da página Autoridades de certificação privadas em uma nova janela. Ao terminar de criar uma CA, retorne ao console do AWS Private CA Connector para Active Directory e atualize a lista de CAs. Sua nova CA estará disponível para seleção.

Configurar a marcação

No painel Tags – opcional, você pode aplicar e remover metadados no seu recurso do AD. Etiquetas são pares de strings de chave/valor em que a chave deve ser exclusiva do recurso e o valor é opcional. O painel exibe todas as etiquetas existentes para o recurso em uma tabela. As ações a seguir são compatíveis.

- Escolha Gerenciar etiquetas para abrir a página Gerenciar etiquetas.
- Escolha Adicionar nova etiqueta para criar uma etiqueta. Preencha o campo Chave e, opcionalmente, Valor. Escolha Salvar alterações para aplicar a etiqueta.
- Escolha o botão Remover ao lado de uma etiqueta para marcá-la para exclusão e escolha Salvar alterações para confirmar.

Examinar e criar

Depois de fornecer as informações necessárias e analisar suas opções, escolha Criar conector. Isso abre a página de detalhes dos Conectores para Active Directory, onde você pode ver o progresso do seu conector à medida que ele é criado.

Depois que o processo de criação de um conector for concluído, atribua a ele um nome de entidade principal de serviço.

Criar um conector para o Active Directory (AWS CLI)

Para criar um conector para o Active Directory com a CLI, use o comando create-connector na seção AWS Private CA Connector para Active Directory da AWS CLI.

Criar um conector para o Active Directory (API)

Para criar um conector para o Active Directory com a API, use a CreateConnectoração na API AWS Private CA Connector for Active Directory.

Criar um modelo de conector

Criar um modelo de conector (console)

Conclua os procedimentos a seguir para criar e configurar um modelo de conector usando o console do AWS.

Tarefas

- Abrir console
- Escolher conector
- Localizar seção de modelo
- Método de criação de modelo
- Configurações do modelo
- Definir configurações de certificado
- Definir configurações de processamento e registro de solicitações
- Configurar extensões de uso de chaves
- Atribuir políticas de aplicação
- Configurar políticas personalizadas de aplicações
- Definir configurações de criptografia
- Configurar grupos e permissões
- Configurar modelos de substituição
- Configurar a marcação
- Analisar e criar

Abrir console

Faça login na sua conta da AWS e abra o console do AWS Private CA Connector para Active Directory em **`https://console.aws.amazon.com/pca-connector-ad/home`**.

Escolher conector

Escolha um conector na lista Conectores para Active Directory e escolha Visualizar detalhes.

Localizar seção de modelo

Na página de detalhes do conector, encontre a seção Modelos e escolha Criar modelo.

Método de criação de modelo

Na página Criar modelo, na seção Método de criação de modelo, escolha uma das opções de método.

- Começar com um modelo predefinido (padrão): escolha em uma lista de modelos predefinidos para aplicações do AD:
  - Assinatura de código

- Computador
- Autenticação do controlador de domínio
- Agente de recuperação do EFS
- Agente de inscrição
- Agente de inscrição (computador)
- IPSec
- Autenticação Kerberos
- Servidor RAS e IAS
- Logon com cartão inteligente
- Assinatura de lista de confiança
- Assinatura de usuário
- Autenticação de estação de trabalho

- Comece com um modelo existente que você criou: escolha em uma lista de modelos personalizados que você criou anteriormente.
- Comece com um modelo em branco: escolha essa opção para começar a criar um modelo completamente novo.

Configurações do modelo

Na seção Configurações do modelo, forneça as seguintes informações:

- Nome do modelo: o nome do modelo.
- Versão do esquema do modelo: a versão do esquema do modelo. A versão do esquema afeta a disponibilidade das opções de modelo da seguinte forma:

  Versão 2 do esquema

  - Oferece suporte à compatibilidade de clientes Windows XP/Windows Server 2003 e versões superiores.
  - Oferece suporta somente a provedores de serviços criptográficos antigos.

  Versão 3 do esquema

  - Oferece suporte à compatibilidade de clientes Windows Vista/Windows Server 2008 e versões superiores.
  - Permite que o solicitante renove com a chave existente.

- Oferece suporte somente a provedores de armazenamento de chaves.

Versão 4 do esquema

- Oferece suporte à compatibilidade de clientes Windows 8/Windows Server 2012 e versões superiores.
- Permite que o solicitante renove com a chave existente.
- Oferece suporte a provedores de serviços criptográficos antigos e provedores de armazenamento de chaves.

- Compatibilidade do cliente: o nível mínimo do sistema operacional compatível com o modelo. Escolha uma das opções listadas:
  - Windows XP/Windows Server 2003
  - Windows Vista/Windows Server 2008
  - Windows 7 / Windows Server 2008 R2
  - Windows 8 e posterior/Windows Server 2012
  - Windows 8 e posterior/Windows Server 2012 R2
  - Windows 8 e posterior/ Windows Server 2016 e posterior

Definir configurações de certificado

Na seção Configurações do certificado, defina as seguintes configurações para certificados com base nesse modelo.

- Tipo de certificado: especifique se deseja criar certificados de usuário ou computador.
- Inscrição automática: escolha se deseja ativar a inscrição automática para certificados com base nesse modelo.
- Período de validade: especifique um período de validade do certificado como um valor inteiro de horas, dias, semanas, meses ou anos. O valor mínimo é de 2 horas.
- Período de renovação: especifique um período de renovação do certificado como um valor inteiro de horas, dias, semanas, meses ou anos. O período de renovação não deve exceder 75% do período de validade.
- Nome do requerente: escolha uma ou mais opções a serem incluídas no nome do requerente com base nas informações contidas no Active Directory.

> **Note**
>
> Pelo menos um nome de requerente ou opção de nome alternativo de requerente deve ser especificado.

  - Nome comum
  - DNS como nome comum
  - Caminho do diretório
  - E-mail
- Nome alternativo do requerente: escolha uma ou mais opções a serem incluídas no nome alternativo do requerente com base nas informações contidas no Active Directory.

> **Note**
>
> Pelo menos um nome de requerente ou opção de nome alternativo de requerente deve ser especificado.

  - GUID de diretório
  - Nome DNS
  - DNS do domínio
  - E-mail
  - Nome da entidade principal do serviço (SPN)
  - Nome da entidade principal do usuário (UPN)

Definir configurações de processamento e registro de solicitações

Na seção Opções de registro e tratamento de solicitações de certificados, especifique a finalidade dos certificados com base no modelo, escolhendo uma das opções a seguir.

- Signature
- Criptografia
- Assinatura e criptografia

- Assinatura e login com cartão inteligente

Em seguida, escolha qual dos seguintes recursos ativar. As opções variam dependendo do propósito do certificado.

- Excluir certificados inválidos (não arquivar)
- Inclua algoritmos simétricos
- Chave privada exportável

Por fim, escolha uma opção de inscrição do certificado. As opções variam dependendo do propósito do certificado.

- Nenhuma entrada do usuário é necessária
- Solicitar ao usuário durante a inscrição
- Solicitar ao usuário durante a inscrição e exigir entrada do usuário

Configurar extensões de uso de chaves

Na seção Configurações da extensão de uso da chave, escolha a opção para uso da assinatura e uso da chave de criptografia.

Uso da chave de assinatura

- Assinatura digital
- A assinatura é prova de origem (não repúdio)

Uso da chave de criptografia

- Permitir troca de chaves sem criptografia de chave (contrato de chave)
- Permitir troca de chaves somente com criptografia de chave (criptografia de chave)
- Permitir criptografia de dados do usuário (criptografia de dados)

Você também pode optar por Tornar as extensões de uso de chaves críticas para ambos os tipos de chave.

Atribuir políticas de aplicação

Na seção Políticas de aplicações, escolha todas as políticas de aplicações aplicáveis. As políticas disponíveis estão listadas em várias páginas. Algumas políticas podem ser pré-selecionadas devido a configurações anteriores.

Configurar políticas personalizadas de aplicações

Na seção Políticas personalizadas de aplicações, você pode adicionar OIDs personalizados ao modelo e especificar se as extensões da política de aplicação são críticas.

Definir configurações de criptografia

Na seção Configurações de criptografia, escolha as seguintes categorias de configurações de criptografia para certificados com base nesse modelo.

1. O conteúdo na parte superior da seção é determinado pela Método de criação de modelo e Configurações do modelo que você escolheu anteriormente.

   - Se você aceitou a Versão 2 do modelo padrão em Configurações do modelo, as seguintes mensagens de status serão exibidas aqui:
     - Categoria do provedor de criptografia
     - Provedor antigo de serviços criptográficos

     Nesse caso, não há configurações para definir e você pode avançar para a etapa seguinte.
   - Se você especificou a Versão 3 do modelo em Configurações do modelo, as mensagens de status a seguir serão exibidas aqui:
     - Categoria do provedor de criptografia
     - Provedor de armazenamento de chaves

     Você também deve escolher um Algoritmo de chave entre as opções listadas ECDH_P256, ECDH_P384, ECDH_P521 e RSA.
   - Se você especificou a Versão 4 do modelo em Configurações do modelo, deverá escolher entre um Provedor de armazenamento de chaves e um Provedor de serviços criptográficos legados. Se você escolher Provedor de armazenamento de chaves, um Algoritmo de chave também deverá ser escolhido entre as opções listadas: ECDH_P256, ECDH_P384, ECDH_P521 e RSA.
2. Tamanho mínimo da chave (bits): especifique o tamanho mínimo da chave. Essa configuração afetará quais provedores de criptografia estão disponíveis.

3. Escolher quais provedores criptográficos podem ser usados para solicitações: escolha uma das duas opções disponíveis:

   - Solicitações podem usar qualquer provedor disponível no computador do requerente
   - Solicitações devem usar um dos seguintes provedores selecionados

     A escolha dessa opção abre uma lista de Provedores de criptografia. É possível selecionar e priorizar fornecedores usando os botões na coluna Ordem. Os seguintes provedores são compatíveis:

     - Microsoft Base Cryptographic Provider v1.0
     - Microsoft Base DSS and Diffie-Hellman Cryptographic Provider
     - Microsoft Base Smart Card Crypto Provider
     - Microsoft DH SChannel Cryptographic Provider
     - Microsoft Enhanced Cryptographic Provider v1.0
     - Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider
     - Microsoft Enhanced RSA and AES Cryptographic Provider
     - Microsoft RSA SChannel Cryptographic Provider

Configurar grupos e permissões

Na seção Grupos e permissões, você pode ver os modelos de grupos existentes e as permissões para inscrição ou pode escolher o botão Adicionar novos grupos e permissões para adicionar novos. O botão abre um formulário que exige estas informações:

- Nome de exibição
- Identificador de segurança (SID)
- Inscrição, com as opções ALLOW | DENY | NOT SET
- Inscrição automática, com as opções ALLOW | DENY | NOT SET

Configurar modelos de substituição

Na seção Modelos de substituição, você pode notificar o Active Directory de que o modelo atual substitui um ou mais modelos criados no AD. Aplique o modelo substitutivo escolhendo Adicionar modelo do Active Directory para substituir e especificando o nome comum do modelo de substituição.

Configurar a marcação

No painel Tags – opcional, você pode aplicar e remover metadados no seu recurso do AD. Etiquetas são pares de strings de chave/valor em que a chave deve ser exclusiva do recurso e o valor é opcional. O painel exibe todas as etiquetas existentes para o recurso em uma tabela. As ações a seguir são compatíveis.

- Escolha Gerenciar etiquetas para abrir a página Gerenciar etiquetas.
- Escolha Adicionar nova etiqueta para criar uma etiqueta. Preencha o campo Chave e, opcionalmente, Valor. Escolha Salvar alterações para aplicar a etiqueta.
- Escolha o botão Remover ao lado de uma etiqueta para marcá-la para exclusão e escolha Salvar alterações para confirmar.

Analisar e criar

Depois de fornecer as informações necessárias e analisar suas opções, escolha Criar modelo. Isso abre os Detalhes do modelo, em que você pode revisar as configurações do novo modelo, editar ou excluir o modelo, gerenciar grupos e permissões, gerenciar modelos substituídos, gerenciar etiquetas e definir a reinscrição automática para titulares de certificados.

Criar um modelo de conector (CLI)

Use o comando create-template na seção AWS Private CA Conector para Active Directory da AWS CLI.

Criar um modelo de conector (API)

Use a ação CreateTemplate na API do AWS Private CA Connector para Active Directory.

Listar conectores para o Active Directory

É possível usar o console AWS Private CA Connector for Active Directory ou a AWS CLI para listar os conectores que você possui.

Para listar seus conectores usando o console

1. Faça login na sua conta da AWS e abra o console do AWS Private CA Connector para Active Directory em **https://console.aws.amazon.com/pca-connector-ad/home**.

2. Revise as informações na lista Conectores para Active Directory. É possível navegar por várias páginas de conectores usando os números de página no canto superior direito. Cada conector ocupa uma linha exibindo as seguintes colunas de informações por padrão.

- ID do conector: o ID exclusivo do conector.
- Nome do diretório: o recurso do Active Directory associado ao conector.
- Status do conector: o status do conector. Os valores possíveis são: Criando | Ativo | Excluindo | Falha.
- Status do nome da entidade principal do serviço: status do nome da entidade principal do serviço (SPN) associada ao conector. Os valores possíveis são: Criando | Ativo | Excluindo | Falha.
- Status de registro do diretório: status de registro do diretório associado. Os valores possíveis são: Criando | Ativo | Excluindo | Falha.
- Criado em: data e hora da criação do conector.

Ao selecionar o ícone de engrenagem no canto superior direito do console, é possível personalizar o número de conectores mostrados em uma página usando a preferência Tamanho da página.

Para listar seus conectores usando a AWS CLI

Use o comando list-connectors para listar seus conectores.

Para listar seus conectores usando a API

Use a ação ListConnectors na API do AWS Private CA Connector para Active Directory.

Listar modelos de conectores

É possível usar o console do AWS Private CA Connector para Active Directory ou a AWS CLI listar modelos de conectores que você possui. Os modelos de conectores são baseados em modelos BlankenDentityCertificate_APIPassthrough/v1 do AWS Private CA.

Para listar seus modelos usando o console

1. Faça login na sua conta da AWS e abra o console do AWS Private CA Connector para Active Directory em **https://console.aws.amazon.com/pca-connector-ad/home**.
2. Escolha um conector na lista Conectores para Active Directory e escolha Visualizar detalhes.

3. Na página de detalhes do conector, analise as informações na seção Modelos. É possível navegar por várias páginas de modelos usando os números de página no canto superior direito. Cada modelo ocupa uma linha exibindo as seguintes colunas de informações.

- Nome do modelo: o nome legível do modelo.
- Status do modelo> o status do modelo. Os valores possíveis são: Ativo | Excluindo.
- ID do modelo: o identificador exclusivo do modelo.

Para listar seus modelos usando a AWS CLI

Use o comando list-templates para listar modelos para o conector especificado.

Para listar seus modelos usando a API

Use a ação ListTemplates na API AWS Private CA Connector for Active Directory para listar modelos para o conector especificado.

Visualizar detalhes do conector

Use os procedimentos a seguir para visualizar os detalhes de configuração de um conector no console, na linha de comando ou na API do AWS Private CA Connector para Active Directory.

Visualizar um conector (console)

Para visualizar os detalhes de um conector (console)

1. Faça login na sua conta da AWS e abra o console do AWS Private CA Connector para Active Directory em **https://console.aws.amazon.com/pca-connector-ad/home**.
2. Escolha um conector na lista Conectores para Active Directory e escolha Visualizar detalhes.
3. Na página de detalhes do conector, revise as informações no painel Detalhes do conector, que inclui:

   - ID do conector
   - Status do conector
   - Detalhes adicionais de status
   - ARN do conector
   - Endpoint do servidor de políticas de inscrição de certificados

- Diretório denominado
- ID de diretório
- Requerente do CA privada da AWS
- Status de CA privada da AWS
- Endpoint de VPC e grupos de segurança

4. No painel Modelos, é possível criar ou gerenciar modelos associados ao conector.
5. No painel Nome da entidade principal do serviço (SPN), é possível visualizar o nome da entidade principal de serviço associada ao conector.
6. No painel Registro de diretório, é possível visualizar ou alterar o registro do diretório associado ao conector.
7. No painel Etiquetas - opcional, é possível criar ou gerenciar etiquetas associadas ao conector.

Visualizar um conector (CLI)

Use o comando get-connector na seção AWS Private CA Conector para Active Directory da AWS CLI.

Visualizar um conector (API)

Use a ação GetConnector na API do AWS Private CA Connector para Active Directory.

Visualizar detalhes de um modelo de conector

Use os procedimentos a seguir para visualizar os detalhes de configuração de um modelo de conector no console, na linha de comando ou na API do AWS Private CA Connector para Active Directory

Visualizar um modelo (console)

Para visualizar os detalhes de um modelo de conector (console)

1. Faça login na sua conta da AWS e abra o console do AWS Private CA Connector para Active Directory em **https://console.aws.amazon.com/pca-connector-ad/home**.
2. Escolha um conector na lista Conectores para Active Directory e escolha Visualizar detalhes.
3. Na página de detalhes do conector, analise as informações na seção Modelos e selecione o modelo que você deseja inspecionar. Em seguida, escolha Exibir detalhes.

4. Na página de detalhes, o painel Detalhes do modelo mostra as seguintes informações sobre o modelo:

   - Nome do modelo
   - ID do modelo
   - Status do modelo
   - Versão do esquema do modelo
   - Versão do modelo
   - ARN do modelo
   - Tipo de certificado
   - Inscrição automática ativada
   - Período de validade
   - Período de renovação
   - Requisitos de nome do requerente
   - Requisitos de nome alternativo do requerente
   - Configurações de solicitação e inscrição de certificado
   - Categoria do provedor de criptografia
   - Algoritmo de chave
   - Tamanho mínimo da chave (bits)
   - Algoritmo hash
   - Provedores de criptografia
   - Configurações da extensão de uso da chave

   Nesse painel, também é possível fazer as seguintes ações usando os botões Editar, Excluir e Ações.

   - Edite
   - Excluir
   - Gerenciar grupos e permissões: para obter mais informações, consulte Configurar grupos e permissões.
   - Gerenciar modelos substituídos: para obter mais informações, consulte Analisar e criar.

- Gerenciar etiquetas: para obter mais informações, consulte Marcar recursos do Connector para AD.
- Reinscrever todos os detentores de certificados: essa configuração permite que a versão principal de um modelo seja aumentada automaticamente. Todos os membros de grupos do Active Directory que têm permissão para se inscrever em um modelo receberão um novo certificado emitido usando esse modelo. Para obter mais informações, consulte a API UpdateTemplate.

5. O painel inferior mostra uma linha de guias que permitem alterações na configuração do modelo.

- Grupos e permissões: visualize e gerencie permissões de grupos do Active Directory para registrar certificados usando esse modelo. Para obter mais informações, consulte Configurar grupos e permissões
- Políticas de aplicação: visualize e gerencie modelos de políticas de aplicação. Para obter mais informações, consulte Atribuir políticas de aplicação.
- Modelos substituídos: visualize e gerencie modelos substituídos. Para obter mais informações, consulte Analisar e criar.
- Tag opcional: visualize e gerencie a marcação com etiquetas neste modelo. Para obter mais informações, consulte Marcar recursos do Connector para AD.

Para visualizar os detalhes de um modelo de conector (AWS CLI)

Visualizar um modelo (CLI)

Use o comando get-template na seção AWS Private CA Conector para Active Directory da AWS CLI.

Visualizar um modelo (API)

Para visualizar os detalhes de um modelo de conector (API)

Use a ação GetTemplate na API do AWS Private CA Connector para Active Directory.

Gerenciar inscrições de diretório

Para gerenciar inscrições de diretório (console)

Inscrições de diretório para conectores podem ser gerenciadas a partir do nível superior do console do AWS Private CA Connector para Active Directory. Este tópico mostra as opções de gerenciamento disponíveis.

1. Faça login na sua conta da AWS e abra o console do AWS Private CA Connector para Active Directory em **https://console.aws.amazon.com/pca-connector-ad/home**.
2. Na área de navegação à esquerda, escolha Inscrições de diretório.
3. A página Registros de diretórios mostra uma tabela de diretórios registrados com os seguintes campos:

   - ID do diretório: o ID exclusivo do diretório
   - Nome do diretório: o nome do site do domínio do diretório
   - Tipo de diretório
   - Registrado: o status do registro. Os valores compatíveis são CREATING | ACTIVE | DELETING | FAILED.
   - Status do diretório: o status do diretório

   Você pode usar Registrar diretório para criar um novo registro.
4. É possível selecionar um dos registros listados para gerenciá-lo. Isso ativa os botões Exibir detalhes do registro e Cancelar registro de diretório. O botão Exibir detalhes do registro abre a página de detalhes do registro.
5. O painel de Detalhes do registro do diretório exibe as seguintes informações:

   - Nome do site do domínio do diretório
   - ID do diretório: o ID exclusivo do diretório. A escolha do link leva você ao console do AWS Directory Service.
   - Tipo de diretório
   - Status: status do diretório
   - ARN do registro do diretório: o nome do recurso Amazon do registro de diretório
   - Informações adicionais de status
6. No painel Conectores e nomes de entidade principal de serviço (SPNs), você pode gerenciar SPNs para o conector. Para obter mais informações, consulte Visualizar detalhes do conector.
7. No painel Tags – opcional, você pode aplicar e remover metadados no seu recurso do AD. Etiquetas são pares de strings de chave/valor em que a chave deve ser exclusiva do recurso e o valor é opcional. O painel exibe todas as etiquetas existentes para o recurso em uma tabela. As ações a seguir são compatíveis.

   - Escolha Gerenciar etiquetas para abrir a página Gerenciar etiquetas.

- Escolha Adicionar nova etiqueta para criar uma etiqueta. Preencha o campo Chave e, opcionalmente, Valor. Escolha Salvar alterações para aplicar a etiqueta.
- Escolha o botão Remover ao lado de uma etiqueta para marcá-la para exclusão e escolha Salvar alterações para confirmar.

Para gerenciar inscrições de diretório (CLI)

Criar: use o comando create-directory-registration na seção AWS Private CA Connector para Active Directory da AWS CLI.

Recuperar: comando get-directory-registration na seção AWS Private CA Connector para Active Directory da AWS CLI.

Listar: comando list-directory-registrations na seção AWS Private CA Connector para Active Directory da AWS CLI.

Excluir: comando delete-directory-registration na seção AWS Private CA Connector para Active Directory da AWS CLI.

Para gerenciar inscrições de diretório (API)

Criar: ação CreateDirectoryRegistration na API do AWS Private CA Connector para Active Directory.

Recuperar: ação GetDirectoryRegistration na API do AWS Private CA Connector para Active Directory.

Listar: ação ListDirectoryRegistrations na API do AWS Private CA Connector para Active Directory.

Excluir: ação DeleteDirectoryRegistration na API do AWS Private CA Connector para Active Directory.

Gerenciar grupos e permissões do AD para modelos

Para gerenciar grupos e permissões de modelos (console)

Grupos e permissões para um modelo existente podem ser gerenciados na página de detalhes do modelo. Para obter mais informações, consulte Visualizar detalhes do modelo de conector.

Defina permissões sobre quais grupos podem ou não inscrever certificados para o modelo específico. Você fornece o identificador de segurança (SID) do grupo. Em seguida, defina as

permissões de inscrição e inscrição automática para o grupo. Para inscrição automática, tanto a inscrição quanto a inscrição automática devem ser definidas como “Permitir”.

Procure o identificador de segurança do grupo no Active Directory

É possível usar o script abaixo para pesquisar o identificador de segurança do grupo no Active Directory.

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

Para gerenciar grupos e permissões de modelos (CLI)

Criar: comando create-template-group-access-control-entry na seção AWS Private CA Connector para Active Directory da AWS CLI

Atualizar: comando update-template-group-access-control-entry na seção AWS Private CA Connector para Active Directory da AWS CLI.

Recuperar: comando get-template-group-access-control-entry na seção AWS Private CA Connector para Active Directory do AWS CLI.

Listar: comando list-template-group-access-control-entries na seção AWS Private CA Connector para Active Directory da AWS CLI.

Excluir: comando delete-template-group-access-control-entries na seção AWS Private CA Connector para Active Directory do AWS CLI.

Para gerenciar grupos e permissões de modelos (API)

Criar: ação CreateTemplateGroupAccessControlEntry na API do AWS Private CA Connector para Active Directory.

Atualizar: ação UpdateTemplateGroupAccessControlEntry na API do AWS Private CA Connector para Active Directory.

Recuperar: ação GetTemplateGroupAccessControlEntry na API do AWS Private CA Connector para Active Directory.

Listar: ação ListTemplateGroupAccessControlEntries na API do AWS Private CA Connector para Active Directory.

Excluir: ação DeleteTemplateGroupAccessControlEntry na API do AWS Private CA Connector para Active Directory.

Configurar o nome da entidade principal do serviço

Para gerenciar nomes de entidades principais de serviço (console)

O nome da entidade principal do serviço (SPN) de um conector AD existente pode ser gerenciado na página de detalhes do conector. Para obter mais informações, consulte Gerenciar a inscrição de diretório Visualizar detalhes do conector

Para gerenciar nomes de entidades principais de serviço (CLI)

Criar: comando create-service-principal-name na seção AWS Private CA Connector para Active Directory da AWS CLI.

Recuperar: comando get-service-principal-name na seção AWS Private CA Connector para Active Directory da AWS CLI.

Listar: comando list-service-principal-names na seção AWS Private CA Connector para Active Directory da AWS CLI.

Excluir: comando delete-service-principal-name na seção AWS Private CA Connector para Active Directory da AWS CLI.

Para gerenciar nomes de entidades principais de serviço (API)

Criar: ação CreateServicePrincipalName na API do AWS Private CA Connector para Active Directory.

Recuperar: ação GetServicePrincipalName na API do AWS Private CA Connector para Active Directory.

Listar: ação ListServicePrincipalNames na API do AWS Private CA Connector para Active Directory.

Excluir: ação DeleteServicePrincipalName na API do AWS Private CA Connector para Active Directory.

Marcar recursos do Connector para AD

É possível aplicar etiquetas a conectores, modelos e inscrições de diretório. A marcação com etiquetas adiciona metadados a um recurso, o que pode ajudar em tarefas de organização e gerenciamento.

Para gerenciar a marcação de recursos (console)

A marcação de recursos existentes com etiquetas é gerenciada na página de detalhes do recurso. Para obter mais informações, consulte os procedimentos a seguir:

- Visualizar detalhes de um modelo de conector
- Gerenciar inscrições de diretório

Para gerenciar a marcação de recursos (CLI)

Marcar: comando tag-resource na seção AWS Private CA Connector para Active Directory da AWS CLI.

Listar etiquetas: comando list-tags-for-resource na seção AWS Private CA Connector para Active Directory da AWS CLI.

Desmarcar: comando untag-resource na seção AWS Private CA Connector para Active Directory da AWS CLI.

Para gerenciar a marcação de recursos (API)

Marcar: ação TagResource na API do AWS Private CA Connector para Active Directory.

Listar etiquetas: ação ListTagResource na API do AWS Private CA Connector para Active Directory.

Desmarcar: ação UntagResource na API do AWS Private CA Connector para Active Directory.

Importante: é aceitável usar etiquetas para identificar objetos que contêm dados confidenciais. No entanto, as etiquetas propriamente ditas não devem conter informações de identificação pessoal (PII), informações confidenciais ou sensíveis.

CA privada da AWS Conector para Simple Certificate Enrollment Protocol (SCEP) (versão prévia)

O conector para SCEP está em versão prévia AWS Private CA e está sujeito a alterações.

O que é Connector for SCEP?

O Connector for Simple Certificate Enrollment Protocol (SCEP) se conecta AWS Private Certificate Authority aos seus dispositivos móveis e equipamentos de rede habilitados para SCEP. Com o Connector for SCEP, você pode usar AWS Private CA para emitir certificados e registrar seus dispositivos SCEP. O Connector for SCEP está disponível para uso com sistemas populares de gerenciamento de dispositivos móveis (MDM) e foi projetado para funcionar com clientes ou endpoints que oferecem suporte ao SCEP.

Tópicos

Características do conector para SCEP

Support for SCEP — O SCEP é um protocolo amplamente adotado para obter certificados de identidade digital de uma autoridade de certificação (CA) e distribuí-los para dispositivos móveis e equipamentos de rede. Você pode usar o Connector for SCEP para ajudá-lo a inscrever seus endpoints usando o SCEP.

Registro de dispositivos móveis - Você pode usar o Connector for SCEP com sistemas MDM populares, incluindo Microsoft Intune e Jamf Pro.

Emita certificados em grande escala - Depois de configurar seus dispositivos habilitados para SCEP para solicitar certificados por meio do endpoint SCEP do conector, seus clientes podem solicitar certificados automaticamente. AWS Private CA

Como começar a usar o Connector for SCEP

Para começar, inicie o assistente guiado no console de gerenciamento Connector for SCEP, que ajuda você a criar um conector e designar a CA privada a ser usada com o conector. Depois de concluir essas etapas, o Connector for SCEP fornece um endpoint e outros parâmetros de configuração que você pode inserir em seus sistemas MDM ou equipamentos de rede. Depois de configurar seus sistemas MDM ou equipamentos de rede, seus clientes solicitarão automaticamente certificados de. AWS Private CA Para saber mais sobre como começar a usar o Connector for SCEP, consulte. Introdução ao AWS Private Certificate Authority Connector for SCEP

Serviços relacionados

O conector para SCEP está relacionado aos seguintes AWS serviços.

- AWS Private Certificate Authority- AWS Private CA fornece um serviço de CA privada de alta disponibilidade sem o investimento inicial e os custos de manutenção contínuos de operar sua própria CA privada.
- AWS Private CA Conector para Active Directory - O conector para AD vincula seu Active Directory (AD) AWS Private CA a. O conector intermedia a troca de certificados entre AWS Private CA usuários e máquinas gerenciadas pelo seu AD.

Conector de acesso para SCEP

Você pode criar, acessar e gerenciar seus conectores Connector for SCEP usando qualquer uma das seguintes interfaces:

- AWS Management Console- Fornece uma interface web que você pode usar para acessar o Connector for SCEP. Consulte Conector para console de gerenciamento SCEP.
- AWS Command Line Interface- Fornece comandos para um amplo conjunto de AWS serviços, incluindo o Connector for SCEP. O AWS CLI é compatível com Windows, macOS e Linux. Para ter mais informações, consulte AWS Command Line Interface.
- AWS SDKs — forneça APIs específicas de linguagem e cuide de muitos detalhes da conexão, como calcular assinaturas, lidar com novas tentativas de solicitação e tratamento de erros. Para ter mais informações, consulte AWS Command Line Interface.
- Conector para API SCEP - fornece ações de API de baixo nível que você chama usando solicitações HTTPS. Usar a API Connector for SCEP é a maneira mais direta de acessar o serviço. No entanto, a API Connector for SCEP exige que seu aplicativo manipule detalhes de baixo nível,

como a geração do hash para assinar a solicitação e o tratamento de erros. Para obter mais informações, consulte a referência da API Connector for SCEP.

Preços do Connector for SCEP

O conector para SCEP é oferecido como um recurso sem CA privada da AWS custo adicional. Você paga somente pelas AWS Private Certificate Authority operações e certificados usados para criar e atualizar conectores.

Para obter as informações mais recentes sobre CA privada da AWS preços, consulte AWS Private Certificate Authority Preços. Você também pode usar a Calculadora de preços da AWS para estimar os custos.

Conector para conceitos SCEP

O conector para SCEP está em versão prévia AWS Private CA e está sujeito a alterações.

O conector para SCEP é um recurso adicional para. AWS Private Certificate Authority

A seguir estão os principais conceitos do Connector for SCEP:

Solicitação de assinatura de certificado (CSR)

As informações necessárias fornecidas a uma CA para que um certificado digital seja emitido. Essas informações contêm uma chave pública e uma identidade.

Senha do desafio

O protocolo SCEP usa senhas de desafio para autenticar uma solicitação antes de emitir um certificado de uma CA. O Connector for SCEP manipula as senhas de desafio do SCEP com base no tipo de conector. Para ter mais informações, consulte Como funciona o Connector for SCEP.

Revogação do certificado

A revogação do certificado é o processo de revogar um certificado emitido antes da data de expiração. Você pode revogar o certificado de CA privado associado a um conector RevokeCertificatechamando a API, o AWS SDK ou. AWS Command Line Interface AWS CloudFormation

Conector para SCEP

Um conector para links SCEP para seus dispositivos habilitados AWS Private CA para SCEP.

Gerenciamento de dispositivos móveis

O gerenciamento de dispositivos móveis (MDM) permite que os administradores de TI controlem, protejam e apliquem políticas em smartphones, tablets e outros terminais ou dispositivos. Muitos sistemas MDM fornecem integrações integradas para inscrição de certificados com base em SCEP.

CEPO

O SCEP é um protocolo padronizado (RFC 8894) para distribuir certificados automaticamente.
O protocolo fornece um ponto final para os dispositivos solicitarem certificados de uma CA.
O SCEP usa senhas de desafio para autorizar a emissão de certificados para dispositivos.
O SCEP é comumente aplicado para sistemas de gerenciamento de dispositivos móveis (MDM) e equipamentos de rede. As soluções MDM permitem que os administradores de TI controlem, protejam e apliquem políticas em smartphones, tablets e outras entidades, como estações de trabalho da Apple. A maioria das soluções de MDM oferece suporte ao SCEP, como Microsoft Intune, Apple MDM e Jamf Pro. A maioria dos equipamentos de rede, como roteadores, balanceadores de carga, hubs Wi-Fi, dispositivos VPN e firewalls, usa o SCEP para registro automatizado de certificados.

Perfil SCEP

Um perfil SCEP contém parâmetros de configuração que são usados para definir o perfil do certificado. Isso inclui o período de validade do certificado, o tamanho da chave, o nome da configuração do SCEP, a senha do desafio, o número de tentativas malsucedidas e o intervalo de repetições, além de outras informações relevantes para a emissão de certificados. Os sistemas MDM e as plataformas de gerenciamento de certificados normalmente enviam o perfil SCEP ao cliente que solicitará um certificado para autenticação.

Como funciona o Connector for SCEP

O conector para SCEP está em versão prévia AWS Private CA e está sujeito a alterações.

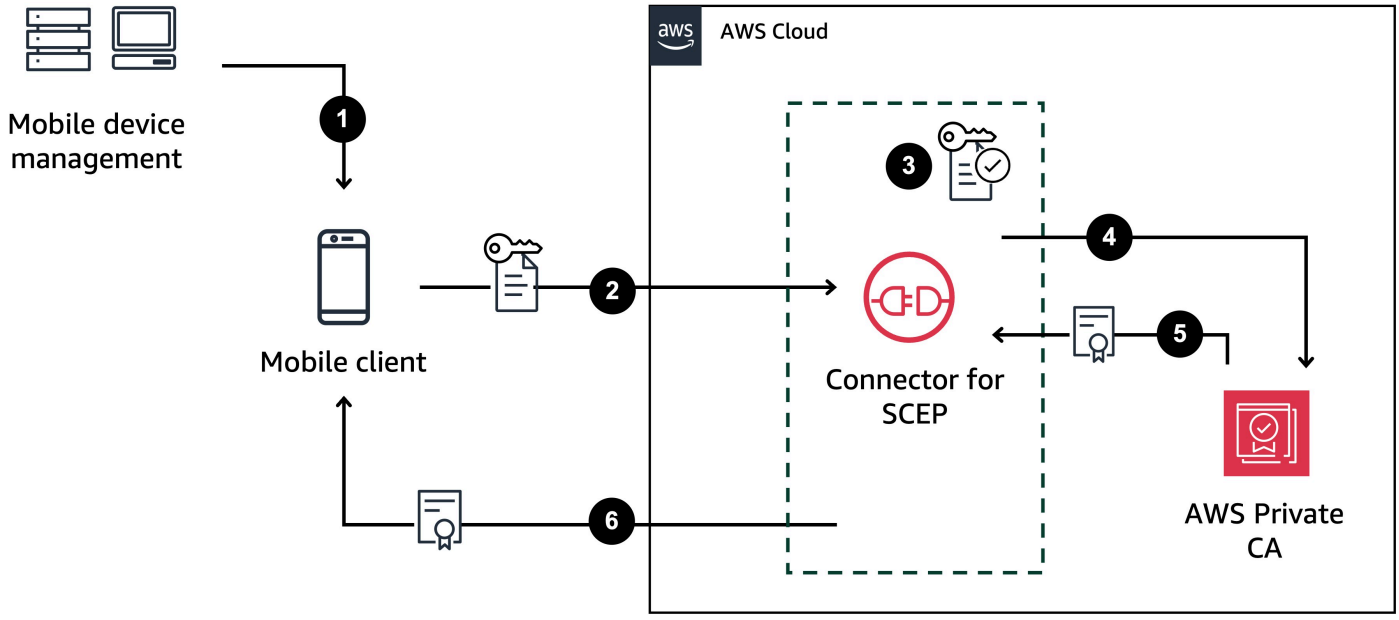
O Simple Certificate Enrollment Protocol (SCEP) é um protocolo padrão usado para inscrição e renovação de certificados. O Connector for SCEP é um servidor SCEP baseado em RFC 8894 que

emite certificados automaticamente para seus clientes SCEP. AWS Private Certificate Authority Quando você cria um conector, o Connector for SCEP fornece um endpoint HTTPS para os clientes SCEP solicitarem certificados. Os clientes se autenticam usando uma senha de desafio incluída como parte da solicitação de assinatura de certificado (CSR) para o serviço. Você pode usar o Connector for SCEP com soluções populares de gerenciamento de dispositivos móveis (MDM), incluindo Microsoft Intune e Jamf Pro, para inscrever dispositivos móveis. Ele funciona com qualquer cliente ou endpoint que ofereça suporte ao SCEP.

O Connector for SCEP oferece dois tipos de conectores: de uso geral e Connector for SCEP para Microsoft Intune. As seções a seguir descrevem como elas funcionam.

Uso geral

Um conector de uso geral foi projetado para funcionar com endpoints que oferecem suporte ao SCEP, exceto o Microsoft Intune, para o qual temos um conector especial. Com os conectores de uso geral, você gerencia as senhas de desafio do SCEP. O diagrama a seguir usa um sistema de gerenciamento de dispositivos móveis (MDM) como exemplo, mas a mesma funcionalidade se aplica se você estiver usando um sistema ou dispositivo análogo habilitado para SCEP.



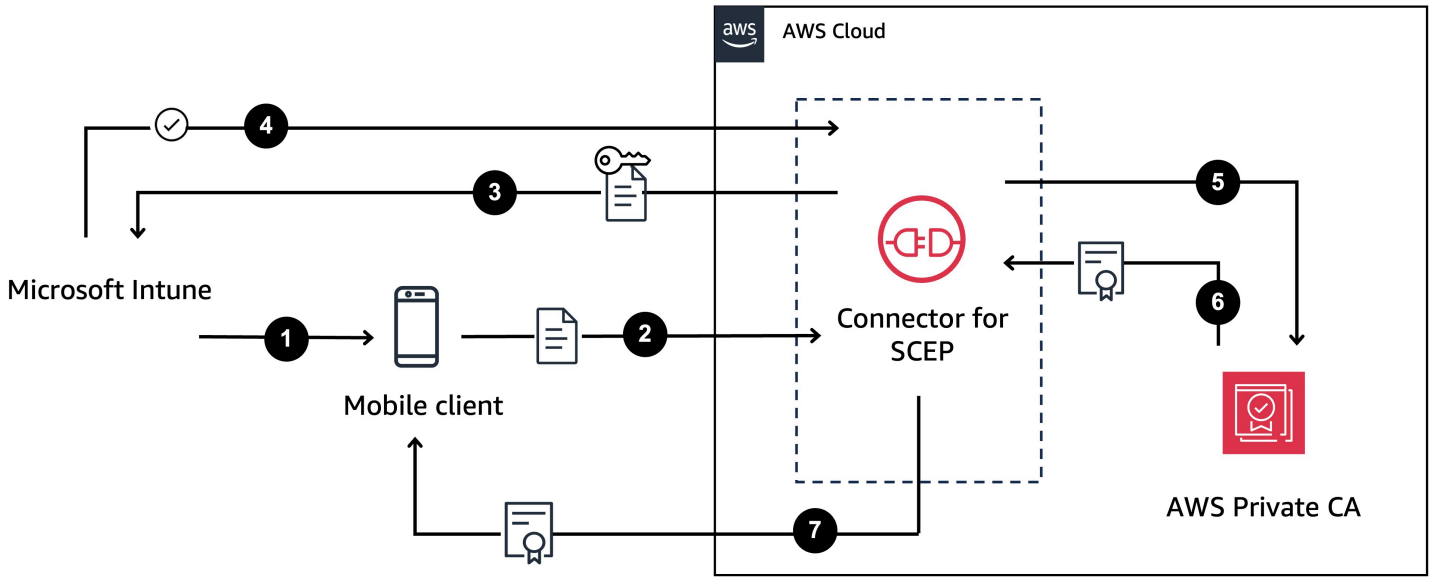
1. O sistema MDM (ou dispositivo ou sistema análogo) envia um perfil SCEP para o cliente móvel. Um perfil SCEP contém parâmetros de configuração que são usados para definir o perfil do certificado, incluindo o período de validade do certificado, o tamanho da chave, o nome da configuração do SCEP, a senha do desafio, o número de tentativas malsucedidas e o intervalo de repetição e outras informações relevantes para a emissão de certificados.

2. O cliente móvel solicita um certificado e também envia uma solicitação de assinatura de certificado (CSR) que inclui uma senha de desafio.
3. O conector para SCEP valida a senha do desafio. Se for válido, o serviço solicitará um certificado AWS Private CA em nome do cliente móvel.
4. AWS Private CA emite o certificado e o envia ao Connector for SCEP.
5. O conector para SCEP envia o certificado emitido para o cliente móvel.

AWS Private Certificate Authority Conector para SCEP para Microsoft Intune

AWS Private CA O Connector for SCEP para Microsoft Intune foi projetado para ser usado com o Microsoft Intune. Com o tipo de conector Connector for SCEP para Microsoft Intune, você usará o Microsoft Intune para gerenciar suas senhas de desafio do SCEP. Para obter mais informações sobre como usar o Connector for SCEP com o Microsoft Intune, consulte. Usando o Connector for SCEP para Microsoft Intune

Quando você usa o Connector for SCEP com o Microsoft Intune, determinadas funcionalidades são habilitadas acessando o Microsoft Intune por meio da API da Microsoft. O uso do Conector para SCEP e AWS serviços associados não elimina a necessidade de ter uma licença válida para o uso do serviço Microsoft Intune. Você também deve revisar as Políticas de Proteção de Aplicativos do Microsoft Intune®.



1. O Microsoft Intune envia um perfil SCEP para o cliente móvel. O perfil contém uma senha de desafio criptografada que o cliente móvel coloca na CSR.

2. O cliente móvel solicita um certificado e envia o CSR para o Connector for SCEP.
3. O Connector for SCEP envia o CSR ao Microsoft Intune para autorização.
4. O Microsoft Intune descriptografa a senha de desafio no CSR. Se for válido, o Microsoft Intune envia a aprovação ao Connector for SCEP para emitir o certificado para o cliente móvel.
5. O Connector for SCEP solicita um certificado AWS Private CA em nome do cliente móvel.
6. AWS Private CA emite o certificado e o envia ao Connector for SCEP.
7. O conector para SCEP envia o certificado emitido para o cliente móvel.

Considerações e limitações ao trabalhar com o Connector for SCEP

Considerações

Modos de operação CA

Você só pode usar o Connector for SCEP com CAs privadas que usam um modo operacional de uso geral. O padrão do Connector for SCEP é emitir certificados com um período de validade de um ano. Uma CA privada usando um modo de certificado de curta duração não suporta a emissão de certificados com um período de validade superior a sete dias. Para obter informações sobre os modos operacionais, consulteModos de autoridades certificadoras.

Desafie as senhas

- Distribua suas senhas de desafio com muito cuidado e compartilhe somente com pessoas e clientes altamente confiáveis. Uma única senha de desafio pode ser usada para emitir qualquer certificado, com qualquer assunto e SANs, o que representa um risco de segurança.
- Se estiver usando um conector de uso geral, recomendamos que você alterne manualmente suas senhas de desafio com frequência.

Conformidade com a RFC 8894

O conector para SCEP se desvia do protocolo RFC 8894 fornecendo pontos de extremidade HTTPS em vez de pontos de extremidade HTTP.

CSRs

- Se uma solicitação de assinatura de certificado (CSR) enviada ao Connector for SCEP não incluir a extensão Extended Key Usage (EKU), definiremos o valor de EKU como. `clientAuthentication` Para obter informações, consulte 4.2.1.12. Uso estendido da chave no RFC 5280.
- Oferecemos suporte `ValidityPeriod` e `ValidityPeriodUnits` personalizamos atributos em CSRs. Se sua CSR não incluir um`ValidityPeriod`, emitimos um certificado com um período de validade de um ano. Lembre-se de que talvez você não consiga definir esses atributos em seu sistema MDM. Mas se você puder configurá-los, nós os apoiaremos. Para obter informações sobre esses atributos, consulte SZENrollment_name_value_pair.

Compartilhamento de endpoints

Distribua os endpoints de um conector somente para partes confiáveis. Trate os endpoints como secretos, pois qualquer pessoa que possa encontrar seu nome de domínio e caminho exclusivos e totalmente qualificados pode recuperar seu certificado CA.

Limitações

As limitações a seguir se aplicam ao Connector for SCEP.

Senhas de desafio dinâmico

Você só pode criar senhas de desafio estáticas com conectores de uso geral. Para usar senhas dinâmicas com um conector de uso geral, você deve criar seu próprio mecanismo de rotação que empregue as senhas estáticas do conector. Os tipos de conector Connector for SCEP para Microsoft Intune oferecem suporte para senhas dinâmicas, que você gerencia usando o Microsoft Intune.

HTTP

O Connector for SCEP suporta somente HTTPS e cria redirecionamentos para chamadas HTTP. Se seu sistema depende de HTTP, certifique-se de que ele possa acomodar os redirecionamentos HTTP fornecidos pelo Connector for SCEP.

CAs privadas compartilhadas

Você só pode usar o Connector for AD com CAs privadas das quais você é o proprietário.

Configurando o conector para SCEP

O conector para SCEP está em versão prévia AWS Private CA e está sujeito a alterações.

Os procedimentos nesta seção ajudam você a começar a usar o Connector for SCEP. Ele pressupõe que você já tenha criado uma AWS conta. Depois de concluir as etapas desta página, você pode continuar com a criação de um conector para o SCEP.

Tópicos

Etapa 1: criar uma AWS Identity and Access Management política

Para criar um conector para o SCEP, você precisa criar uma política do IAM que conceda ao Connector for SCEP a capacidade de criar e gerenciar os recursos necessários ao conector e emitir certificados em seu nome. Para obter mais informações sobre o IAM, consulte O que é o IAM? no Guia do usuário do IAM.

O exemplo a seguir é uma política gerenciada pelo cliente que você pode usar para o Connector for SCEP.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "pca-connector-scep:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
```

```
                "acm-pca:ListCertificateAuthorities",
                "acm-pca:ListTags",
                "acm-pca:PutPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "acm-pca:IssueCertificate",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APICSRPassthrough/V*"
                },
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "pca-connector-scep.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:CreateResourceShare",
                "ram:GetResourcePolicies",
                "ram:GetResourceShareAssociations",
                "ram:GetResourceShares",
                "ram:ListPrincipals",
                "ram:ListResources",
                "ram:ListResourceSharePermissions",
                "ram:ListResourceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

Etapa 2: criar uma CA privada

Para usar o Connector for SCEP, você precisa associar uma CA privada AWS Private Certificate Authority ao conector. Recomendamos que você use uma CA privada que seja somente para o conector, devido às vulnerabilidades de segurança inerentes que estão presentes no protocolo SCEP.

A CA privada deve atender aos seguintes requisitos:

- Ele deve estar em um estado ativo e usar o modo operacional de uso geral.
- Você deve possuir a CA privada. Você não pode usar uma CA privada que foi compartilhada com você por meio do compartilhamento entre contas.

Esteja ciente das seguintes considerações ao configurar sua CA privada para uso com o Connector for SCEP:

- Restrições de nome DNS — Considere usar restrições de nome DNS como forma de controlar quais domínios são permitidos ou proibidos nos certificados emitidos para seus dispositivos SCEP. Para obter mais informações, consulte Como impor restrições de nome DNS em. AWS Private Certificate Authority
- Revogação — habilite OCSP ou CRLs em sua CA privada para permitir a revogação. Para ter mais informações, consulte Configurar um método de revogação de certificado.
- PII — Recomendamos que você não adicione informações de identificação pessoal (PII) ou outras informações confidenciais ou sigilosas em seus certificados CA. No caso de uma falha de segurança, isso ajuda a limitar a exposição de informações confidenciais.
- Armazene certificados raiz em repositórios confiáveis — Armazene seus certificados CA raiz nos repositórios confiáveis do dispositivo, para que você possa verificar os certificados e os valores de retorno de GetCertificateAuthorityCertificate. Para obter informações sobre lojas fiduciárias relacionadas a elas AWS Private CA, consulteCA raiz .

Para obter informações sobre como criar uma CA privada, consulteCriar uma CA privada.

Etapa 3: criar um compartilhamento de recursos usando AWS Resource Access Manager

Se você estiver usando o Connector for SCEP programaticamente usando a API AWS Command Line Interface, AWS SDK ou Connector for SCEP, precisará compartilhar sua CA privada com o Connector for SCEP usando o compartilhamento principal do serviço. AWS Resource Access Manager Isso dá ao Connector for SCEP acesso compartilhado à sua CA privada. Quando você cria um conector no AWS console, criamos automaticamente o compartilhamento de recursos para você. Para obter informações sobre compartilhamento de recursos, consulte Criar um compartilhamento de recursos no Guia AWS RAM do usuário.

Para criar um compartilhamento de recursos usando o AWS CLI, você pode usar o AWS RAM create-resource-share comando. O comando a seguir cria um compartilhamento de recursos. *Especifique o ARN da CA privada que você deseja compartilhar como o valor de resource-arns.*

```
$  aws ram create-resource-share \
--region us-east-1 \
--name MyPcaConnectorScepResourceShare \
--permission-arns arn:aws:ram::aws:permission/
AWSRAMBlankEndEntityCertificateAPICSRPassthroughIssuanceCertificateAuthority \
--resource-arns arn:aws:acm-pca:Region:account:certificate-authority/CA_ID \
--principals pca-connector-scep.amazonaws.com \
--sources account
```

O responsável pelo serviço que liga `CreateConnector` tem permissões de emissão de certificados na CA privada. Para evitar que os principais serviços que usam o Connector for SCEP tenham acesso geral aos seus CA privada da AWS recursos, restrinja suas permissões de uso. `CalledVia`

Introdução ao AWS Private Certificate Authority Connector for SCEP

O conector para SCEP está em versão prévia AWS Private CA e está sujeito a alterações.

Com o AWS Private Certificate Authority Connector for SCEP, você pode emitir certificados de sua CA privada para dispositivos habilitados para SCEP e sistemas de gerenciamento de dispositivos móveis (MDM). Ao criar um conector, AWS Private Certificate Authority cria uma URL SCEP pública para você solicitar certificados e também fornece informações que você pode usar para integrar aos seus sistemas MDM.

Para emitir certificados, você deve criar uma CA AWS Private Certificate Authority privada, criar um conector e configurar seus sistemas e dispositivos MDM habilitados para SCEP para solicitar certificados do conector.

Tópicos

- Antes de começar

Antes de começar

O tutorial a seguir orienta você pelo processo de criação de um conector para o SCEP.

Para seguir este tutorial, você precisará de uma CA privada e de um dispositivo compatível com SCEP. Você também deve primeiro cumprir os pré-requisitos listados na seção. Configurando o conector para SCEP

O procedimento a seguir orienta você sobre como criar um conector usando o AWS console.

Tarefas

Etapa 1: criar um conector

Você criará um conector para uso geral ou um conector para SCEP para Microsoft Intune. Os conectores de uso geral são projetados para uso com endpoints habilitados para SCEP, e você gerencia as senhas de desafio do SCEP. O Connector for SCEP para Microsoft Intune é para uso com o Microsoft Intune, e você gerencia as senhas de desafio usando o Microsoft Intune.

General-purpose

Para criar um conector para uso geral

Faça login na sua AWS conta e abra o console do Connector for SCEP em. **https://console.aws.amazon.com/pca-connector-scep/home**

1. Escolha Criar conector.
2. Na página Criar conector, opcionalmente, dê ao conector um nome amigável no campo Etiqueta de nome. O nome será exibido na sua lista de conectores. Se desejar, você pode adicionar mais tags ao conector selecionando Adicionar mais tags. Uma tag é um rótulo que você atribui a um AWS recurso. Cada tag consiste em uma chave e um valor opcional. Você pode usar tags para pesquisar e filtrar seus recursos ou monitorar seus AWS custos.

3. Em Tipo de conector, escolha Uso geral.
4. Em CA privada, escolha a CA privada a ser usada com esse conector. Ou crie uma nova selecionando Criar CA privada. Devido às vulnerabilidades inerentes ao protocolo SCEP, recomendamos o uso de uma CA privada dedicada a esse conector. Se você criou uma nova CA, ao terminar de criá-la AWS Private CA, retorne ao console Connector for SCEP e atualize a lista de CAs privadas. Sua nova CA privada deve estar disponível para seleção.
5. Em Senha do desafio, selecione Gerar automaticamente a senha do desafio. Geraremos uma senha de desafio estática para você quando criarmos esse conector.
6. Selecione Criar conector.

Microsoft Intune

Para criar um conector para SCEP para Microsoft Intune

Faça login na sua AWS conta e abra o console do Connector for SCEP em. **https://console.aws.amazon.com/pca-connector-scep/home**

1. Escolha Criar conector.
2. Na página Criar conector, opcionalmente, dê ao conector um nome amigável no campo Etiqueta de nome. O nome será exibido na sua lista de conectores. Se desejar, você pode adicionar mais tags ao conector selecionando Adicionar mais tags. Uma tag é um rótulo que você atribui a um AWS recurso. Cada tag consiste em uma chave e um valor opcional. Você pode usar tags para pesquisar e filtrar seus recursos ou monitorar seus AWS custos.
3. Em Tipo de conector, escolha Microsoft Intune.
   a. Em ID do aplicativo (cliente), insira o ID do aplicativo (cliente) do registro do aplicativo Microsoft Entra ID. Para obter informações sobre como usar o Microsoft Intune com o Connector for SCEP, consulte. Usando o Connector for SCEP com sistemas MDM
   b. Para ID do diretório (inquilino) ou domínio primário, insira o ID do diretório (inquilino) ou o domínio principal do registro do aplicativo Microsoft Entra ID.
4. Em CA privada, escolha a CA privada a ser usada com esse conector. Ou crie uma nova selecionando Criar CA privada. Devido às vulnerabilidades inerentes ao protocolo SCEP, recomendamos o uso de uma CA privada dedicada a esse conector. Se você criou uma nova CA, ao terminar de criá-la AWS Private CA, retorne ao console Connector for SCEP e atualize a lista de CAs privadas. Sua nova CA privada deve estar disponível para seleção.
5. Selecione Criar conector.

Etapa 2: Copie os detalhes do conector em seu sistema MDM

Depois de criar seu conector, você precisará copiar os seguintes detalhes do conector para o seu sistema MDM. Para ver os detalhes de um conector usando o console, selecione o conector na lista na página Conectores para console SCEP.

- URL pública do SCEP - Esse é o endpoint do conector do qual seus clientes do SCEP solicitarão certificados. Tome cuidado para fornecer esse endpoint somente para entidades confiáveis.
- (Uso geral) Senha de desafio - Em Senhas de desafio, selecione a senha que você gerou automaticamente no procedimento anterior e, em seguida, selecione Exibir senha para ver a senha. Para criar uma senha adicional, selecione Criar senha. Tenha o cuidado de distribuir senhas com cuidado e somente para pessoas e clientes altamente confiáveis. Uma única senha de desafio pode ser usada para emitir qualquer certificado, com qualquer assunto e SANs, e, portanto, deve ser tratada com cuidado.
- (Microsoft Intune) Valores do Open ID - Se você estiver integrando com o Microsoft Intune, deverá copiar o emissor do Open ID, o assunto do Open ID e o público do Open ID na credencial do OpenID Connect (OIDC) do registro do aplicativo Microsoft Entra. Para ter mais informações, consulte Usando o Connector for SCEP com sistemas MDM.

Usando o Connector for SCEP com sistemas MDM

O conector para SCEP está em versão prévia AWS Private Certificate Authority e está sujeito a alterações.

As seções a seguir descrevem como usar o Connector for SCEP com sistemas específicos de gerenciamento de dispositivos móveis (MDM).

Tópicos

- Usando o conector para SCEP com o Jamf Pro
- Usando o Connector for SCEP para Microsoft Intune

Usando o conector para SCEP com o Jamf Pro

Você pode usar AWS Private CA como autoridade de certificação externa (CA) com a solução Jamf Pro de gerenciamento de dispositivos móveis (MDM). Este guia fornece instruções sobre como

configurar o Jamf Pro como um proxy SCEP depois de criar um AWS Private Certificate Authority conector para SCEP.

Requisitos do Jamf Pro

Sua implementação do Jamf Pro deve atender aos seguintes requisitos.

- Você deve usar o Jamf Pro 10.0.0 ou posterior.
- Você deve habilitar a configuração Ativar autenticação baseada em certificado no Jamf Pro. Você pode encontrar detalhes sobre essa configuração na página Configurações de segurança do Jamf Pro na documentação do Jamf Pro.

Pré-requisitos

Para usar o Connector for SCEP com o Jamf Pro, você deve primeiro criar uma CA privada e um conector de uso geral para o SCEP. Para obter instruções, consulte Configurando o conector para SCEP.

Configurar AWS Private CA como uma CA externa no Jamf Pro

Depois de criar um conector para o SCEP, você deve definir AWS Private CA como CA externo no Jamf Pro. Você pode definir AWS Private CA como uma CA externa global. Ou você pode usar um perfil de configuração do Jamf Pro para emitir certificados diferentes AWS Private CA para diferentes casos de uso, como emitir certificados para um subconjunto de dispositivos em sua organização. A orientação sobre a implementação dos perfis de configuração do Jamf Pro está além do escopo deste documento.

Para configurar AWS Private CA como uma CA externa no Jamf Pro

1. No console Jamf Pro, acesse a página de configurações dos certificados PKI acessando Configurações > Global > Certificados PKI.
2. Selecione a guia Modelo de certificado de gerenciamento.
3. Selecione CA externa.
4. Selecione Edit (Editar).
5. (Opcional) Selecione Ativar Jamf Pro como proxy SCEP para perfis de configuração. Você pode usar os perfis de configuração do Jamf Pro para emitir diferentes certificados personalizados para casos de uso específicos. Para obter orientação sobre como usar perfis de configuração

no Jamf Pro, consulte Habilitando o Jamf Pro como proxy SCEP para perfis de configuração na documentação do Jamf Pro.

6. Selecione Usar uma CA externa habilitada para SCEP para registro de computadores e dispositivos móveis.
7. (Opcional) Selecione Usar Jamf Pro como proxy SCEP para registro de computadores e dispositivos móveis. Se você tiver falhas na instalação do perfil, consulteSolucionar problemas de falhas na instalação do perfil.
8. Copie e cole o URL SCEP público do Connector for SCEP dos detalhes do conector para o campo URL no Jamf Pro. Para ver os detalhes de um conector, escolha o conector na lista Conectores para SCEP. Como alternativa, você pode obter o URL chamando GetConnectore copiando o `Endpoint` valor da resposta.
9. (Opcional) Insira o nome da instância no campo Nome. Por exemplo, você pode nomeá-lo AWS Private CA.
10. Selecione Estático para o tipo de desafio.
11. Copie a senha de desafio do seu conector e cole-a no campo Desafio. Para ver as senhas de desafio do seu conector, navegue até a página de detalhes do conector no AWS console e selecione o botão Exibir senha. Como alternativa, você pode obter a senha de desafio de um conector chamando GetChallengePassword e copiando o `Password` valor da resposta.
12. Cole a senha do desafio no campo Verificar desafio.
13. Escolha um tamanho de chave. Recomendamos um tamanho de chave de 2048 ou superior.
14. (Opcional) Selecione Usar como assinatura digital. Selecione essa opção para fins de autenticação para conceder aos dispositivos acesso seguro a recursos como Wi-Fi e VPN.
15. (Opcional) Selecione Usar para criptografia de chave.
16. (Opcional) Insira uma string hexadecimal no campo Impressão digital. Para obter instruções sobre como criar uma impressão digital da sua CA privada, consulte(Opcional) Adicionar uma impressão digital da CA.
17. Selecione Save (Salvar).

Crie e faça upload de um certificado de assinatura de perfil

Para usar o Connector for SCEP com o Jamf Pro, você deve fornecer os certificados de assinatura e CA para a CA privada associada ao seu conector. Você pode fazer isso carregando um repositório de chaves de certificado de assinatura de perfil para o Jamf Pro que contém os dois certificados. Você precisa gerar uma solicitação de assinatura de certificado (CSR) usando seus processos

internos e fazer com que ela seja assinada por AWS Private Certificate Authority. As instruções a seguir explicam como criar um repositório de chaves de certificado e carregá-lo no Jamf Pro. O exemplo a seguir usa o OpenSSL, mas você pode gerar uma solicitação de assinatura de certificado usando seu método preferido.

1. Usando o OpenSSL, gere uma chave privada executando o seguinte comando:

```
openssl genrsa -out local.key 2048
```

2. Gere uma solicitação de assinatura de certificado (CSR):

```
openssl req -new -key local.key -sha512 -out local.csr -
subj "/CN=MySigningCertificate/O=MyOrganization" -addext
 keyUsage=critical,digitalSignature,nonRepudiation
```

3. Usando o AWS CLI, emita o certificado de assinatura usando o CSR que você gerou na etapa dois. Execute o comando a seguir e anote o ARN do certificado na resposta:

```
aws acm-pca issue-certificate --certificate-authority-arn <SAME CA AS USED ABOVE,
 SO IT'S TRUSTED> --csr fileb://local.csr --signing-algorithm SHA512WITHRSA --
validity Value=365,Type=DAYS
```

4. Obtenha o certificado de assinatura executando o seguinte comando usando o ARN do certificado da etapa 3:

```
aws acm-pca get-certificate --certificate-authority-arn <SAME CA AS USED ABOVE, SO
 IT'S TRUSTED> --certificate-arn <ARN OF NEW CERTIFICATE> | jq -r '.Certificate'
 >local.crt
```

5. Obtenha o certificado CA executando o seguinte comando:

```
aws acm-pca get-certificate-authority-certificate --certificate-authority-arn <SAME
 CA AS USED ABOVE, SO IT'S TRUSTED> | jq -r '.Certificate' > ca.crt
```

6. Usando o OpenSSL, imprima o repositório de chaves do certificado de assinatura no formato p12. Você usará os `crt` arquivos que gerou durante as etapas quatro e cinco. Execute o seguinte comando:

```
openssl pkcs12 -export -in local.crt -inkey local.key -certfile ca.crt -name "CA
 Chain" -out local.p12
```

7. Quando solicitado, insira uma senha de exportação. Essa senha é a senha do seu keystore e você precisará usá-la mais tarde.
8. No Jamf Pro, navegue até o Modelo de Certificado de Gerenciamento e vá para o painel CA externo.
9. Na parte inferior do painel CA externa, selecione Alterar assinatura e certificados CA.
10. Siga as instruções na tela para carregar os certificados de assinatura e CA para a CA externa.

(Opcional) Adicionar uma impressão digital da CA

Adicionar uma impressão digital da CA permite que os dispositivos gerenciados verifiquem a CA e solicitem somente certificados da CA.

1. Obtenha o certificado CA privado de qualquer AWS Private CA console ou usando GetCertificateAuthorityCertificateo. Salve-o como `ca.pem` arquivo.
2. No OpenSSL, execute o seguinte comando:

```
openssl x509 -in ca.pem -sha256 -fingerprint
```

3. Copie e cole a saída no campo Impressão digital mencionado no procedimento anterior.

(Opcional) Instalar o certificado durante a inscrição iniciada pelo usuário

Para instalar o certificado CA privado do seu conector em um cliente ou dispositivo durante a inscrição iniciada pelo usuário, defina as configurações de inscrição iniciada pelo usuário do Jamf Pro. Isso ajuda o Jamf Pro a instalar seus AWS Private CA certificados no cliente ou dispositivo quando eles solicitam um certificado. É sua responsabilidade testar sua configuração para garantir que ela seja compatível com a implementação do Connector for SCEP. Para obter informações sobre as configurações de inscrição iniciadas pelo usuário do Jamf Pro, consulte Configurações de inscrição iniciadas pelo usuário na documentação do Jamf Pro.

Solucionar problemas de falhas na instalação do perfil

Se você estiver enfrentando falhas na instalação do perfil após ativar o Use Jamf Pro as SCEP Proxy para registro de computadores e dispositivos móveis, tente o seguinte.

Mensagem de erro	Mitigação
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:15001>`	Se você receber essa mensagem de erro ao tentar se inscrever, tente se inscrever novamente. Podem ser necessárias várias tentativas até que a inscrição seja bem-sucedida.
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:14006>`	Sua senha de desafio pode estar configurada incorretamente. Verifique se a senha de desafio no Jamf Pro corresponde à senha de desafio do seu conector.

Usando o Connector for SCEP para Microsoft Intune

Você pode usar AWS Private CA como uma autoridade de certificação (CA) externa com o sistema Microsoft Intune de gerenciamento de dispositivos móveis (MDM). Este guia fornece instruções sobre como configurar o Microsoft Intune depois de criar um conector para SCEP para Microsoft Intune.

Pré-requisitos

Antes de criar um conector para SCEP para Microsoft Intune, você deve preencher os seguintes pré-requisitos.

- Crie um ID Entra.
- Crie um inquilino do Microsoft Intune.
- Crie um registro de aplicativo em seu Microsoft Entra ID. Consulte Atualizar as permissões solicitadas de um aplicativo no Microsoft Entra ID na documentação do Microsoft Entra para obter informações sobre como gerenciar permissões em nível de aplicativo para seu Registro de Aplicativo. O registro do aplicativo deve ter as seguintes permissões:
  - No Intune, defina scep_challenge_provider.
  - Para o Microsoft Graph, defina Application.Read.All e User.Read.
- Você deve conceder ao aplicativo o consentimento do administrador do Registro do Aplicativo. Para obter informações, consulte Conceder consentimento de administrador de todo o locatário a um aplicativo na documentação do Microsoft Entra.

> **Tip**
>
> Ao criar o Registro do Aplicativo, anote o ID do Aplicativo (cliente) e o ID do Diretório (locatário) ou domínio principal. Ao criar seu Connector for SCEP para Microsoft Intune, você inserirá esses valores. Para obter informações sobre como obter esses valores, consulte Criar um aplicativo e um principal de serviço do Microsoft Entra que possa acessar recursos na documentação do Microsoft Entra.

Conceda AWS Private CA permissão para usar o aplicativo Microsoft Entra ID

Depois de criar um Conector para SCEP para Microsoft Intune, você deve criar uma credencial federada no Registro de Aplicativos da Microsoft para que o Conector para SCEP possa se comunicar com o Microsoft Intune.

Para configurar AWS Private CA como uma CA externa no Microsoft Intune

1. No console Microsoft Entra ID, navegue até os registros do aplicativo.
2. Escolha o aplicativo que você criou para usar com o Connector for SCEP. A ID do aplicativo (cliente) do aplicativo em que você clica deve corresponder à ID especificada ao criar o conector.
3. Selecione Certificados e segredos no menu suspenso Gerenciado.
4. Selecione a guia Credenciais federadas.
5. Selecione Adicionar uma credencial.
6. No menu suspenso Cenário de credenciais federadas, escolha Outro emissor.
7. Copie e cole o valor do emissor do OpenID dos detalhes do Connector for SCEP for Microsoft Intune no campo Emissor. Para ver os detalhes de um conector, escolha o conector na lista Conectores para SCEP no console. AWS Como alternativa, você pode obter o URL chamando GetConnectore copiando o `Issuer` valor da resposta.
8. Copie e cole o valor do OpenID Audience dos detalhes do Connector for SCEP for Microsoft Intune no campo Audience. Para ver os detalhes de um conector, escolha o conector na lista Conectores para SCEP no console. AWS Como alternativa, você pode obter o URL chamando GetConnectore copiando o `Subject` valor da resposta.
9. (Opcional) Insira o nome da instância no campo Nome. Por exemplo, você pode nomeá-lo AWS Private CA.

10. (Opcional) Insira uma descrição no campo Descrição.
11. Selecione Editar (opcional) no campo Público. Copie e cole o valor do assunto do OpenID do seu conector no campo Assunto. Você pode ver o valor do emissor do OpenID na página de detalhes do conector no console. AWS Como alternativa, você pode obter o URL chamando GetConnectore copiando o `Audience` valor da resposta.
12. Selecione Adicionar.

Configurar um perfil de configuração do Microsoft Intune

Depois de dar AWS Private CA a permissão para chamar o Microsoft Intune, você deve usar o Microsoft Intune para criar um perfil de configuração do Microsoft Intune que instrua os dispositivos a entrar em contato com o Connector for SCEP para emissão de certificados.

1. Crie um perfil de configuração de certificado confiável. Você deve carregar o certificado CA raiz da cadeia que está usando com o Connector for SCEP no Microsoft Intune para estabelecer confiança. Para obter informações sobre como criar um perfil de configuração de certificado confiável, consulte Perfis de certificado raiz confiáveis para o Microsoft Intune na documentação do Microsoft Intune.
2. Crie um perfil de configuração do certificado SCEP que direcione seus dispositivos para o conector quando precisarem de um novo certificado. O tipo de perfil do perfil de configuração deve ser Certificado SCEP. Para o certificado raiz do perfil de configuração, certifique-se de usar o certificado confiável que você criou na etapa anterior.

   Para URLs do servidor SCEP, copie e cole o URL público do SCEP dos detalhes do seu conector no campo URLs do servidor SCEP. Para ver os detalhes de um conector, escolha o conector na lista Conectores para SCEP. Como alternativa, você pode obter o URL chamando e GetConnector, em seguida, copiar o `Endpoint` valor da resposta. Para obter orientação sobre a criação de perfis de configuração no Microsoft Intune, consulte Criar e atribuir perfis de certificado SCEP no Microsoft Intune na documentação do Microsoft Intune.

   > **Note**
   >
   > Para dispositivos que não sejam Mac OS e iOS, se você não definir um período de validade no perfil de configuração, o Connector for SCEP emitirá um certificado com validade de um ano. Se você não definir um valor de uso estendido da chave (EKU) no perfil de configuração, o Connector for SCEP emitirá um certificado com o EKU definido com. `Client Authentication (Object Identifier: 1.3.6.1.5.5.7.3.2)`

Para dispositivos macOS `ExtendedKeyUsage` ou iOS, o Microsoft Intune não respeita nossos `Validity` parâmetros em seus perfis de configuração. Para esses dispositivos, o Connector for SCEP emite um certificado com um período de validade de um ano para esses dispositivos por meio da autenticação do cliente.

Verifique a conexão com o Connector for SCEP

Depois de criar um perfil de configuração do Microsoft Intune que aponta para o endpoint Connector for SCEP, confirme se um dispositivo registrado pode solicitar um certificado. Para confirmar, verifique se não há falhas na atribuição de políticas. Para confirmar, no portal do Intune, navegue até Dispositivos > Gerenciar dispositivos > Configuração e verifique se não há nada listado em Falhas de atribuição de política de configuração. Se houver, confirme sua configuração com as informações dos procedimentos anteriores. Se sua configuração estiver correta e ainda houver falhas, consulte Coletar dados disponíveis do dispositivo móvel.

Para obter informações sobre o registro de dispositivos, consulte O que é registro de dispositivos? na documentação do Microsoft Intune.

Solução de problemas

Consulte os tópicos a seguir se você encontrar problemas ao usar o CA privada da AWS.

Tópicos

Criar e assinar um certificado CA privado

Depois de criar a CA privada, você deve recuperar a CSR e enviá-la para um intermediário ou CA raiz em sua infraestrutura X.509. A CA usa a CSR para criar o certificado de CA privada e, em seguida, assina o certificado antes de retorná-lo para você.

Infelizmente, não é possível fornecer instruções específicas sobre problemas relacionados à criação e assinatura do certificado de CA privada. Os detalhes da infraestrutura X.509 e hierarquia da CA nela estão além do escopo desta documentação do CA privada da AWS .

Latência em respostas do OCSP

A capacidade de resposta do OCSP pode ser mais lenta se o chamador estiver geograficamente distante de um cache de borda regional ou da região da CA emissora. Para obter mais informações sobre a disponibilidade regional do cache de borda, consulte Rede de borda global. Convém emitir certificados em uma região próxima de onde eles serão usados.

Configurar o Amazon S3 para permitir a criação de um bucket de CRL

A CA privada poderá falhar ao criar um bucket de CRL se o Bloqueio de Acesso Público do Amazon S3 (configurações do bucket) for aplicado à sua conta. Verifique as configurações do Amazon S3 se isso ocorrer. Para obter mais informações, consulte Usar o Amazon S3 Block Public Access.

Revogar um certificado de CA autoassinado

Não é possível revogar um certificado CA autoassinado. Em vez disso, você deve excluir a CA.

Tratamento de exceções

Um CA privada da AWS comando pode falhar por vários motivos. Para obter informações sobre cada exceção e recomendações para resolvê-las, consulte a tabela abaixo.

CA privada da AWS Exceções

Exceção retornada por CA privada da AWS	Descrição	Correção
`AccessDeniedException`	As permissões necessárias para usar o comando fornecido não foram delegadas por uma CA privada à conta da chamada.	Para obter informações sobre como delegar permissões em CA privada da AWS, consulteAtribuir permissões de renovação de certificado ao ACM.
`InvalidArgsException`	Uma solicitação de criação ou renovação de certificado foi feita com parâmetros inválidos.	Consulte a documentação individual do comando para verificar se os parâmetros de entrada são válidos. Se estiver criando um novo certificado, verifique se o algoritmo de assinatura solicitado pode ser usado com o tipo de chave da CA.

Exceção retornada por CA privada da AWS	Descrição	Correção
`InvalidStateException`	A CA privada associada não pode renovar o certificado porque não ele está no estado `ACTIVE`.	Tente restaurar a CA privada. Se a CA privada estiver fora do período de restauração, a CA não poderá ser restaurada e o certificado não poderá ser renovado.
`LimitExceededException`	Cada autoridade de certificação (CA) tem uma cota de certificados que pode emitir. A CA privada associada ao certificado designado atingiu sua cota. Para obter mais informações, consulte Service Quotas, no Guia do Referência geral da AWS .	Entre em contato com o AWS Support Centro para solicitar um aumento de cota.
`MalformedCSRException`	A solicitação de assinatura de certificado (CSR) que foi enviada ao CA privada da AWS não pode ser verificada ou validada.	Confirme se sua CSR foi gerada e configurada corretamente.
`OtherException`	Um erro interno fez com que a solicitação falhasse.	Tente executar o comando novamente. Se o problema persistir, entre em contato com o AWS Support Centro.
`RequestFailedException`	Um problema de rede em seu AWS ambiente fez com que a solicitação falhasse.	Repetir a solicitação . Se a falha continuar, verifique a configuração da Amazon VPC (VPC).

Exceção retornada por CA privada da AWS	Descrição	Correção
`ResourceNotFoundException`	A CA privada que emitiu o certificado foi eliminada e não existe mais.	Solicite um novo certificado de outra CA ativa.
`ThrottlingException`	Uma ação de API solicitada falhou porque excedeu uma cota.	Confirme se você não está emitindo mais chamadas do que o permitido pelo CA privada da AWS. Um erro de `ThrottlingException` também pode ocorrer porque você encontrou uma condição transitória em vez de uma cota excedida. Ao encontrar o erro, se você não estiver fazendo chamadas acima da cota, tente a solicitação novamente. Se estiver enfrentando um limite de cota, poderá solicitar um aumento. Para obter mais informações, consulte Service Quotas no Referência geral da AWS Guia.

Exceção retornada por CA privada da AWS	Descrição	Correção
`ValidationException`	Os parâmetros de entrada da solicitação foram formatados incorretamente ou o período de validade do certificado raiz termina antes do período de validade do certificado solicitado.	Verifique os requisitos de sintaxe dos parâmetros de entrada do comando, bem como o período de validade do certificado raiz da CA. Para obter informações sobre como alterar o período de validade, consulte Atualizar a CA privada.

Usar o padrão Matter

O padrão de conectividade Matter especifica configurações de certificado que melhoram a segurança e a consistência de dispositivos com Internet das Coisas (IoT). Exemplos Java para criar certificados de CA raiz, de CA intermediária e de entidade final compatíveis com Matter podem ser encontrados em Usando a API do CA privada da AWS para implementar o padrão Matter (exemplos Java).

Para ajudar na solução de problemas, os desenvolvedores do Matter fornecem uma ferramenta de verificação de certificados que se chama chip-cert. Os erros relatados po essa ferramenta estão listados na tabela a seguir, com as respectivas correções.

Código de erro	Significado	Correção
0x00000035	As extensões `BasicConstraints` , `KeyUsage` e `ExtensionKeyUsage` devem ser marcadas como críticas.	Certifique-se de selecionar o modelo correto para o seu caso d
0x00000000	A extensão do identificador de chave da autoridade deve estar presente.	CA privada da AWS não define a extensão do identificador da c autoridade nos certificados raiz. Você deve gerar um AuthorityK fier valor codificado em Base64 usando o CSR e, em seguida, p por um. CustomExtension Para obter mais informações, consul

Código de erro	Significado	Correção
		uma CA raiz para certificados operacionais de nós (NOC). e Ati Autoridade de Atestado de Produto (PAA).
0x000000E	O certificado está expirado.	Certifique-se de que o certificado em uso não esteja expirado.

Código de erro	Significado	Correção

0x000000 4	Falha na validação da cadeia de certificados.	Esse erro pode ser encontrado se você tentar criar um certifica entidade final compatível com o Matter sem usar os exemplos J fornecidos, que usam a CA privada da AWS API para passar ur certificado configurado corretamente. KeyUsage
		
		Por padrão, CA privada da AWS gera valores de KeyUsage ext nove bits, com o nono bit resultando em um byte extra. O Matte esse byte extra durante as conversões de formato, causando fa validação da cadeia. No entanto, a CustomExtensionno `APIPas rough` modelo pode ser usado para definir o número exato de no `KeyUsage` valor. Para ver um exemplo, consulte Crie um ce operacional de nó (NOC).
		
		Se você modificar o código de amostra ou usar um utilitário X.5 alternativo, como o OpenSSL, precisará realizar a verificação m para evitar erros de validação de cadeia.
		
		Para verificar se as conversões ocorrem sem perdas
		
		1. Use openssl para verificar se um certificado de nó (entidad contém uma cadeia válida. Nesse exemplo, `rcac.pem` é o do de CA raiz, `icac.pem` é o certificado de CA intermediár `noc.pem` é o certificado de nó.
		
		`openssl verify -verbose -CAfile <(cat rcac.pem icac noc.pem`
		
		2. Use chip-cert para converter o certificado de nó formatado no formato TLV (etiqueta, comprimento, valor) e vice-versa
		
		`./chip-cert convert-cert noc.pem noc.chip -c`
		`./chip-cert convert-cert noc.chip noc_converted.pem`

Código de erro	Significado	Correção
		Os arquivos `noc.pem` e `noc_converted.pem` devem se exatamente os mesmos confirmados por uma ferramenta d comparação de strings.

Conector para erros e falhas do AD

Use as informações aqui para ajudá-lo a diagnosticar e corrigir erros e falhas de criação ao trabalhar com o Connector for AD.

Tópicos

Erros

O Connector for AD envia mensagens de erro por vários motivos. Para obter informações sobre cada erro e recomendações sobre como resolvê-los, consulte a tabela a seguir. Você pode receber esses erros assinando os eventos do Amazon EventBridge Scheduler (fonte do evento:`aws.pca-connector-ad`) ou usando a inscrição manual no Windows.

Código de erro	Significado	Correção
0x8FFFA000	Falha na autenticação Kerberos	Se você estiver usando inscrição automática, corrija a entidade principal do serviço de recursos da AWS . Se estiver usando a UI do Active Directory para obter um certificado, execute `gpupdate /force`.

Código de erro	Significado	Correção
0x8FFFA001	A mensagem SOAP deve conter um cabeçalho de ação.	Adicione um cabeçalho de ação.
0x8FFFA002	O conector não tem acesso à CA privada à qual está conectado.	Compartilhe sua CA privada com o conector criando um AWS Resou rce Access Manager (RAM) para compartilhar entre sua CA privada e o serviço Connector para AD.
0x8FFFA003	A CA privada para esse conector não está ativa.	Coloque a CA privada para no estado Ativo. Se a CA privada estiver no estado de certificado pendente, instale o certificado de CA.
0x8FFFA004	A CA privada para esse conector não existe.	Coloque a autoridade de certifica ção no estado Ativo se ela estiver no estado Excluído. Se a CA privada for excluída permanentemente, crie um novo conector com uma CA diferente .
0x8FFFA005	O modelo especificou o atributo `directoryGuid` para o requerent e do certificado ou o nome alternati vo do requerente, mas o atributo não foi encontrado no objeto do AD do solicitante.	O Active Directory não gerou um `directoryGuid` para o diretório . Solucionar problemas no Active Directory.

Código de erro	Significado	Correção
0x8FFFA006	O modelo especificou o atributo `dnsHostName` para o requerente do certificado ou o nome alternativo do requerente, mas o atributo não foi encontrado no objeto do AD do solicitante.	Adicione o atributo `dnsHostName` ao seu objeto do AD.
0x8FFFA007	O modelo especificou o atributo de e-mail a ser incluído no requerente do certificado ou no nome alternativo do requerente, mas esse atributo não foi encontrado no objeto do AD do solicitante.	Adicione o atributo email ao seu objeto do AD
0x8FFFA008	A mensagem SOAP deve ter um cabeçalho de ação `http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy/IPolicy/GetPolicies` ou `http://schemas.microsoft.com/windows/pki/2009/01/enrollment/RST/wstep` .	Atualize o cabeçalho da ação para usar um dos valores especificados.
0x8FFFA009	O BinarySecurityToken deve ser codificado em`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary` .	Atualize o tipo de token de segurança binário.

Código de erro	Significado	Correção
0x8FFFA00A	O BinarySecurityToken é inválido.	Verifique se a CSR foi gerada corretamente.
0x8FFFA00B	O BinarySecurityToken deve ter um tipo de valor de `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#PKCS7 ouhttp://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10` .	Atualize o tipo de valor do token de segurança binário para um valor válido.
0x8FFFA00C	O CMS inválido BinarySecurityToken contido.	O Base64 é válido, mas a sintaxe da mensagem criptográfica (CMS) é inválida. Revise a sintaxe CMS.
0x8FFFA00D	O BinarySecurityToken continha uma CSR inválida.	Verifique se a CSR foi gerada corretamente.
0x8FFFA00E	A CA privada não pôde emitir um certificado usando o modelo específico.	Revise a exceção de validação de AWS Private CA. Você pode ver a exceção de validação na Amazon EventBridge ou AWS CloudTrail.
0x8FFFA00F	A mensagem SOAP deve ter o tipo de solicitação de `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .	Defina o tipo de solicitação como `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .

Código de erro	Significado	Correção
0x8FFFA010	A mensagem SOAP deve ter um cabeçalho “to” do campo `CertificateEnrollmentPolicyServerEndpoint` do conector ou do campo de URI na resposta XCEP.	Defina o cabeçalho do token de segurança da solicitação como o campo `CertificateEnrollmentPolicyServerEndpoint` ou o campo de URI na resposta XCEP.
0x8FFFA011	A mensagem SOAP deve ter apenas um cabeçalho de ação.	Revise o cabeçalho da mensagem SOAP do token de segurança da solicitação e defina-o corretamente.
0x8FFFA012	A mensagem SOAP deve ter apenas um cabeçalho `messageId` .	Revise o cabeçalho da mensagem SOAP do token de segurança da solicitação e defina-o corretamente.
0x8FFFA013	A mensagem SOAP deve ter apenas um cabeçalho “to”.	Revise o cabeçalho da mensagem SOAP do token de segurança da solicitação e defina-o corretamente.
0x8FFFA014	O solicitante não tem acesso ao modelo solicitado.	Permita que o grupo do solicitante se inscreva usando o modelo solicitado, criando uma entrada de controle de acesso.
0x8FFFA015	A extensão `CertificateTemplateInformation` ou a `CertificateTemplateName` extensão devem estar presentes no BinarySecurityToken.	Adicione a extensão de segurança à sua CSR.

Código de erro	Significado	Correção
0x8FFFA016	O modelo solicitado não foi encontrado para o conector especificado.	Modelos são recursos secundários de cada conector. Crie o modelo do conector usando `createTemplate` .
0x8FFFA017	A solicitação foi negada devido ao controle de utilização da solicitação.	Reduza a taxa de solicitações.
0x8FFFA018	A mensagem SOAP deve conter um cabeçalho `to`.	Analise o cabeçalho da mensagem SOAP.
0x8FFFA019	Não foi possível processar a mensagem SOAP devido a um cabeçalho não reconhecido.	Analise o cabeçalho da mensagem SOAP.
0x8FFFA01A	O modelo especificou o atributo de UPN a ser incluído no requerente do certificado ou no nome alternativo do requerente, mas esse atributo não foi encontrado no objeto do AD do solicitante.	Adicione um UPN ao objeto do Active Directory.

Falhas na criação do conector

A criação do conector pode falhar por vários motivos. Quando a criação do conector falhar, você receberá o motivo da falha na resposta da API. Se você estiver usando o console, o motivo da falha será exibido na página de detalhes do conector, no campo Detalhes adicionais do status, no contêiner de detalhes do conector. A tabela a seguir descreve os motivos da falha e as etapas recomendadas para resolução.

Status da falha	Descrição	Correção
`CA_CERTIFICATE_REGISTRATION_FAILED`	O Connector for AD não consegue importar certificados CA para o seu diretório.	Consulte a página de pré-requisitos e verifique se sua conta de serviço tem as permissões corretas. Depois de delegar as permissões corretas à sua conta de serviço, exclua o conector com falha e crie um novo. Para obter informações sobre delegação de permissões, consulte Delegar privilégios à sua conta de serviço no AWS Directory Service Guia de Administração.
`DIRECTORY_ACCESS_DENIED`	O conector do AD não consegue acessar seu diretório.	Você deve conceder ao Connector for AD acesso ao seu diretório. Revise a Política do IAM seção para verificar se a política do IAM associada à sua AWS conta permite acessar e descrever diretórios. Depois de conceder as permissões corretas para sua AWS função, exclua o conector com falha e crie um novo.
`INTERNAL_FAILURE`	O conector para AD sofreu uma falha interna.	Tente novamente mais tarde. Exclua o conector com falha e crie um novo.

Status da falha	Descrição	Correção
`PRIVATECA_ACCESS_DENIED`	O Connector for AD não consegue acessar sua CA privada.	Revise a página de pré-requisitos e verifique se você tem as permissões para criar um conector. Para mais informações, consulte Política do IAM. Se você estiver criando um conector por meio AWS CLI de nossa API, consulte a página de pré-requisitos e verifique se você compartilhou a CA privada com o Connector for AD usando. AWS Resource Access Manager Depois de verificar e corrigir as permissões do IAM e o compartilhamento de AWS RAM recursos, exclua o conector com falha e crie um novo.
`PRIVATECA_RESOURCE_NOT_FOUND`	O Connector for AD não consegue encontrar a CA privada especificada.	Certifique-se de especificar o CA Amazon Resource Name (ARN) privado correto, depois exclua o conector com falha e crie um novo usando o ARN privado pretendido da CA.

Status da falha	Descrição	Correção
SECURITY_GROUP_NOT_IN_VPC	O grupo de segurança não está na VPC que hospeda seu diretório.	Use um grupo de segurança que esteja na VPC que hospeda seu diretório. Para ter mais informações, consulte Grupos de segurança. Exclua o conector com falha e crie um novo com um grupo de segurança que esteja na VPC.
VPC_ACCESS_DENIED	O Connector for AD não pode acessar a Amazon VPC que hospeda seu diretório.	Confira as suas permissões do IAM. Exclua o conector com falha e crie um novo. Para ver um exemplo de política do IAM que inclui permissões de acesso, consulte Política do IAM
VPC_ENDPOINT_LIMIT_EXCEEDED	O Connector for AD não pode criar um endpoint em sua Amazon VPC. Você atingiu o limite de VPC endpoints que você pode criar para sua conta.	Exclua endpoints da Amazon VPC ou solicite um aumento de limite. Depois de concluir uma das duas etapas, exclua o conector com falha e crie um novo. Para obter informações sobre cotas, consulte Cotas do Amazon Virtual Private Cloud Service.

Status da falha	Descrição	Correção
VPC_RESOURCE_NOT_FOUND	O Connector for AD não consegue encontrar a VPC especificada.	Verifique se você especific ou a VPC correta e se a VPC existe. Em seguida, exclua o conector com falha e crie um novo usando o ID VPC correto.

Falhas na criação do SPN

A criação do nome principal do serviço (SPN) pode falhar por vários motivos. Quando a criação do SPN falhar, você receberá o motivo da falha na resposta da API. Se você estiver usando o console, o motivo da falha será exibido na página de detalhes do conector, no campo Detalhes adicionais do status, no contêiner do nome principal do serviço (SPN). A tabela a seguir descreve os motivos da falha e as etapas recomendadas para resolução.

Status da falha	Descrição	Correção
DIRECTORY_ACCESS_DENIED	O Connector for AD não consegue acessar seu diretório.	Conceda ao Connector for AD acesso ao seu diretório. Para ver um exemplo de política do IAM que inclui permissões que concedem acesso ao diretório, consultePolítica do IAM.
DIRECTORY_NOT_REACHABLE	O Connector for AD não consegue acessar seu diretório.	Verifique a rede entre AWS e seu diretório e tente criar um SPN novamente.
DIRECTORY_RESOURCE_NOT_FOUND	O conector para AD não consegue encontrar o diretório especificado.	Certifique-se de especificar a ID de diretório correta, depois exclua o conector com falha e

Status da falha	Descrição	Correção
		crie um novo usando a ID de diretório pretendida.
`INTERNAL_FAILURE`	O conector para AD sofreu uma falha interna.	Tente novamente mais tarde.
`SPN_EXISTS_ON_DIFFERENT_AD_OBJECT`	O nome principal do serviço (SPN) existe em um objeto diferente do Active Directory.	Exclua o SPN do objeto do Active Directory e tente criar o SPN novamente.
`SPN_LIMIT_EXCEEDED`	O Connector for AD não pode criar o SPN porque você atingiu o limite de SPNs por diretório. O número máximo de SPNs por diretório é 10.	Exclua um ou mais SPNs da sua conta e tente criar o SPN novamente.

Erros de falha na criação do conector AD

O conector para a criação do conector AD pode falhar por vários motivos. Quando a criação do conector falhar, você receberá o motivo da falha na resposta da API. Se você estiver usando o console, o motivo da falha será exibido na página de detalhes do conector, no campo Detalhes adicionais do status, no contêiner Detalhes do conector. A tabela a seguir descreve os motivos da falha e as etapas recomendadas para resolução.

Termos e conceitos

Os seguintes termos e conceitos podem ajudá-lo conforme você trabalha com o AWS Private Certificate Authority (CA privada da AWS).

Tópicos

Confiança

Para que um navegador da web confie na identidade de um site, o navegador deve ser capaz de verificar o certificado do site. Os navegadores, no entanto, confiam em apenas um pequeno número de certificados conhecidos como certificados CA raiz. Um terceiro confiável, conhecido como uma autoridade certificadora (CA), valida a identidade do site e emite um certificado digital assinado para o operador do site. O navegador pode, então, verificar a assinatura digital para validar a identidade do site. Se a validação for bem-sucedida, o navegador exibe um ícone de cadeado na barra de endereços.

Certificados do servidor TLS

As transações HTTPS exigem certificados de servidor para autenticar um servidor. Um certificado de servidor é uma estrutura de dados X.509 v3 que vincula a chave pública no certificado ao assunto

do certificado. Um certificado TLS é assinado por uma autoridade de certificação (CA). Ele contém o nome do servidor, o período de validade, a chave pública, o algoritmo de assinatura e muito mais.

Assinatura de certificado

Uma assinatura digital é um hash criptografado sobre um certificado. Uma assinatura é usada para confirmar a integridade dos dados do certificado. A CA privada cria uma assinatura usando uma função de hash criptográfico, como SHA256 sobre o conteúdo de certificado de tamanho variável. Esta função hash produz uma cadeia de dados de tamanho fixo que não pode ser falsificada. Essa string é chamada de hash. Em seguida, a CA criptografa o valor do hash com sua chave privada e concatena o hash criptografado com o certificado.

Autoridade certificadora

Uma autoridade de certificado (CA) emite e, se necessário, revoga certificados digitais. O tipo mais comum de certificado é baseado no padrão ISO X.509. Um certificado X.509 afirma a identidade do objeto do certificado e vincula essa identidade a uma chave pública. O objeto pode ser um usuário, um aplicativo, um computador ou outro dispositivo. A CA assina um certificado aplicando hash ao conteúdo e criptografando o hash com a chave privada relacionada à chave pública no certificado. Um aplicativo cliente, como um navegador da Web que precisa confirmar a identidade de um objeto, usa a chave pública para descriptografar a assinatura do certificado. Em seguida, ele aplica hash ao conteúdo do certificado e compara o valor de hash com a assinatura descriptografada para determinar se correspondem. Para obter mais informações sobre assinatura de certificado, consulte Assinatura de certificado.

É possível usar o CA privada da AWS para criar uma CA privada e usar a CA privada para emitir certificados. Sua CA privada emite apenas certificados SSL/TLS privados para uso em sua organização. Para ter mais informações, consulte Certificado privado. Sua CA privada também requer um certificado para poder ser usada. Para ter mais informações, consulte Certificado CA.

CA raiz

Um bloco de criação criptográfica e raiz de confiança em que os certificados podem ser emitidos. Ela é composta de uma chave privada para assinar (emitir) certificados e um certificado raiz que identifica a CA raiz e vincula a chave privada ao nome da CA. O certificado raiz é distribuído aos armazenamentos de confiança de cada entidade em um ambiente. Os administradores constroem armazenamentos de confiança para incluir apenas as CAs em que confiam. Os administradores

atualizam ou criam os armazenamentos de confiança nos sistemas operacionais, instâncias e imagens de máquina host de entidades em seu ambiente. Quando os recursos tentam se conectar uns com os outros, eles verificam os certificados apresentados por cada entidade. Um cliente verifica a validade dos certificados e se existe uma cadeia do certificado a um certificado raiz instalada no armazenamento de confiança. Se essas condições forem atendidas, um handshake é realizado entre os recursos. Este handshake comprova criptograficamente a identidade de cada entidade para a outra e cria um canal de comunicação criptografado (TLS/SSL) entre elas.

Certificado CA

Um certificado de autoridade de certificado (CA) afirma a identidade da CA e a vincula à chave pública contida no certificado.

Você pode usar o CA privada da AWS para criar uma CA raiz privada ou uma CA subordinada privada, cada uma com o suporte de um certificado CA. Os certificados CA subordinados são assinados por outro certificado CA superior em uma cadeia de confiança. Mas, no caso de uma CA raiz, o certificado é autoassinado. Você também pode estabelecer uma autoridade raiz externa (hospedada on-premises, por exemplo). Depois, você pode usar sua autoridade raiz para assinar um certificado CA raiz subordinado hospedado pelo CA privada da AWS.

O exemplo a seguir mostra os campos típicos contidos em um certificado CA X.509 do CA privada da AWS. Observe que para um certificado de CA, o valor `CA:` no campo `Basic Constraints` é definido como `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 CA raiz

Uma autoridade certificadora (CA) normalmente existe em uma estrutura hierárquica que contém várias outras CAs com relacionamentos pai-filho claramente definidos entre eles. CAs subordinadas (ou filhas) são certificadas pelas CAs de seus pais, criando uma cadeia de certificados. A CA no alto da hierarquia é chamada de CA raiz, e seu certificado é chamado de certificado raiz. Este certificado é geralmente autoassinado.

Certificado de entidade final

Um certificado de entidade final identifica um recurso, como um servidor, instância, contêiner ou dispositivo. Ao contrário dos certificados de CA, os certificados de entidade final não podem ser usados para emitir certificados. Outros termos comuns para certificado de entidade final são certificado “cliente” ou “folha”.

Certificados autoassinados

Um certificado assinado pelo emissor em vez de uma CA superior. Ao contrário dos certificados emitidos de uma raiz segura, mantida por uma CA, os certificados autoassinados atuam como sua própria raiz. Como resultado, eles apresentam limitações significativas: podem ser usados para criptografar comunicações, mas não verificam a identidade nem podem ser revogados. Do ponto de vista da segurança, esses certificados não são aceitáveis Mas, todavia, são usados pelas organizações porque podem ser gerados facilmente, não exigem especialização nem infraestrutura e são aceitos por vários aplicativos. Não há controles implementados para emitir certificados

autoassinados. As organizações que usam esses certificados correm maior risco de interrupções causadas por expirações de certificados porque não contam com uma forma de controlar as datas de expiração.

Certificado privado

Os certificados do CA privada da AWS são certificados SSL/TLS privados que podem ser usados em sua organização, mas não são confiáveis na Internet pública. Use-os para identificar recursos, como clientes, servidores, aplicativos, serviços, dispositivos e usuários. Ao estabelecer um canal de comunicações criptografadas seguras, cada recurso usa um certificado como o seguinte, bem como técnicas criptográficas para provar sua identidade para outro recurso. Endpoints de API internos, servidores da Web, usuários de VPN, dispositivos IoT e muitos outros aplicativos usam certificados privados para estabelecer canais de comunicação criptografados que são necessários para sua operação segura. Por padrão, os certificados privados não são publicamente confiáveis. Um administrador interno deve configurar explicitamente aplicativos para confiar em certificados privados e distribuir os 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
```

Caminho do certificado

Um cliente que depende de um certificado valida que existe um caminho do certificado da entidade final, possivelmente por meio de uma cadeia de certificados intermediários, até uma raiz confiável. O cliente verifica se cada certificado ao longo do caminho é válido (não revogado). Ele também verifica se o certificado de entidade final não expirou, é íntegro (não foi adulterado ou modificado) e se as restrições no certificado estão impostas.

Restrição de comprimento de caminho

As restrições básicas de um certificado pathLenConstraintde CA definem o número de certificados de CA subordinados que podem existir na cadeia abaixo dele. Por exemplo, um certificado de CA com uma restrição de comprimento de caminho igual a zero não pode ter uma CA subordinada. Uma CA com uma restrição de comprimento de caminho igual a um pode ter até um nível de CAs subordinadas abaixo dela. O RFC 5280 define isso como “o número máximo de certificados non-self-issued intermediários que podem seguir esse certificado em um caminho de certificação válido”. O valor do comprimento do caminho exclui o certificado de entidade final, embora uma linguagem informal sobre o “comprimento” ou a “profundidade” de uma cadeia de validação possa incluí-lo, causando confusão.

Histórico do documento

A tabela a seguir descreve alterações significativas nesta documentação desde janeiro de 2018. Além das principais alterações listadas aqui, também atualizamos a documentação com frequência para melhorar as descrições e os exemplos e abordar os comentários que você nos envia. Para ser notificado sobre alterações significativas, use o link no canto superior direito para assinar os feeds RSS.

Alteração	Descrição	Data
AWS Private CA agora suporta Connector for SCEP (Preview)	Use o Connector for SCEP para se conectar AWS Private CA aos seus clientes e dispositivos habilitados para SCEP.	11 de junho de 2024
Guia de solução de problemas do novo conector	Foram adicionadas duas novas seções sobre solução de problemas de falhas na criação de conectores e SPN.	4 de abril de 2024
Adicionando extensão CDP para Matter	Adiciona suporte à extensão CDP (Certificate Revocation List Distribution Point) para Matter.	25 de janeiro de 2024
AWS Private CA Suporte de API para mDL	Foi adicionado suporte à API para criar certificados em conformidade com o padrão ISO/IEC para carteira de motorista móvel (mDL).	16 de janeiro de 2024
AWS Private CA Conector para Active Directory	Suporte via Guia do usuário, API e CLI para o Connector para AD. Para obter mais informações, consulte	24 de agosto de 2023

	Connector para AD na documentação.	
Alteração de nomes de políticas de segurança para que correspondam ao novo nome do serviço	Adoção de novos nomes para políticas AWS gerenciadas do IAM que especificam permissões padrão em CA privada da AWS. Para obter mais informações, consulte Políticas gerenciadas do AWS.	13 de fevereiro de 2023
Adicionando um rastreador de alterações para políticas AWS gerenciadas	Documentação adicionada para rastrear alterações nas políticas AWS gerenciadas do IAM que especificam permissões padrão em CA privada da AWS. Para ter mais informações, consulte Atualizações em políticas gerenciadas pela AWS para o CA privada da AWS.	11 de novembro de 2022
Suporte via API e CLI para CAs que emitem certificados de curta duração	Com a introdução de modos de usos de CA, uma CA pode ser configurada para emitir certificados de uso geral ou exclusivamente de curta duração. Para ter mais informações, consulte Modos de autoridades de certificado.	24 de outubro de 2022

Renomeação do serviço e atualização do console	O serviço foi renomeado para AWS Private Certificate Authority (CA privada da AWS). O CA privada da AWS console recebe melhorias de usabilidade, incluindo painéis de ajuda integrados com links para a documentação completa.	27 de setembro de 2022
Suporte para certificados em conformidade com o Matter	Três novos modelos de certificado adicionam suporte a certificados de CA e de entidade final compatíveis com o Matter. Para obter mais informações, consulte Compreender modelos de certificado.	20 de julho de 2022
Suporte à nova região	Endpoint adicionado para Ásia-Pacífico (Jacarta). Para obter uma lista completa dos AWS Private CA endpoints , consulte Endpoints and Quotas da Autoridade de Certificação Privada do ACM.	4 de maio de 2022
Suporte para atributos e extensões personalizados	Use o CustomAttributeobjeto para configurar CAs e certificados personalizados e o CustomExtension objeto para configurar certificados personalizados.	16 de março de 2022

Suporte para OCSP gerenciado	Consulte Configurar um método de revogação de certificado para opções de revogação, incluindo o OCSP.	18 de agosto de 2021
Suporte para o recurso Bloqueio do acesso público do S3 para CRLs	Consulte Habilitar o recurso Bloqueio do acesso público do S3.	27 de maio de 2021
Exemplos de implementação Java novos e atualizados	Consulte Usar a API de CA privada do ACM (exemplos Java).	9 de setembro de 2020
Suporte à nova região	Endpoints adicionados para África (Cidade do Cabo) e Europa (Milão). Para obter uma lista completa de endpoints do CA privada da AWS , consulte Endpoints e cotas de autoridades de certificação privadas do AWS Certificate Manager.	27 de agosto de 2020
Acesso à CA privada entre contas com suporte	AWS Certificate Manager os usuários podem ser autorizados a emitir certificados usando CAs privadas que não são de sua propriedade. Para obter informações, consulte Acesso entre contas a CAs privadas.	17 de agosto de 2020

Suporte para VPC endpoints () PrivateLink	Foi adicionado suporte para o uso de VPC endpoints (AWS PrivateLink) para melhorar a segurança da rede. Para obter mais informações, consulte ACM Private CA VPC Endpoints AWS PrivateLink ().	26 de março de 2020
Seção de segurança dedicada adicionada	A documentação de segurança do AWS foi consolidada em uma seção de segurança dedicada. Para obter informações sobre segurança, consulte Segurança na Autoridade de Certificação AWS Certificate Manager Privada.	26 de março de 2020
ARN de modelo adicionado aos relatórios de auditoria.	Para obter mais informações, consulte Criar um relatório de auditoria para sua CA privada.	6 de março de 2020
CloudFormation apoio	Support adicionado para AWS CloudFormation. Para obter mais informações, consulte Referência de tipo de recurso da ACMPCA no Guia do usuário do AWS CloudFormation .	22 de janeiro de 2020

CloudWatch Integração de eventos	Integração com CloudWatch Eventos para eventos assíncronos, incluindo criação de CA, emissão de certificados e criação de CRL. Para obter mais informações, consulte Usando CloudWatch eventos.	23 de dezembro de 2019
Endpoints do FIPS	Endpoints FIPS adicionados para AWS GovCloud (Leste dos EUA) e AWS GovCloud (Oeste dos EUA). Para obter uma lista completa dos CA privada da AWS endpoints, consulte Endpoints and Quotas da Autoridade de Certificação AWS Certificate Manager Privada.	13 de dezembro de 2019
Permissões baseadas em tags	As permissões baseadas em tags compatíveis usando as novas APIs `TagResource`, `UntagResource` e `ListTagsForResource`. Para obter informações gerais sobre controles baseados em tags, consulte Controlar o acesso aos/para usuários e funções do IAM usando tags de recursos do IAM.	5 de novembro de 2019

Imposição de restrições de nome	Adição de suporte para impor restrições de nome de assunto em certificados CA importados. Para obter mais informações, consulte Impor restrições de nome em uma CA Privada.	28 de outubro de 2019
Novos modelos de certificado	Novos modelos de certificado adicionados, incluindo modelos para assinatura de código com AWS Signer. Para obter mais informações, consulte Usar modelos.	1 de outubro de 2019
Planejar sua CA	Adição de nova seção sobre como planejar sua PKI usando CA privada da AWS. Para obter mais informações, consulte Planejar sua implantação de CA privada do ACM.	30 de setembro de 2019
Adição de suporte à região	Foi adicionado suporte regional para a região AWS Ásia-Pacífico (Hong Kong). Para obter uma lista completa de regiões compatíveis, consulte Endpoints e cotas de autoridades de certificação privadas do AWS Certificate Manager Certificate Manager.	24 de julho de 2019
Adição de suporte completo à hierarquia de CA privada	O suporte para criar e hospedar CAs raiz remove a necessidade de um pai externo.	20 de junho de 2019

Adição de suporte à região	Foi adicionado suporte regional para as regiões AWS GovCloud (Oeste dos EUA e Leste dos EUA). Para obter uma lista completa de regiões compatíveis, consulte Endpoints e cotas de autoridades de certificação privadas do AWS Certificate Manager.	8 de maio de 2019
Adição de suporte à região	Foi adicionado suporte regional para as regiões AWS Ásia-Pacífico (Mumbai e Seul), Oeste dos EUA (Norte da Califórnia) e UE (Paris e Estocolmo). Para obter uma lista completa de regiões compatíveis, consulte Endpoints e cotas de autoridades de certificação privadas do AWS Certificate Manager Certificate Manager.	4 de abril de 2019
Testar fluxos de trabalho de renovação de certificados	Agora, os clientes podem testar manualmente a configuração do seu fluxo de trabalho de renovação gerenciada do ACM. Para obter mais informações, consulte Testar a configuração de renovação gerenciada do ACM.	14 de março de 2019

Adição de suporte à região	Foi adicionado suporte regional para a região AWS da UE (Londres). Para obter uma lista completa de regiões compatíveis, consulte Endpoints e cotas de autoridades de certificação privadas do AWS Certificate Manager.	1º de agosto de 2018
Restaurar CAs excluídas	A restauração da CA privada permite que os clientes restaurem autoridades de certificação (CAs) por até 30 dias após terem sido excluídas . Para obter mais informaçõ es, consulte Restaurar sua CA privada.	20 de junho de 2018

Atualizações anteriores

A tabela a seguir descreve o histórico de lançamento da documentação AWS Private Certificate Authority antes de junho de 2018.

Alteração	Descrição	Data
Novo guia	Essa versão apresenta o AWS Private Certificate Authority.	04 de abril de 2018

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.