aws

Guide de l’utilisateur

AWS Private Certificate Authority

Version latest

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Private Certificate Authority: Guide de l’utilisateur

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d’Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n’est pas d’Amazon, d’une manière susceptible de créer une confusion parmi les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce que c'est Autorité de certification privée AWS ?

Autorité de certification privée AWS permet de créer des hiérarchies d'autorités de certification (CA) privées, y compris des autorités de certification racine et subordonnées, sans les coûts d'investissement et de maintenance liés à l'exploitation d'une autorité de certification sur site. Vos autorités de certification privées peuvent émettre des certificats X.509 d'entité finale utiles dans les scénarios suivants :

- Création de canaux de communication TLS chiffrés
- Authentification d'utilisateurs, d'ordinateurs, de points de terminaison d'API et d'appareils IoT
- Signature de code par cryptographie
- Mise en œuvre du protocole OCSP (Online Certificate Status Protocol) pour obtenir le statut de révocation de certificats

Autorité de certification privée AWS les opérations sont accessibles depuis le AWS Management Console, à l'aide de l' Autorité de certification privée AWS API ou à l'aide du AWS CLI.

Rubriques

Quel est le service de certification le mieux adapté à mes besoins ?

Il existe deux AWS services permettant d'émettre et de déployer des certificats X.509. Choisissez celui qui correspond le mieux à vos besoins. Vous devez notamment déterminer si vous avez besoin de certificats publics ou privés, de certificats personnalisés, de certificats que vous souhaitez déployer dans d'autres AWS services ou d'une gestion et d'un renouvellement automatisés des certificats.

1. Autorité de certification privée AWS : ce service est destiné aux entreprises clientes qui construisent une infrastructure à clé publique (PKI) dans le cloud AWS . Il est également destiné à un usage privé au sein d'une organisation. Vous pouvez ainsi créer votre propre hiérarchie d'autorités de certification et émettre des certificats à l'aide de celle-ci pour authentifier les utilisateurs internes, les ordinateurs, les applications, les services, les serveurs et autres appareils, ainsi que pour signer le code informatique. Autorité de certification privée AWS Les certificats émis par une autorité de certification privée ne sont approuvés qu'au sein de votre organisation, et non sur Internet.

   Après avoir créé une autorité de certification privée, vous pouvez émettre des certificats directement (c'est-à-dire sans obtenir de validation auprès d'une autorité de certification tierce) et les personnaliser pour répondre aux besoins internes de votre organisation. Par exemple, vous pouvez :

   - Créer des certificats avec n'importe quel nom d'objet.
   - Créer des certificats avec n'importe quelle date d'expiration.
   - d'utiliser les algorithmes de clé privée et les longueurs de clé pris en charge ;
   - d'utiliser les algorithmes de signature pris en charge ;
   - Contrôler l'émission de certificats à l'aide de modèles.

   Vous êtes au bon endroit pour ce service. Pour commencer, connectez-vous à la console https://console.aws.amazon.com/acm-pca/.
2. AWS Certificate Manager (ACM) —Ce service gère les certificats pour les entreprises clientes qui ont besoin d'une présence Web sécurisée et publiquement fiable à l'aide du protocole TLS. Vous pouvez déployer des certificats ACM dans AWS Elastic Load Balancing, Amazon CloudFront, Amazon API Gateway et d'autres services intégrés. L'application la plus courante de ce type est un site web public sécurisé avec des exigences de trafic importantes.

   Avec ce service, vous pouvez utiliser des certificats publics fournis par ACM (certificats ACM) ou des certificats que vous importez dans ACM. Si vous créez une autorité Autorité de certification privée AWS de certification, ACM peut gérer l'émission de certificats à partir de cette autorité de certification privée et automatiser les renouvellements de certificats.

   Pour plus d'informations, consultez le Guide de l'utilisateur AWS Certificate Manager.

Régions

Comme la plupart AWS des ressources, les autorités de certification privées (CA) sont des ressources régionales. Pour utiliser des autorités de certification privées dans plusieurs régions, vous devez les créer dans ces régions. Vous ne pouvez pas copier les autorités de certification entre les régions. Consultez AWS Régions et points de terminaison dans la Références générales AWS ou le Tableau des régions AWS pour consulter la disponibilité régionale pour Autorité de certification privée AWS.

> **Note**
>
> ACM est actuellement disponible dans certaines régions, mais ce n' Autorité de certification privée AWS est pas le cas.

Services intégrés à AWS Private Certificate Authority

Si vous demandez AWS Certificate Manager un certificat privé, vous pouvez associer ce certificat à n'importe quel service intégré à ACM. Cela s'applique à la fois aux certificats liés à une Autorité de certification privée AWS racine et aux certificats liés à une racine externe. Pour plus d'informations, consultez la section Services intégrés dans le guide de AWS Certificate Manager l'utilisateur.

Vous pouvez également intégrer des autorités de certification privées dans Amazon Elastic Kubernetes Service pour délivrer des certificats au sein d'un cluster Kubernetes. Pour plus d'informations, consultez Sécuriser Kubernetes avec Autorité de certification privée AWS.

> **Note**
>
> Amazon Elastic Kubernetes Service n'est pas un service intégré ACM.

Si vous utilisez l' Autorité de certification privée AWS API, AWS CLI pour émettre un certificat ou pour exporter un certificat privé depuis ACM, vous pouvez installer le certificat où vous le souhaitez.

Algorithmes cryptographiques pris en charge

Autorité de certification privée AWS prend en charge les algorithmes cryptographiques suivants pour la génération de clés privées et la signature de certificats.

Algorithme supporté

Algorithmes à clé privée	Algorithmes de signature
RSA_2048	SHA256 AVEC ECDSA
RSA_4096	SHA384 AVEC ECDSA
EC_Prime 256v1	SHA512 AVEC ECDSA
EC_SECP384R1	SHA256WITHRSA SHA384WITHRSA
	SHA512WITHRSA

Cette liste s'applique uniquement aux certificats émis directement par le Autorité de certification privée AWS biais de sa console, de son API ou de sa ligne de commande. Lorsqu'il AWS Certificate Manager émet des certificats à l'aide d'une autorité de certification Autorité de certification privée AWS, celui-ci prend en charge certains de ces algorithmes, mais pas tous. Pour plus d'informations, consultez la section Demander un certificat privé dans le guide de AWS Certificate Manager l'utilisateur.

> **Note**
>
> Dans tous les cas, la famille d'algorithmes de signature spécifiée (RSA ou ECDSA) doit correspondre à la famille d'algorithmes de la clé privée de l'autorité de certification.

Quotas

Autorité de certification privée AWS attribue des quotas au nombre autorisé de certificats et d'autorités de certification. Les taux de demandes pour les actions d'API sont également soumis à des quotas. Autorité de certification privée AWS les quotas sont spécifiques à un AWS compte et à une région.

Autorité de certification privée AWS limite les demandes d'API à des taux différents en fonction du fonctionnement de l'API. La limitation signifie le Autorité de certification privée AWS rejet d'une demande par ailleurs valide parce que la demande dépasse le quota de l'opération pour le nombre de demandes par seconde. Lorsqu'une demande est limitée, Autorité de certification privée AWS

renvoie une ThrottlingExceptionerreur. Autorité de certification privée AWS ne garantit pas un taux de requêtes minimum pour les API.

Pour savoir quels quotas peuvent être ajustés, consultez le tableau Autorité de certification privée AWS des quotas dans le Références générales AWS.

Vous pouvez consulter vos quotas actuels et demander des augmentations de quotas à l'aide de AWS Service Quotas.

Pour consulter la up-to-date liste de vos Autorité de certification privée AWS quotas

1. Connectez-vous à votre AWS compte.
2. Ouvrez la console Service Quotas à l'adresse https://console.aws.amazon.com/servicequotas/.
3. Dans la liste Services, sélectionnez AWS Certificate Manager Private Certificate Authority (ACM PCA). Chaque quota de la liste des quotas de service indique la valeur de quota actuellement appliquée, la valeur de quota par défaut et indique si le quota est ajustable ou non. Choisissez le nom d'un quota pour plus d'informations à ce sujet.

Pour demander une augmentation de quota

1. Dans la liste des quotas de service, cliquez sur le bouton radio pour obtenir un quota ajustable.
2. Cliquez sur le bouton Demander une augmentation du quota.
3. Remplissez et soumettez le formulaire de demande d'augmentation du quota.

Autorité de certification privée AWS est intégré à AWS Certificate Manager. Vous pouvez utiliser la console ACM ou l'API ACM pour demander des certificats privés à une autorité de certification privée existante. AWS CLI Ces certificats PKI privés, qui sont gérés par ACM, sont soumis à la fois aux quotas PCA et aux quotas qu'ACM place sur les certificats publics et importés. Pour plus d'informations sur les exigences d'ACM, consultez la section Demander un certificat privé et des quotas dans le guide de l' AWS Certificate Manager utilisateur.

Conformité à la RFC

Autorité de certification privée AWS n'applique pas certaines contraintes définies dans la RFC 5280. La situation inverse est également vraie : certaines contraintes supplémentaires appropriées à une autorité de certification privée sont appliquées.

Appliqué

- Pas après la date. Conformément à la RFC 5280, Autorité de certification privée AWS empêche l'émission de certificats ayant une date `Not After` postérieure à la date `Not After` du certificat d'une autorité de certification émettrice.
- Contraintes de base. Autorité de certification privée AWS applique les contraintes de base et la longueur du chemin dans les certificats CA importés.

  Les contraintes basiques indiquent si la ressource identifiée par le certificat est une autorité de certification ou non et peut émettre des certificats. Les certificats d'une autorité de certification importés dans Autorité de certification privée AWS doivent inclure l'extension des contraintes basiques et l'extension doit être marquée `critical`. En plus du `critical` drapeau, `CA=true` il doit être installé. Autorité de certification privée AWS applique les contraintes de base en échouant avec une exception de validation pour les raisons suivantes :

  - L'extension n'est pas incluse dans le certificat d'une autorité de certification.
  - L'extension n'est pas marquée `critical`.

  La longueur du chemin (chemin LenConstraint) détermine le nombre de CA subordonnées qui peuvent exister en aval du certificat d'autorité de certification importé. Autorité de certification privée AWS applique la longueur du chemin en échouant avec une exception de validation pour les raisons suivantes :

  - L'importation d'un certificat d'une autorité de certification violerait la contrainte de longueur du chemin d'accès dans le certificat d'une autorité de certification ou dans tout certificat d'une autorité de certification de la chaîne.
  - L'émission d'un certificat violerait une contrainte de longueur de chemin d'accès.
- Les contraintes de nom indiquent un espace de nom dans lequel doivent figurer tous les noms de sujets figurant dans les certificats suivants d'un chemin de certification. Des restrictions s'appliquent au nom distinctif du sujet et aux noms alternatifs du sujet.

Non appliqué

- Politiques relatives aux certificats. Les politiques de certification régissent les conditions dans lesquelles une autorité de certification émet des certificats.
- Empêchez AnyPolicy. Utilisé dans les certificats délivrés aux autorités de certification.
- Nom alternatif de l'émetteur. Permet d'associer des identités supplémentaires à l'émetteur du certificat CA.

- Contraintes politiques. Ces contraintes limitent la capacité d'une autorité de certification à émettre des certificats d'une autorité de certification subordonnée.
- Mappages de politiques. Utilisé dans les certificats CA. Répertorie une ou plusieurs paires d'OID ; chaque paire inclut un issuerDomainPolicy et un sujetDomainPolicy.
- Attributs du répertoire des sujets. Utilisé pour transmettre les attributs d'identification du sujet.
- Accès aux informations sur le sujet. Comment accéder aux informations et aux services relatifs au sujet du certificat dans lequel apparaît l'extension.
- Identifiant de clé d'objet (SKI) et Identifiant de clé d'autorité (AKI). La RFC exige un certificat d'une autorité de certification pour contenir l'extension SKI. Les certificats émis par l'autorité de certification doivent contenir une extension AKI correspondant au SKI du certificat de l'autorité de certification. AWS ne fait pas appliquer ces exigences. Si votre certificat d'une autorité de certification ne contient pas de SKI, l'entité finale ou le certificat d'une autorité de certification subordonnée AKI sera à la place le hachage SHA-1 de la clé publique de l'auteur.
- SubjectPublicKeyInfoet nom alternatif du sujet (SAN). Lors de l'émission d'un certificat, Autorité de certification privée AWS copie les extensions SAN SubjectPublicKeyInfo et à partir du CSR fourni sans effectuer de validation.

Tarification

Un tarif mensuel est facturé à votre compte pour chaque autorité de certification privée à partir du moment où vous l'avez créée. Vous êtes également facturé pour chaque certificat que vous émettez. Ces frais incluent les certificats que vous exportez depuis ACM et les certificats que vous créez à partir de l' Autorité de certification privée AWS API ou de la Autorité de certification privée AWS CLI. Vous n'êtes pas facturé pour l'autorité de certification privée après sa suppression. Toutefois, si vous restaurez une autorité de certification privée, vous serez facturé pour la durée écoulée entre la suppression et la restauration. Les certificats privés pour lesquels vous ne disposez d'aucun accès à la clé privée sont gratuits. Il s'agit notamment des certificats utilisés avec des services intégrés tels que Elastic Load Balancing et API Gateway. CloudFront

Pour obtenir les dernières informations sur Autorité de certification privée AWS les prix, consultez la section AWS Private Certificate Authority Tarification. Vous pouvez également utiliser le calculateur de AWS prix pour estimer les coûts.

Sécurité dans AWS Private Certificate Authority

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cela comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS Private Certificate Authority, voir Services AWS Portée par programme de conformitéServices AWS .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de son utilisation Autorité de certification privée AWS. Les rubriques suivantes expliquent comment procéder à la configuration Autorité de certification privée AWS pour atteindre vos objectifs de sécurité et de conformité. Vous apprenez également à utiliser d'autres outils Services AWS qui vous aident à surveiller et à sécuriser vos Autorité de certification privée AWS ressources.

Rubriques

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

L'accès à Autorité de certification privée AWS nécessite des informations d'identification qui AWS peuvent être utilisées pour authentifier vos demandes. Les rubriques suivantes fournissent des informations sur la façon dont vous pouvez utiliser AWS Identity and Access Management (IAM) pour sécuriser vos autorités de certification privées en contrôlant qui peut y accéder.

Dans Autorité de certification privée AWS, la principale ressource avec laquelle vous travaillez est une autorité de certification (CA). Chaque autorité de certification que vous possédez ou contrôlez est identifiée par un Amazon Resource Name (ARN) de la forme suivante.

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

Le propriétaire d'une ressource est l'entité principale du AWS compte dans lequel une AWS ressource est créée. Les exemples suivants illustrent comment cela fonctionne.

- Si vous utilisez les informations d'identification de votre autorité de certification Utilisateur racine d'un compte AWS pour créer une autorité de certification privée, l'autorité de certification appartient à votre AWS compte.

  > **Important**
  >
  > - Nous vous déconseillons d'utiliser un Utilisateur racine d'un compte AWS pour créer des CA.
  > - Nous vous recommandons vivement d'utiliser l'authentification multifactorielle (MFA) à chaque fois que vous y accédez. Autorité de certification privée AWS

- Si vous créez un utilisateur IAM dans votre AWS compte, vous pouvez lui accorder l'autorisation de créer une autorité de certification privée. Cependant, le compte auquel appartient cet utilisateur possède l'autorité de certification.
- Si vous créez un rôle IAM dans votre AWS compte et que vous lui accordez l'autorisation de créer une autorité de certification privée, toute personne pouvant assumer ce rôle peut créer l'autorité de certification. Cependant, le compte auquel appartient le rôle possèdera l'autorité de certification privée.

Une permissions policy (politique d'autorisation) décrit qui a accès à quoi. La discussion suivante explique les options disponibles pour créer des stratégies d'autorisations.

> **Note**
>
> Cette documentation décrit l'utilisation d'IAM dans le contexte de Autorité de certification privée AWS. Elle ne fournit pas d'informations détaillées sur le service IAM. Pour une documentation IAM complète, consultez le Guide de l'utilisateur IAM. Pour plus d'informations sur la syntaxe des politiques IAM et pour obtenir des descriptions, consultez le AWS Guide de référence des politiques IAM.

Autorité de certification privée AWS Opérations et autorisations de l'API

Lorsque vous configurez des politiques de contrôle d'accès et d'autorisation que vous prévoyez d'associer à une identité IAM (politiques basées sur l'identité), utilisez le tableau suivant comme référence. La première colonne du tableau répertorie chaque opération Autorité de certification privée AWS d'API. Vous indiquez les actions dans l'élément `Action` d'une politique. Les autres colonnes fournissent les informations supplémentaires.

Autorité de certification privée AWS Opérations d'API	Autorisations nécessaires	Ressources
CreateCertificateAuthority	`acm-pca:CreateCertificateAuthority` `acm-pca:TagCertificateAuthority` (Nécessaire uniquement lors de la création d'une autorité de certification avec des balises.)	`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`

Autorité de certification privée AWS Opérations d'API	Autorisations nécessaires	Ressources
CreatePermission	acm-pca:CreatePermission	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DeleteCertificateAuthority	acm-pca:DeleteCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DeletePermission	acm-pca:DeletePermission	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DeletePolicy	acm-pca:DeletePolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DescribeCertificateAuthority	acm-pca:DescribeCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

Autorité de certification privée AWS Opérations d'API	Autorisations nécessaires	Ressources
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*

Autorité de certification privée AWS Opérations d'API	Autorisations nécessaires	Ressources
ImportCertificateAuthorityCertificate	acm-pca:ImportCertificateAuthorityCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
IssueCertificate	acm-pca:IssueCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListCertificateAuthorities	acm-pca:ListCertificateAuthorities	N/A
ListPermissions	acm-pca:ListPermissions	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListTags	acm-pca:ListTags	N/A
PutPolicy	acm-pca:PutPolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

Autorité de certification privée AWS Opérations d'API	Autorisations nécessaires	Ressources
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*

Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

  Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center .

- Utilisateurs gérés dans IAM par un fournisseur d'identité :

  Créez un rôle pour la fédération d'identité. Pour plus d'informations, voir la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) du Guide de l'utilisateur IAM.
- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM du Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

AWS politiques gérées

Autorité de certification privée AWS inclut un ensemble de politiques AWS gérées prédéfinies pour Autorité de certification privée AWS les administrateurs, les utilisateurs et les auditeurs. La présentation de ces stratégies peut vous aider à mettre en œuvre Politiques gérées par le client.

Choisissez l'une des politiques répertoriées ci-dessous pour voir les détails et un exemple de code de politique.

AWSPrivateCAFullAccess

Accorde un contrôle administratif illimité.

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

AWSPrivateCAReadOnly

Accorde un accès limité aux opérations d'API en lecture seule.

```
{
   "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

Permet d'émettre et de révoquer des certificats CA. Cette stratégie n'a pas d'autres capacités administratives et n'est pas en mesure d'émettre des certificats d'entité finale. Les autorisations sont mutuellement exclusives avec la stratégie User.

```
{
   "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

Accorder la capacité d'émettre et de révoquer des certificats d'entité finale. Cette stratégie n'a pas de capacités administratives et aucune possibilité d'émettre des certificats d'une autorité de certification. Les autorisations s'excluent mutuellement de la PrivilegedUserpolitique.

```
{
   "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

Accordez l'accès aux opérations d'API en lecture seule et l'autorisation de générer un rapport d'audit 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":"*"
      }
   ]
}
```

Mises à jour des politiques AWS gérées pour Autorité de certification privée AWS

Dans le tableau suivant, consultez les détails des mises à jour des politiques AWS gérées Autorité de certification privée AWS depuis que le service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant toutes les modifications apportées Autorité de certification privée AWS, abonnez-vous au flux RSS de la Historique du document page.

Changements de politique gérés

Modification	Description	Date
Nouveaux noms de politiques : • `AWSPrivateCAFullAccess` • `AWSPrivateCAReadOnly` • `AWSPrivateCAPrivilegedUser` • `AWSPrivateCAAuditor` • `AWSPrivateCAUser`	Les préfixes du nom de la politique ont été modifiés de `AWSCertificateManagerPrivateCA` à. `AWSPrivateCA` Les fonctionnalités restent inchangées.	13 février 2023

Politiques gérées par le client

Il est recommandé de ne pas utiliser votre Utilisateur racine d'un compte AWS pour interagir avec AWS, y compris Autorité de certification privée AWS. Utilisez plutôt AWS Identity and Access Management (IAM) pour créer un utilisateur IAM, un rôle IAM ou un utilisateur fédéré. Créez un groupe d'administrateurs et ajoutez-vous au groupe. Puis, connectez-vous en tant qu'administrateur. Si nécessaire, ajoutez des utilisateurs supplémentaires au groupe.

Une autre bonne pratique consiste à créer une politique IAM gérée par le client que vous pouvez attribuer aux utilisateurs. Les stratégies gérées par le client sont des stratégies autonomes basées sur l'identité, que vous créez et que vous pouvez associer à plusieurs utilisateurs, groupes ou rôles dans votre compte AWS . Une telle politique limite les utilisateurs à effectuer uniquement les Autorité de certification privée AWS actions que vous spécifiez.

Dans l'exemple suivant, la stratégie gérée par le client permet à un utilisateur de créer un rapport d'audit d'une autorité de certification. Il s'agit uniquement d'un exemple. Vous pouvez choisir les Autorité de certification privée AWS opérations que vous souhaitez. Pour obtenir plus d'exemples, consultez Politiques en ligne.

Pour créer une stratégie gérée par le client

1. Connectez-vous à la console IAM à l'aide des informations d'identification d'un AWS administrateur.

2. Dans le volet de navigation de la console, choisissez Stratégies.
3. Sélectionnez Créer une politique.
4. Choisissez l'onglet JSON.
5. Copiez la stratégie suivante et collez-la dans l'éditeur.

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

6. Choisissez Examiner une politique.
7. Pour Name (Nom), tapez `PcaListPolicy`.
8. (Facultatif) Tapez une description.
9. Choisissez Créer une politique.

Un administrateur peut associer la politique à n'importe quel utilisateur IAM afin de limiter Autorité de certification privée AWS les actions que l'utilisateur peut effectuer. Pour savoir comment appliquer une politique d'autorisations, consultez la section Modification des autorisations pour un utilisateur IAM dans le guide de l'utilisateur IAM.

Politiques en ligne

Les stratégies en ligne sont des stratégies que vous créez, gérez et intégrez directement à un utilisateur, un groupe ou un rôle. Les exemples de politique suivants montrent comment attribuer des autorisations pour effectuer des Autorité de certification privée AWS actions. Pour obtenir des informations générales sur les politiques intégrées, consultez la section Utilisation des politiques intégrées dans le guide de l'utilisateur IAM. Vous pouvez utiliser l'API AWS Management Console, le AWS Command Line Interface (AWS CLI) ou l'API IAM pour créer et intégrer des politiques intégrées.

> **Important**
>
> Nous vous recommandons vivement d'utiliser l'authentification multifactorielle (MFA) à chaque fois que vous y accédez. Autorité de certification privée AWS

Rubriques

Liste des autorités de certification privées

La stratégie suivante permet à un utilisateur d'établir la liste de toutes les autorités de certification privées figurant dans un compte.

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

Récupération d'un certificat CA privé

La stratégie suivante permet à un utilisateur de récupérer un certificat d'une autorité de certification privée spécifique.

```
{
   "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"
   }
}
```

Importation d'un certificat CA privé

La stratégie suivante permet à un utilisateur d'importer un certificat d'une autorité de certification privée.

```
{
   "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"
   }
}
```

Supprimer une autorité de certification privée

La stratégie suivante permet à un utilisateur de supprimer une autorité de certification privée.

```
{
   "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"
   }
```

```
}
```

T ag-on-create : Attacher des tags à une autorité de certification au moment de sa création

La politique suivante permet à un utilisateur d'appliquer des balises lors de la création de l'autorité de certification.

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

T ag-on-create : Balisage restreint

La tag-on-create politique suivante empêche l'utilisation de la paire clé-valeur Environment=Prod lors de la création de l'autorité de certification. Le balisage avec d'autres paires clé-valeur est autorisé.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"acm-pca:*",
         "Resource":"*"
      },
      {
         "Effect":"Deny",
         "Action":"acm-pca:TagCertificateAuthority",
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "aws:ResourceTag/Environment":[
```

```
                "Prod"
            ]
         }
      }
   }
]
}
```

Contrôle de l'accès à l'autorité de certification privée à l'aide de balises

La politique suivante n'autorise l'accès qu'aux autorités de certification dotées de la paire clé-valeur Environment=. PreProd Elle exige également que les nouvelles autorités de certification incluent cette balise.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm-pca:*"
         ],
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "aws:ResourceTag/Environment":[
                  "PreProd"
               ]
            }
         }
      }
   ]
}
```

Accès en lecture seule à Autorité de certification privée AWS

La stratégie suivante permet à un utilisateur de décrire et de répertorier des autorités de certification privées et de récupérer le certificat d'une autorité de certification privée et la chaîne de certificats.

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

Accès complet à Autorité de certification privée AWS

La politique suivante permet à un utilisateur d'effectuer n'importe quelle Autorité de certification privée AWS action.

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

Accès administrateur à toutes les ressources AWS

La politique suivante permet à un utilisateur d'effectuer n'importe quelle action sur n'importe quelle AWS ressource.

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

```
            "Action":"*",
            "Resource":"*"
        }
    ]
}
```

Bonnes pratiques de sécurité pour l'accès entre comptes aux autorités de certification privées

Un Autorité de certification privée AWS administrateur peut partager une autorité de certification avec les principaux (utilisateurs, rôles, etc.) d'un autre AWS compte. Lorsqu'une action a été reçue et acceptée, le principal peut utiliser l'autorité de certification pour émettre des certificats d'entité finale en utilisant nos Autorité de certification privée AWS AWS Certificate Manager ressources. Le principal peut utiliser l'autorité de certification pour émettre des certificats d'autorité de certification subordonnés à l'aide Autorité de certification privée AWS de.

> **Important**
>
> Les frais associés à un certificat émis dans le cadre d'un scénario multi-comptes sont facturés au AWS compte émetteur du certificat.

Pour partager l'accès à une autorité de certification, Autorité de certification privée AWS les administrateurs peuvent choisir l'une des méthodes suivantes :

- Utilisez AWS Resource Access Manager (RAM) pour partager le CA en tant que ressource avec un mandant d'un autre compte ou avec AWS Organizations. La RAM est une méthode standard de partage de AWS ressources entre comptes. Pour plus d'informations sur la RAM, consultez le guide de AWS RAM l'utilisateur. Pour plus d'informations AWS Organizations, consultez le guide de AWS Organizations l'utilisateur.
- Utilisez l' Autorité de certification privée AWS API ou la CLI pour associer une politique basée sur les ressources à une autorité de certification, accordant ainsi l'accès au principal d'un autre compte. Pour plus d'informations, consultez Politiques basées sur les ressources.

La Contrôle de l'accès à une autorité de certification privée section de ce guide fournit des flux de travail permettant d'accorder l'accès aux autorités de certification dans le cadre de scénarios à compte unique ou multicompte.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des politiques d'autorisation que vous créez et associez manuellement à une ressource (dans ce cas, une autorité de certification privée) plutôt qu'à une identité ou à un rôle d'utilisateur. Ou, au lieu de créer vos propres politiques, vous pouvez utiliser des politiques AWS gérées pour AWS Private CA. AWS RAM Pour appliquer une politique basée sur les ressources, un Autorité de certification privée AWS administrateur peut partager l'accès à une autorité de certification avec un utilisateur d'un autre AWS compte, directement ou par le biais de celui-ci. AWS Organizations Un Autorité de certification privée AWS administrateur peut également utiliser les API PCA et/ou les PutPolicy AWS CLI commandes correspondantes DeletePolicyput-policy, get-policy et delete-policy pour appliquer et gérer des politiques basées sur les ressources. GetPolicy

Pour des informations générales sur les politiques basées sur les ressources, voir Politiques basées sur l'identité et Politiques basées sur les ressources et Contrôle de l'accès à l'aide de politiques.

Pour afficher la liste des politiques basées sur les ressources AWS gérées pour AWS Private CA, accédez à la bibliothèque d'autorisations gérées dans la AWS Resource Access Manager console et recherchez. CertificateAuthority Comme pour toute politique, avant de l'appliquer, nous vous recommandons de l'appliquer dans un environnement de test afin de vous assurer qu'elle répond à vos exigences.

AWS Certificate Manager (ACM) disposant d'un accès partagé entre comptes à une autorité de certification privée peuvent émettre des certificats gérés signés par l'autorité de certification. Les émetteurs multicomptes sont limités par une politique basée sur les ressources et n'ont accès qu'aux modèles de certificats d'entité finale suivants :

- EndEntityCertificate/V1
- EndEntityClientAuthCertificate/V1
- EndEntityServerAuthCertificate/V1
- BlankEndEntityCertificate_APIPassThrough/V1
- BlankEndEntityCertificate_APICSR PassThrough/V1
- Certificat CA subordonné _ 0/V1 PathLen

Exemples de politiques

Cette section fournit des exemples de politiques multi-comptes adaptées à différents besoins. Dans tous les cas, le modèle de commande suivant est utilisé pour appliquer une politique :

```
$ 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
```

En plus de spécifier l'ARN d'une autorité de certification, l'administrateur fournit un identifiant de AWS compte ou un AWS Organizations identifiant qui sera autorisé à accéder à l'autorité de certification. Le JSON de chacune des politiques suivantes est formaté sous forme de fichier pour plus de lisibilité, mais peut également être fourni sous forme d'arguments de CLI en ligne.

> **Note**
>
> La structure des politiques basées sur les ressources JSON présentée ci-dessous doit être suivie avec précision. Seuls les champs d'identification des principaux (le numéro de AWS compte ou l'identifiant AWS des Organizations) et les ARN CA peuvent être configurés par les clients.

1. Fichier : policy1.json — Partage de l'accès à une autorité de certification avec un utilisateur d'un autre compte

   Remplacez *555555555555* par l'ID de AWS compte qui partage l'autorité de certification.

   Pour l'ARN de la ressource, remplacez les valeurs suivantes par vos propres valeurs :

   - *aws*- La AWS cloison. Par exemple, `aws`aws-us-gov,aws-cn,, etc.
   - *us-east-1*- La AWS région dans laquelle la ressource est disponible, par exemple`us-west-1`.
   - *111122223333*- L'identifiant de AWS compte du propriétaire de la ressource.
   - *11223344-1234-1122-2233-112233445566*- L'ID de ressource de l'autorité de certification.

   ```
   {
     "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. Fichier : policy2.json — Partage de l'accès à une autorité de certification via AWS Organizations

   Remplacez *o-a1b2c3d4z5* par l'ID. AWS Organizations

Pour l'ARN de la ressource, remplacez les valeurs suivantes par vos propres valeurs :

- *aws*- La AWS cloison. Par exemple, `aws`aws-us-gov,aws-cn,, etc.
- *us-east-1*- La AWS région dans laquelle la ressource est disponible, par exemple`us-west-1`.
- *111122223333*- L'identifiant de AWS compte du propriétaire de la ressource.
- *11223344-1234-1122-2233-112233445566*- L'ID de ressource de l'autorité de certification.

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

Protection des données dans AWS Private Certificate Authority

Le modèle de responsabilité AWS partagée de s'applique à la protection des données dans AWS Private Certificate Authority. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour en savoir plus sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.

- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-2 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour en savoir plus sur les points de terminaison FIPS (Federal Information Processing Standard) disponibles, consultez Federal Information Processing Standard (FIPS) 140-2 (Normes de traitement de l'information fédérale).

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Name (Nom). Cela inclut lorsque vous travaillez avec Autorité de certification privée AWS ou d'autres Services AWS utilisateurs de la console, de l'API ou AWS des SDK. AWS CLI Toutes les données que vous saisissez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Conformité du stockage et de la sécurité des clés Autorité de certification privée AWS privées

Les clés privées des autorités de certification privées sont stockées dans des modules de sécurité matériels AWS gérés (HSM). Les HSM sont conformes aux exigences de sécurité FIPS PUB 140-2 de niveau 3 pour les modules cryptographiques.

Chiffrement des données dans AWS Private CA Connector for Active Directory

AWS Private CA Connector for AD stocke les données de configuration des clients concernant les connecteurs, les modèles, les inscriptions aux annuaires, les noms principaux des services et les entrées de contrôle d'accès des groupes de modèles. Ces données sont cryptées en transit et au repos. Les informations relatives aux certificats émis via Connector for AD peuvent être découvertes à l'aide de l'GetCertificateaction de l' AWS Private CA API. Aucune information concernant les certificats émis, ou concernant le client ou la machine demandant un certificat, n'est stockée par AWS.

Validation de la conformité pour AWS Private Certificate Authority

Des auditeurs tiers évaluent la sécurité et AWS Private Certificate Authority la conformité de plusieurs programmes de AWS conformité. Il s'agit notamment des certifications SOC, PCI, FedRAMP, HIPAA et d'autres.

Pour une liste des AWS services concernés par des programmes de conformité spécifiques, voir AWS Services concernés par programme Services AWS de conformité dans Champ . Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Autorité de certification privée AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Pour les entreprises qui doivent chiffrer leurs compartiments Amazon S3, les rubriques suivantes décrivent comment configurer le chiffrement pour prendre Autorité de certification privée AWS en charge les actifs :
  - Chiffrement de vos rapports d'audit
  - Chiffrement de vos listes de révocation de certificats
- Guides de démarrage rapide sur la sécurité et la conformité Guides sur la sécurité et la conformité — Ces guides de déploiement abordent les considérations architecturales et fournissent les étapes à suivre pour déployer des environnements de base axés sur la sécurité et la conformité sur. AWS
- Livre blanc sur l'architecture pour la sécurité et la conformité HIPAA — Ce livre blanc décrit comment les entreprises peuvent créer des applications conformes à la loi HIPAA. AWS
- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- Évaluation des ressources à l'aide des règles du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub— Ce AWS service fournit une vue complète de l'état de votre sécurité interne, AWS ce qui vous permet de vérifier votre conformité aux normes et aux meilleures pratiques du secteur de la sécurité.

Utilisation de rapports d'audit avec votre autorité de certification privée

Vous pouvez créer un rapport d'audit pour répertorier tous les certificats émis ou révoqués par votre autorité de certification privée. Le rapport est enregistré dans un compartiment S3, nouveau ou existant, que vous spécifiez.

Pour de plus amples informations sur l'ajout d'une protection par chiffrement à vos rapports d'audit, veuillez consulter Chiffrer vos rapports d'audit .

Le chemin et le nom du fichier du rapport d'audit sont les suivants. L'ARN d'un compartiment Amazon S3 est la valeur de`bucket-name`. `CA_ID`Est l'identifiant unique de l'autorité de certification émettrice. `UUID`Est l'identifiant unique d'un rapport d'audit.

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

Vous pouvez générer un nouveau rapport toutes les 30 minutes et le télécharger à partir de votre compartiment. L'exemple suivant montre un rapport séparé par des virgules.

```
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
```

L'exemple suivant illustre un rapport au format 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**
>
> Lors du AWS Certificate Manager renouvellement d'un certificat, le rapport d'audit privé de l'autorité de certification remplit le `requestedByServicePrincipal` champ avec. `acm.amazonaws.com` Cela indique que le AWS Certificate Manager service a fait appel `IssueCertificate` à l' Autorité de certification privée AWS API pour le compte d'un client afin de renouveler le certificat.

Préparation d'un compartiment Amazon S3 pour les rapports d'audit

Pour stocker vos rapports d'audit, vous devez préparer un compartiment Amazon S3. Pour plus d'informations, consultez Comment créer un compartiment S3 ?

Votre compartiment S3 doit être sécurisé par une politique d'autorisation attachée. Les utilisateurs autorisés et les responsables du service doivent être `Put` autorisés Autorité de certification privée

AWS à placer des objets dans le compartiment et `Get` à les récupérer. Nous vous recommandons d'appliquer la politique ci-dessous, qui restreint l'accès à la fois à un AWS compte et à l'ARN d'une autorité de certification privée. Pour plus d'informations, consultez Ajouter une politique de compartiment à l'aide de la console Amazon S3.

> **Note**
>
> Au cours de la procédure de création d'un rapport d'audit dans la console, vous pouvez choisir d'autoriser la Autorité de certification privée AWS création d'un nouveau compartiment et d'appliquer une politique d'autorisation par défaut. La politique par défaut n'applique aucune `SourceArn` restriction à l'autorité de certification et est donc plus permissive que la stratégie recommandée. Si vous choisissez la valeur par défaut, vous pourrez toujours la modifier ultérieurement.

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

```
    ]
}
```

Création d'un rapport d'audit

Vous pouvez créer un rapport d'audit à partir de la console ou du AWS CLI.

Pour créer un rapport d'audit (console)

1. Connectez-vous à votre AWS compte et ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home.
2. Sur la page Autorités de certification privées, choisissez votre autorité de certification privée dans la liste.
3. Dans le menu Actions, choisissez Générer un rapport d'audit.
4. Sous Destination du rapport d'audit, pour Créer un nouveau compartiment S3 ? , choisissez Oui et saisissez un nom de compartiment unique, ou choisissez Non et choisissez un compartiment existant dans la liste.

   Si vous choisissez Oui, Autorité de certification privée AWS crée et attache la politique par défaut à votre compartiment. Si vous choisissez Non, vous devez associer une politique à votre compartiment avant de pouvoir générer un rapport d'audit. Utilisez le modèle de politique décrit dansPréparation d'un compartiment Amazon S3 pour les rapports d'audit. Pour plus d'informations sur l'attachement d'une politique, consultez Ajouter une politique de compartiment à l'aide de la console Amazon S3
5. Sous Format de sortie, choisissez JSON pour la notation des JavaScript objets ou CSV pour les valeurs séparées par des virgules.
6. Choisissez Generate audit report (Générer un rapport d'audit).

Pour créer un rapport d'audit (AWS CLI)

1. Si vous n'avez pas encore de compartiment S3 à utiliser, créez-en un.
2. Associez une politique à votre compartiment. Utilisez le modèle de politique décrit dansPréparation d'un compartiment Amazon S3 pour les rapports d'audit. Pour plus d'informations sur l'attachement d'une politique, consultez Ajouter une politique de compartiment à l'aide de la console Amazon S3
3. Utilisez la commande create-certificate-authority-audit-report pour créer le rapport d'audit et le placer dans le compartiment S3 préparé.

```
$ 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
```

Récupération d'un rapport d'audit

Pour récupérer un rapport d'audit à des fins d'inspection, utilisez la console, l'API, la CLI ou le SDK Amazon S3. Pour plus d'informations, consultez la section Téléchargement d'un objet dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Chiffrer vos rapports d'audit

Vous pouvez éventuellement configurer le chiffrement sur le compartiment Amazon S3 contenant vos rapports d'audit. Autorité de certification privée AWS prend en charge deux modes de chiffrement pour les actifs dans S3 :

- Chiffrement automatique côté serveur avec des clés AES-256 gérées par Amazon S3.
- Chiffrement géré par le client à l'aide AWS Key Management Service et AWS KMS key configuré selon vos spécifications.

> **Note**
>
> Autorité de certification privée AWS ne prend pas en charge l'utilisation de clés KMS par défaut générées automatiquement par S3.

Les procédures suivantes décrivent comment configurer chacune des options de chiffrement.

Pour configurer le chiffrement automatique

Procédez comme suit pour activer le chiffrement côté serveur S3.

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Dans le tableau Buckets, choisissez le compartiment qui contiendra vos Autorité de certification privée AWS actifs.

3. Sur la page de votre compartiment, choisissez l'onglet Propriétés .
4. Choisissez la carte de Chiffrement par défaut.
5. Sélectionnez Activer.
6. Choisissez la clé Amazon S3 (SSE-S3).
7. Choisissez Save Changes (Enregistrer les modifications).

Pour configurer le chiffrement personnalisé

Procédez comme suit pour activer le chiffrement à l'aide d'une clé personnalisée.

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Dans le tableau Buckets, choisissez le compartiment qui contiendra vos Autorité de certification privée AWS actifs.
3. Sur la page de votre compartiment, choisissez l'onglet Propriétés .
4. Choisissez la carte de Chiffrement par défaut.
5. Sélectionnez Activer.
6. Choisissez AWS Key Management Service la clé (SSE-KMS).
7. Choisissez Choisir parmi vos AWS KMS clés ou Entrer un AWS KMS key ARN.
8. Choisissez Save Changes (Enregistrer les modifications).
9. (Facultatif) Si vous ne possédez pas encore de clé KMS, créez-en une à l'aide de la commande AWS CLI create-key suivante :

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

La sortie contient l'ID de clé et le nom de ressource Amazon (ARN) de la clé KMS. Voici un exemple de résultat :

```
{
    "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. En suivant les étapes suivantes, vous autorisez le principal du Autorité de certification privée AWS service à utiliser la clé KMS. Par défaut, toutes les clés KMS sont privées ; seul le propriétaire de la ressource peut utiliser une clé KMS pour chiffrer et déchiffrer les données. Cependant, le propriétaire de la ressource peut accorder à d'autres utilisateurs et ressources des autorisations d'accès à la clé KMS. Le principal de service doit se trouver dans la même région que celle où la clé KMS est stockée.

    a. Tout d'abord, enregistrez la politique par défaut pour votre clé KMS à `policy.json` l'aide de la get-key-policycommande suivante :

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

    b. Ouvrez le fichier `policy.json` dans un éditeur de texte. Sélectionnez l'une des déclarations de politique suivantes et ajoutez-la à la politique existante.

    Si votre clé de compartiment Amazon S3 est activée, utilisez l'instruction suivante :

    ```
    {
       "Sid":"Allow ACM-PCA use of the key",
       "Effect":"Allow",
       "Principal":{
          "Service":"acm-pca.amazonaws.com"
       },
       "Action":[
          "kms:GenerateDataKey",
          "kms:Decrypt"
       ],
       "Resource":"*",
       "Condition":{
          "StringLike":{
             "kms:EncryptionContext:aws:s3:arn":"arn:aws:s3:::bucket-name"
          }
       }
    }
    ```

Si votre clé de compartiment Amazon S3 est désactivée, utilisez l'instruction suivante :

```
{
   "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. Enfin, appliquez la politique mise à jour à l'aide de la put-key-policycommande suivante :

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

Sécurité de l'infrastructure dans AWS Private Certificate Authority

En tant que service géré, AWS Private Certificate Authority il est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder AWS Private CA via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Autorité de certification privée AWS Points de terminaison VPC ()AWS PrivateLink

Vous pouvez créer une connexion privée entre votre VPC et configurer un point de Autorité de certification privée AWS terminaison VPC d'interface. Les points de terminaison de l'interface sont AWS PrivateLinkalimentés par une technologie d'accès privé aux opérations Autorité de certification privée AWS d'API. AWS PrivateLink achemine tout le trafic réseau entre votre VPC et Autorité de certification privée AWS via le réseau Amazon, évitant ainsi toute exposition sur Internet ouvert. Chaque point de terminaison d'un VPC est représenté par une ou plusieurs interfaces réseau Elastic avec des adresses IP privées dans vos sous-réseaux VPC.

Le point de terminaison VPC de l'interface connecte directement votre VPC Autorité de certification privée AWS sans passerelle Internet, périphérique NAT, connexion VPN ou connexion. AWS Direct Connect Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour communiquer avec l' Autorité de certification privée AWS API.

Pour l'utiliser Autorité de certification privée AWS via votre VPC, vous devez vous connecter à partir d'une instance située à l'intérieur du VPC. Vous pouvez également connecter votre réseau privé à votre VPC à l'aide d'un AWS Virtual Private Network (AWS VPN) ou. AWS Direct Connect Pour plus d'informations AWS VPN, consultez la section Connexions VPN dans le guide de l'utilisateur Amazon VPC. Pour plus d'informations AWS Direct Connect, voir Création d'une connexion dans le guide de AWS Direct Connect l'utilisateur.

Autorité de certification privée AWS ne nécessite pas l'utilisation de AWS PrivateLink, mais nous vous recommandons de l'utiliser comme couche de sécurité supplémentaire. Pour plus d'informations

sur les points AWS PrivateLink de terminaison VPC, consultez la section Accès aux services via. AWS PrivateLink

Considérations relatives aux points de Autorité de certification privée AWS terminaison VPC

Avant de configurer les points de terminaison VPC d'interface pour Autorité de certification privée AWS, tenez compte des points suivants :

- Autorité de certification privée AWS peut ne pas prendre en charge les points de terminaison VPC dans certaines zones de disponibilité. Lorsque vous créez un point de terminaison VPC, vérifiez d'abord le support dans la console de gestion. Les zones de disponibilité non prises en charge sont marquées « Service non pris en charge dans cette zone de disponibilité ».
- Les points de terminaison d'un VPC ne prennent pas en charge les demandes inter-régionales. Veillez à créer votre point de terminaison dans la même région que celle dans laquelle vous souhaitez envoyer vos appels d'API à Autorité de certification privée AWS.
- Les points de terminaison VPC prennent uniquement en charge le DNS fourni par Amazon via Amazon Route 53. Si vous souhaitez utiliser votre propre DNS, vous pouvez utiliser le transfert DNS conditionnel. Pour en savoir plus, consultez Jeux d'options DHCP dans le Guide de l'utilisateur Amazon VPC.
- Le groupe de sécurité attaché au point de terminaison d'un VPC doit autoriser les connexions entrantes sur le port 443 à partir du sous-réseau privé du VPC.
- AWS Certificate Manager ne prend pas en charge les points de terminaison VPC.
- Les points de terminaison FIPS (et leurs régions) ne prennent pas en charge les points de terminaison d'un VPC.

Autorité de certification privée AWS L'API prend actuellement en charge les points de terminaison VPC dans les domaines suivants : Régions AWS

- USA Est (Ohio)
- USA Est (Virginie du Nord)
- USA Ouest (Californie du Nord)
- USA Ouest (Oregon)
- Afrique (Le Cap)
- Asie-Pacifique (Hong Kong)

- Asia Pacific (Mumbai)
- Asie-Pacifique (Osaka)
- Asia Pacific (Seoul)
- Asie-Pacifique (Singapour)
- Asie-Pacifique (Sydney)
- Asie-Pacifique (Tokyo)
- Canada (Centre)
- Europe (Francfort)
- Europe (Irlande)
- Europe (Londres)
- Europe (Paris)
- Europe (Stockholm)
- Europe (Milan)
- Israël (Tel Aviv)
- Moyen-Orient (Bahreïn)
- Amérique du Sud (Sao Paulo)

Création des points de terminaison VPC pour Autorité de certification privée AWS

Vous pouvez créer un point de terminaison VPC pour le Autorité de certification privée AWS service à l'aide de la console VPC à l'adresse https://console.aws.amazon.com/vpc/ ou du. AWS Command Line Interface Pour plus d'informations, consultez la procédure de création d'un point de terminaison d'interface dans le guide de l'utilisateur Amazon VPC. Autorité de certification privée AWS prend en charge les appels à toutes ses opérations d'API au sein de votre VPC.

Si vous avez activé les noms d'hôte DNS privés pour le point de terminaison, le point de Autorité de certification privée AWS terminaison par défaut devient désormais votre point de terminaison VPC. Pour obtenir une liste complète des points de terminaison de service par défaut, veuillez consulter Points de terminaison et quotas de service.

Si vous n'avez pas activé les noms d'hôte DNS privés, Amazon VPC fournit un nom de point de terminaison DNS que vous pouvez utiliser au format suivant :

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

> **Note**
>
> La *`région`* de valeur représente l'identifiant de région d'une AWS région prise en charge par Autorité de certification privée AWS, par exemple `us-east-2` pour la région USA Est (Ohio). Pour en obtenir la liste Autorité de certification privée AWS, voir AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.

Pour plus d'informations, consultez la section Points de terminaison Autorité de certification privée AWS VPC (AWS PrivateLink) dans le guide de l'utilisateur Amazon VPC.

Création d'une stratégie de point de terminaison de VPC pour Autorité de certification privée AWS

Vous pouvez créer une politique pour les points de terminaison Amazon VPC Autorité de certification privée AWS afin de spécifier les éléments suivants :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, veuillez consulter Contrôle de l'accès aux services avec des points de terminaison d'un VPC dans le Amazon VPC Guide de l'utilisateur.

Exemple — Politique de point de terminaison VPC pour les actions Autorité de certification privée AWS

Lorsqu'elle est attachée à un point de terminaison, la politique suivante accorde l'accès à tous les principaux aux Autorité de certification privée AWS actions `IssueCertificate`,`DescribeCertificateAuthority`,`GetCertificate`,`GetCertificateAuthori` et`ListTags`. La ressource dans chaque strophe est une autorité de certification privée. La première strophe autorise la création de certificats d'entité finale à l'aide de l'autorité de certification privée spécifiée et du modèle de certificat. Si vous ne souhaitez pas contrôler le modèle utilisé, la section `Condition` n'est pas nécessaire. Cependant, la suppression de cette option permet à tous les principaux de créer des certificats d'une autorité de certification ainsi que des certificats d'entité finale.

```
{
```

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

Journalisation et surveillance pour AWS Private Certificate Authority

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité AWS Private Certificate Authority et des performances de vos AWS solutions. Vous devez collecter des données

de surveillance provenant de toutes les parties de votre AWS solution afin de pouvoir corriger plus facilement une défaillance multipoint, le cas échéant.

Les rubriques suivantes décrivent les outils AWS de surveillance du cloud disponibles avec Autorité de certification privée AWS.

Rubriques

CloudWatch Métriques prises en charge

Amazon CloudWatch est un service de surveillance des AWS ressources. Vous pouvez l'utiliser CloudWatch pour collecter et suivre les métriques, définir des alarmes et réagir automatiquement aux modifications de vos AWS ressources. CloudWatch les métriques sont publiées au moins une fois.

Autorité de certification privée AWS prend en charge les CloudWatch métriques suivantes.

Métrique	Description
`CRLGenerated`	Une liste de révocation de certificats a été générée. Cette métrique s'applique uniquement à une autorité de certification privée.
`MisconfiguredCRLBucket`	Le compartiment S3 spécifié pour la liste de révocation de certificats n'est pas correctement configuré. Vérifiez la stratégie de compartiment. Cette métrique s'applique uniquement à une autorité de certification privée.
`Time`	Le délai en millisecondes entre une demande d'émission et la fin (ou l'échec) de l'émissio n. Cette métrique s'applique uniquement à l'IssueCertificateopération.

Métrique	Description
`Success`	Un certificat a été émis avec succès. Cette métrique s'applique uniquement à l'IssueCert ificateopération.
`Failure`	Une opération a échoué. Cette métrique s'applique uniquement à l'IssueCert ificateopération.

Pour plus d'informations sur CloudWatch les métriques, consultez les rubriques suivantes :

- Utilisation d'Amazon CloudWatch Metrics
- Création d' CloudWatchalarmes Amazon

Utilisation des CloudWatch événements

Vous pouvez utiliser Amazon CloudWatch Events pour automatiser vos AWS services et répondre automatiquement aux événements du système tels que les problèmes de disponibilité des applications ou les modifications des ressources. Les événements issus AWS des services sont transmis à CloudWatch Events en temps quasi réel. Vous pouvez rédiger des règles simples pour indiquer les événements qui vous intéressent et les actions automatisées à effectuer lorsqu'un événement correspond à une règle. CloudWatch Les événements sont publiés au moins une fois. Pour plus d'informations, voir Création d'une règle d' CloudWatch événements déclenchant un événement.

CloudWatch Les événements sont transformés en actions à l'aide d'Amazon EventBridge. Avec EventBridge, vous pouvez utiliser des événements pour déclencher des cibles, notamment des AWS Lambda fonctions, AWS Batch des tâches, des rubriques Amazon SNS et bien d'autres. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge ?

Succès ou échec lors de la création d'une autorité de certification privée

Ces événements sont déclenchés par l'CreateCertificateAuthorityopération.

Réussite

En cas de succès, l'opération renvoie l'ARN de la nouvelle autorité de certification.

```
{
   "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"
   }
}
```

Échec

En cas d'échec, l'opération renvoie un ARN pour l'autorité de certification. À l'aide de l'ARN, vous pouvez appeler DescribeCertificateAuthoritypour déterminer le statut de l'autorité de certification.

```
{
   "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"
   }
}
```

Succès ou échec lors de la délivrance d'un certificat

Ces événements sont déclenchés par l'IssueCertificateopération.

Réussite

En cas de succès, l'opération renvoie les ARN de l'autorité de certification et du nouveau certificat.

```
{
   "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"
   }
}
```

Échec

En cas d'échec, l'opération renvoie un ARN de certificat et l'ARN de l'autorité de certification. Avec l'ARN du certificat, vous pouvez appeler GetCertificatepour connaître la raison de l'échec.

```
{
   "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"
   }
}
```

Succès lors de la révocation d'un certificat

Cet événement est déclenché par l'RevokeCertificateopération.

Aucun événement n'est envoyé si la révocation échoue ou si le certificat a déjà été révoqué.

Succès

En cas de succès, l'opération renvoie les ARN de l'autorité de certification et du certificat révoqué.

```
{
   "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"
   }
}
```

Succès ou échec lors de la génération d'une CRL

Ces événements sont déclenchés par l'RevokeCertificateopération, qui doit entraîner la création d'une liste de révocation de certificats (CRL).

Réussite

En cas de succès, l'opération renvoie l'ARN de l'autorité de certification associée à la liste de révocation de certificats.

```
{
   "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"
   }
}
```

Échec 1 : la CRL n'a pas pu être enregistrée sur Amazon S3 en raison d'une erreur d'autorisation

Vérifiez les autorisations de votre compartiment Amazon S3 si cette erreur se produit.

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

Échec 2 : la CRL n'a pas pu être enregistrée sur Amazon S3 en raison d'une erreur interne

Réessayez l'opération si cette erreur se produit.

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

Échec 3 : Autorité de certification privée AWS échec de la création d'une CRL

Pour résoudre cette erreur, vérifiez vos métriques 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."
   }
}
```

Succès ou échec lors de la création d'un rapport d'audit CA

Ces événements sont déclenchés par l'CreateCertificateAuthorityAuditReportopération.

Réussite

En cas de succès, l'opération renvoie l'ARN de l'autorité de certification et l'ID du rapport d'audit.

```
{
   "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"
   }
}
```

Échec

Un rapport d'audit peut échouer en cas d' Autorité de certification privée AWS absence d'PUTautorisations sur votre compartiment Amazon S3, lorsque le chiffrement est activé sur le compartiment ou pour d'autres raisons.

```
{
   "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"
```

```
    }
}
```

En utilisant CloudTrail

Vous pouvez l'utiliser AWS CloudTrailpour enregistrer les appels d'API effectués par AWS Private Certificate Authority. Pour plus d'informations, consultez les rubriques suivantes.

Rubriques

- Création d'une stratégie
- Récupération d'une politique
- Suppression d'une stratégie
- Création d'une autorité de certification
- Générer une CRL
- Générer une réponse OCSP
- Création d'un rapport d'audit
- Supprimer une autorité de certification
- Restauration d'une autorité de certification
- Décrire une autorité de certification
- Récupération d'un certificat d'autorité de certification
- Récupération de la demande de signature de l'autorité de certification
- Récupération d'un certificat
- Importation d'un certificat d'autorité de certification
- Délivrance d'un certificat
- Liste des autorités de certification
- Établissement d'une liste de balises
- Révocation d'un certificat
- Marquage des autorités de certification privées
- Supprimer les balises d'une autorité de certification privée
- Mettre à jour une autorité de certification

Création d'une stratégie

L' CloudTrail exemple suivant montre les résultats d'un appel à l'PutPolicyopération.


```
{
   "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"
}
```

Récupération d'une politique

L' CloudTrail exemple suivant montre les résultats d'un appel à l'GetPolicyopération.


```
{
   "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"
}
```

Suppression d'une stratégie

L' CloudTrail exemple suivant montre les résultats d'un appel à l'DeletePolicyopération.

```
{
   "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"
}
```

Création d'une autorité de certification

L' CloudTrail exemple suivant montre les résultats d'un appel à l'CreateCertificateAuthorityopération.

```
{
   "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"
}
```

Générer une CRL

L' CloudTrail exemple suivant montre l'enregistrement d'un événement 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"
}
```

Générer une réponse OCSP

L' CloudTrail exemple suivant montre l'enregistrement d'un événement GenerateOCSPResponse.

```
{
   "eventVersion":"1.08",
   "userIdentity":{
      "accountId":"account",
      "invokedBy":"acm-pca.amazonaws.com"
   },
   "eventTime":"2021-02-08T23:52:29Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"GenerateOCSPResponse",
   "awsRegion":"region",
   "sourceIPAddress":"acm-pca.amazonaws.com",
   "userAgent":"acm-pca.amazonaws.com",
   "eventID":"01234567-89ab-cdef-0123-456789abcdef",
   "readOnly":false,
   "resources":[
      {
         "type":"AWS::ACMPCA::Certificate",
         "ARN":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
      }
   ]
}
```

Création d'un rapport d'audit

L' CloudTrail exemple suivant montre les résultats d'un appel à l'CreateCertificateAuthorityAuditReportopération.


```
{
   "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"
}
```


Supprimer une autorité de certification

L' CloudTrail exemple suivant montre les résultats d'un appel à l'DeleteCertificateAuthorityopération. Dans cet exemple, l'autorité de certification ne peut pas être supprimée, car son état est `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"
}
```

Restauration d'une autorité de certification

L' CloudTrail exemple suivant montre les résultats d'un appel à l'RestoreCertificateAuthorityopération. Dans cet exemple, l'autorité de certification ne peut pas être restaurée, car son état est `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"
}
```

Décrire une autorité de certification

L' CloudTrail exemple suivant montre les résultats d'un appel à l'DescribeCertificateAuthorityopération.

```
{
   "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"
}
```

Récupération d'un certificat d'autorité de certification

L' CloudTrail exemple suivant montre les résultats d'un appel à l'GetCertificateAuthorityCertificateopération.

```
{
   "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"
}
```

Récupération de la demande de signature de l'autorité de certification

L' CloudTrail exemple suivant montre les résultats d'un appel à l'GetCertificateAuthorityCsropération.

```
{
   "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"
}
```

Récupération d'un certificat

L' CloudTrail exemple suivant montre les résultats d'un appel à l'GetCertificateopération.

```
{
   "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"
}
```

Importation d'un certificat d'autorité de certification

L' CloudTrail exemple suivant montre les résultats d'un appel à l'ImportCertificateAuthorityCertificateopération.

```
{
   "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"
}
```

Délivrance d'un certificat

L' CloudTrail exemple suivant montre les résultats d'un appel à l'IssueCertificateopération.

```
{
   "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"
}
```

Liste des autorités de certification

L' CloudTrail exemple suivant montre les résultats d'un appel à l'ListCertificateAuthoritiesopération.


```
{
   "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"
}
```


Établissement d'une liste de balises

L' CloudTrail exemple suivant montre les résultats d'un appel à l'ListTagsopération.


```
{
   "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"
}
```

Révocation d'un certificat

L' CloudTrail exemple suivant montre les résultats d'un appel à l'RevokeCertificateopération.

```
{
   "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"
}
```

Marquage des autorités de certification privées

L' CloudTrail exemple suivant montre les résultats d'un appel à l'TagCertificateAuthorityopération.

```
{
   "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"
}
```

Supprimer les balises d'une autorité de certification privée

L' CloudTrail exemple suivant montre les résultats d'un appel à l'UntagCertificateAuthorityopération.

```
{
   "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"
}
```

Mettre à jour une autorité de certification

L' CloudTrail exemple suivant montre les résultats d'un appel à l'UpdateCertificateAuthorityopération.


```
{
   "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"
}
```

Planification de votre Autorité de certification privée AWS déploiement

Autorité de certification privée AWS vous donne un contrôle complet, basé sur le cloud, sur la PKI privée (infrastructure à clé publique) de votre entreprise, qu'il s'agisse d'une autorité de certification (CA) racine, en passant par des autorités de certification subordonnées, ou encore des certificats d'entité finale. Une planification approfondie est essentielle pour une PKI sécurisée, gérable, extensible et adaptée aux besoins de votre organisation. Cette section fournit des conseils sur la conception d'une hiérarchie d'autorité de certification, la gestion des cycles de vie de vos certificats d'autorité de certification privée et d'entité finale privée et l'application des bonnes pratiques en matière de sécurité.

Cette section explique comment Autorité de certification privée AWS préparer l'utilisation avant de créer une autorité de certification (CA) privée. Il explique également la possibilité d'ajouter la prise en charge de la révocation via le protocole OCSP (Online Certificate Status Protocol) ou une liste de révocation de certificats (CRL).

En outre, vous devez déterminer si votre organisation préfère héberger ses informations d'identification CA racine privées sur site plutôt que chez AWS. Dans ce cas, vous devez configurer et sécuriser une PKI privée autogérée avant de l'utiliser. Autorité de certification privée AWS Dans ce scénario, vous créez ensuite une autorité de certification subordonnée dans, Autorité de certification privée AWS soutenue par une autorité de certification parent en dehors de Autorité de certification privée AWS. Pour plus d'informations, voir Installation d'un certificat d'autorité de certification subordonnée signé par une autorité de certification parent externe.

Rubriques

- Configuration de votre AWS compte et du AWS CLI
- Conception d'une hiérarchie CA
- Gestion du cycle de vie de l'autorité de certification privée
- Configuration d'une méthode de révocation des certificats
- Modes d'autorité de certification
- Planification de la résilience

Configuration de votre AWS compte et du AWS CLI

Si vous n'êtes pas encore client d'Amazon Web Services (AWS), vous devez vous inscrire pour pouvoir utiliser Autorité de certification privée AWS. Votre compte a automatiquement accès à tous les services disponibles, mais vous êtes facturé uniquement pour les services que vous utilisez.

> **Note**
>
> Autorité de certification privée AWS n'est pas disponible dans le niveau AWS gratuit.

Rubriques

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique en matière de sécurité consiste à attribuer un accès administratif à un utilisateur et à n'utiliser que l'utilisateur root pour effectuer les tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. Vous pouvez afficher l'activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en choisissant Mon compte.

Création d'un utilisateur doté d'un accès administratif

Une fois que vous vous êtes inscrit à un utilisateur administratif Compte AWS, que vous Utilisateur racine d'un compte AWS l'avez sécurisé AWS IAM Identity Center, que vous l'avez activé et que vous en avez créé un, afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur racine, consultez Connexion en tant qu'utilisateur racine dans le Guide de l'utilisateur Connexion à AWS .
2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, consultez la section Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d' AWS IAM Identity Center dans le Guide de l'utilisateur AWS IAM Identity Center .
2. Dans IAM Identity Center, accordez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connectez-vous en tant qu'utilisateur disposant d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribuer l'accès à des utilisateurs supplémentaires

1. Dans IAM Identity Center, créez un ensemble d'autorisations conforme aux meilleures pratiques en matière d'application des autorisations du moindre privilège.

   Pour obtenir des instructions, voir Création d'un ensemble d'autorisations dans le guide de AWS IAM Identity Center l'utilisateur.
2. Affectez des utilisateurs à un groupe, puis attribuez un accès d'authentification unique au groupe.

   Pour obtenir des instructions, voir Ajouter des groupes dans le guide de AWS IAM Identity Center l'utilisateur.

Installez le AWS Command Line Interface

Si vous ne l'avez pas installé AWS CLI mais que vous souhaitez l'utiliser, suivez les instructions sur AWS Command Line Interface. Dans ce guide, nous partons du principe que vous avez configuré votre point de terminaison, votre région et les informations d'authentification, et nous omettons ces paramètres dans les exemples de commandes.

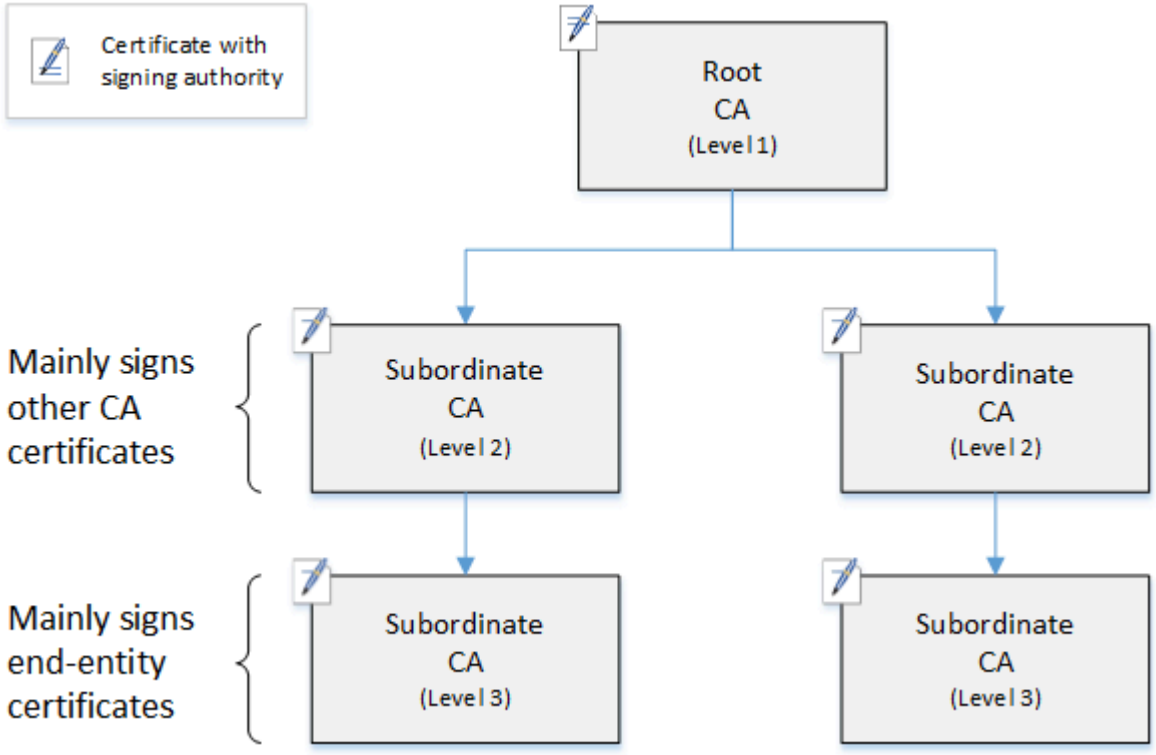
Conception d'une hiérarchie CA

Vous pouvez ainsi créer une hiérarchie d'autorités de certification comportant jusqu'à cinq niveaux. Autorité de certification privée AWS L'autorité de certification racine, située en haut d'une arborescence hiérarchique, peut avoir un nombre quelconque de branches. L'autorité de certification racine peut avoir jusqu'à quatre niveaux d'autorité de certification subordonnées sur chaque branche. Vous pouvez également créer plusieurs hiérarchies, chacune ayant sa propre racine.

Une hiérarchie d'autorité de certification bien conçue offre les avantages suivants :

- Contrôles de sécurité précis adaptés à chaque autorité de certification
- Répartition des tâches administratives pour un meilleur équilibrage de la charge et une meilleure sécurité
- Utilisation des autorités de certification dont la confiance est limitée et révocable pour les opérations quotidiennes
- Périodes de validité et limites de chemin de certificat

Le diagramme suivant illustre une hiérarchie d'autorité de certification simple à trois niveaux.



Chaque autorité de certification de l'arborescence est soutenue par un certificat X.509 v3 doté d'une autorité de signature (symbolisée par l' pen-and-paper icône). Cela signifie qu'en tant qu'autorités de certification, elles peuvent signer d'autres certificats qui leur sont subordonnés. Lorsqu'une autorité de certification signe le certificat d'une autorité de certification de niveau inférieur, elle confère un pouvoir limité et révocable au certificat signé. L'autorité de certification racine du niveau 1 signe les certificats d'une autorité de certification subordonnée de haut niveau au niveau 2. Ces autorités de certification signent à leur tour des certificats pour les autorités de certification de niveau 3 qui sont utilisés par les administrateurs de l'infrastructure à clé publique (PKI) qui gèrent les certificats d'entité finale.

La sécurité dans une hiérarchie d'autorité de certification doit être configurée pour être la plus forte en haut de l'arborescence. Cette disposition protège le certificat d'une autorité de certification racine et sa clé privée. L'autorité de certification racine ancre la confiance pour toutes les autorités de certification subordonnées et les certificats d'entité finale en dessous. Bien que des dommages localisés puissent résulter de la compromission d'un certificat d'entité finale, la compromission de la racine détruit la confiance dans l'ensemble de la PKI. Les autorités de certification racine et subordonnées de haut niveau ne sont utilisées que rarement (généralement pour signer d'autres certificats d'autorité de certification). Par conséquent, ils sont étroitement contrôlés et vérifiés afin de réduire le risque de compromis. Aux niveaux inférieurs de la hiérarchie, la sécurité est moins

restrictive. Cette approche permet les tâches administratives courantes consistant à émettre et à révoquer des certificats d'entité finale pour les utilisateurs, les hôtes d'ordinateur et les services logiciels.

> **Note**
>
> L'utilisation d'une autorité de certification racine pour signer un certificat subordonné est un événement rare qui se produit dans une poignée de circonstances :
>
> - Lorsque la PKI est créée
> - Lorsqu'une autorité de certification de haut niveau doit être remplacée
> - Lorsqu'un répondeur de liste de révocation de certificats (CRL) ou OCSP (Online Certificate Status Protocol) doit être configuré
>
> Les autorités de certification racine et d'autres autorités de certification de haut niveau nécessitent des processus opérationnels hautement sécurisés et des protocoles de contrôle d'accès.

Rubriques

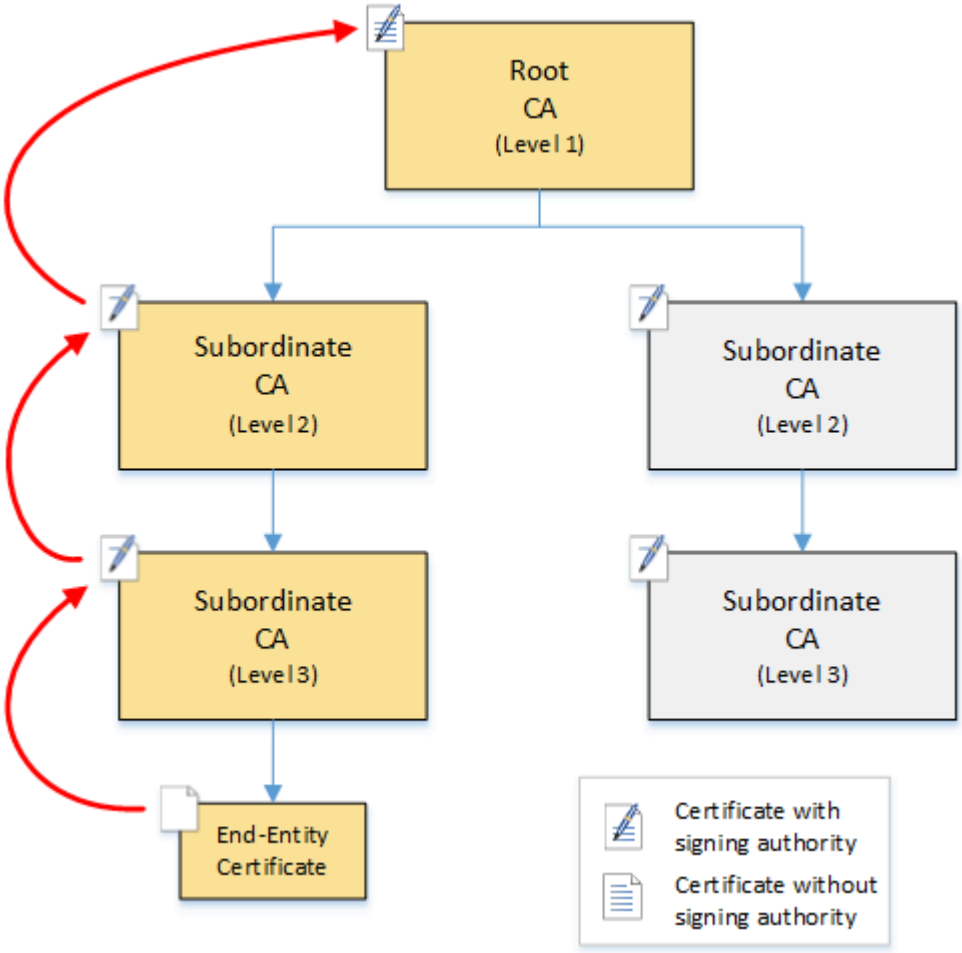
Validation des certificats d'entité finale

Les certificats d'entité finale dérivent leur confiance d'un chemin de certification menant à l'autorité de certification subordonnée à une autorité de certification racine. Lorsqu'un navigateur web ou un autre client est présenté avec un certificat d'entité finale, il tente de construire une chaîne de confiance. Par exemple, il peut vérifier que le nom distinctif de l'émetteur et le nom distinctif de l'objet du certificat correspondent aux champs correspondants du certificat d'une autorité de certification émettrice. La correspondance se poursuivrait à chaque niveau successif jusqu'à ce que le client atteigne une racine de confiance contenue dans son magasin de confiance.

Le magasin de confiance est une bibliothèque d'autorités de certification approuvées que contient le navigateur ou le système d'exploitation. Pour une PKI privée, le service informatique de votre

organisation doit s'assurer que chaque navigateur ou système a précédemment ajouté l'autorité de certification racine privée à son magasin de confiance. Sinon, le chemin de certification ne peut pas être validé, ce qui entraîne des erreurs client.

Le diagramme suivant montre le chemin de validation suivi par un navigateur lorsqu'il est présenté avec un certificat X.509 d'entité finale. Notez que le certificat d'entité finale n'a pas le pouvoir de signature et sert uniquement à authentifier l'entité qui le possède.



Le navigateur inspecte le certificat d'entité finale. Le navigateur trouve que le certificat offre une signature de l'autorité de certification subordonnée (niveau 3) comme informations d'identification d'approbation. Les certificats des autorités de certification subordonnées doivent être inclus dans le même fichier PEM. Alternativement, ils peuvent également se trouver dans un fichier séparé contenant les certificats qui composent la chaîne d'approbation. En les trouvant, le navigateur vérifie le certificat d'une autorité de certification subordonnée (niveau 3) et trouve qu'il offre une signature de l'autorité de certification subordonnée (niveau 2). À son tour, l'autorité de certification subordonnée (niveau 2) offre une signature de l'autorité de certification racine (niveau 1) comme informations d'identification d'approbation. Si le navigateur trouve une copie du certificat d'une

autorité de certification racine privée préinstallée dans son magasin de confiance, il valide le certificat d'entité finale comme approuvé.

En règle générale, le navigateur vérifie également chaque certificat par rapport à une liste de révocation de certificats (CRL). Un certificat expiré, révoqué ou mal configuré est rejeté et la validation échoue.

Planification de la structure d'une hiérarchie d'autorités de certification

En général, votre hiérarchie d'autorité de certification doit refléter la structure de votre organisation. Optez pour une longueur de chemin (c'est-à-dire le nombre de niveaux d'autorité de certification) qui ne soit pas supérieure à ce qui est nécessaire pour déléguer les rôles d'administration et de sécurité. L'ajout d'une autorité de certification à la hiérarchie implique l'augmentation du nombre de certificats dans le chemin de certification, ce qui augmente le temps de validation. Le fait de maintenir la longueur du chemin au minimum réduit également le nombre de certificats envoyés par le serveur au client lors de la validation d'un certificat d'entité finale.

En théorie, une autorité de certification racine, qui n'a aucun pathLenConstraintparamètre, peut autoriser des niveaux illimités d'autorités de certification subordonnées. Une autorité de certification subordonnée peut avoir autant d'autorités de certification subordonnées enfants que le permet sa configuration interne. Autorité de certification privée AWS les hiérarchies gérées prennent en charge les parcours de certification CA jusqu'à cinq niveaux de profondeur.

Les structures de CA bien conçues présentent plusieurs avantages :

- Contrôles administratifs distincts pour différentes unités organisationnelles
- La possibilité de déléguer l'accès aux autorités de certification subordonnées
- Une structure hiérarchique qui protège les autorités de certification de niveau supérieur avec des contrôles de sécurité supplémentaires

Deux structures de CA communes accomplissent tout cela :

- Deux niveaux d'autorité de certification : autorité de certification racine et autorité de certification subordonnée

  Il s'agit de la structure d'autorité de certification la plus simple qui permet des stratégies d'administration, de contrôle et de sécurité distinctes pour l'autorité de certification racine et une autorité de certification subordonnée. Vous pouvez maintenir des contrôles et des stratégies

restrictifs pour votre autorité de certification racine tout en autorisant un accès plus permissif à l'autorité de certification subordonnée. Cette dernière est utilisée pour la délivrance en bloc de certificats d'entité finale.

- Trois niveaux de CA : CA racine et deux couches de CA subordonnée

  Similaire à ce qui précède, cette structure ajoute une couche d'autorité de certification supplémentaire pour séparer davantage l'autorité de certification racine des opérations d'autorité de certification de bas niveau. La couche d'autorité de certification intermédiaire est utilisée uniquement pour signer les autorités de certification subordonnées qui effectuent l'émission de certificats d'entité finale.

Les structures de CA moins courantes sont les suivantes :

- Quatre niveaux de CA ou plus

  Bien qu'elles soient moins courantes que les hiérarchies à trois niveaux, les hiérarchies de CA comportant quatre niveaux ou plus sont possibles et peuvent être nécessaires pour permettre la délégation administrative.
- Un niveau d'autorité de certification : CA racine uniquement

  Cette structure est couramment utilisée pour le développement et les tests lorsqu'une chaîne complète de confiance n'est pas requise. Utilisée en production, elle est atypique. En outre, elle va à l'encontre de la bonne pratique qui consiste à maintenir des stratégies de sécurité distinctes pour l'autorité de certification racine et les autorités de certification qui émettent des certificats d'entité finale.

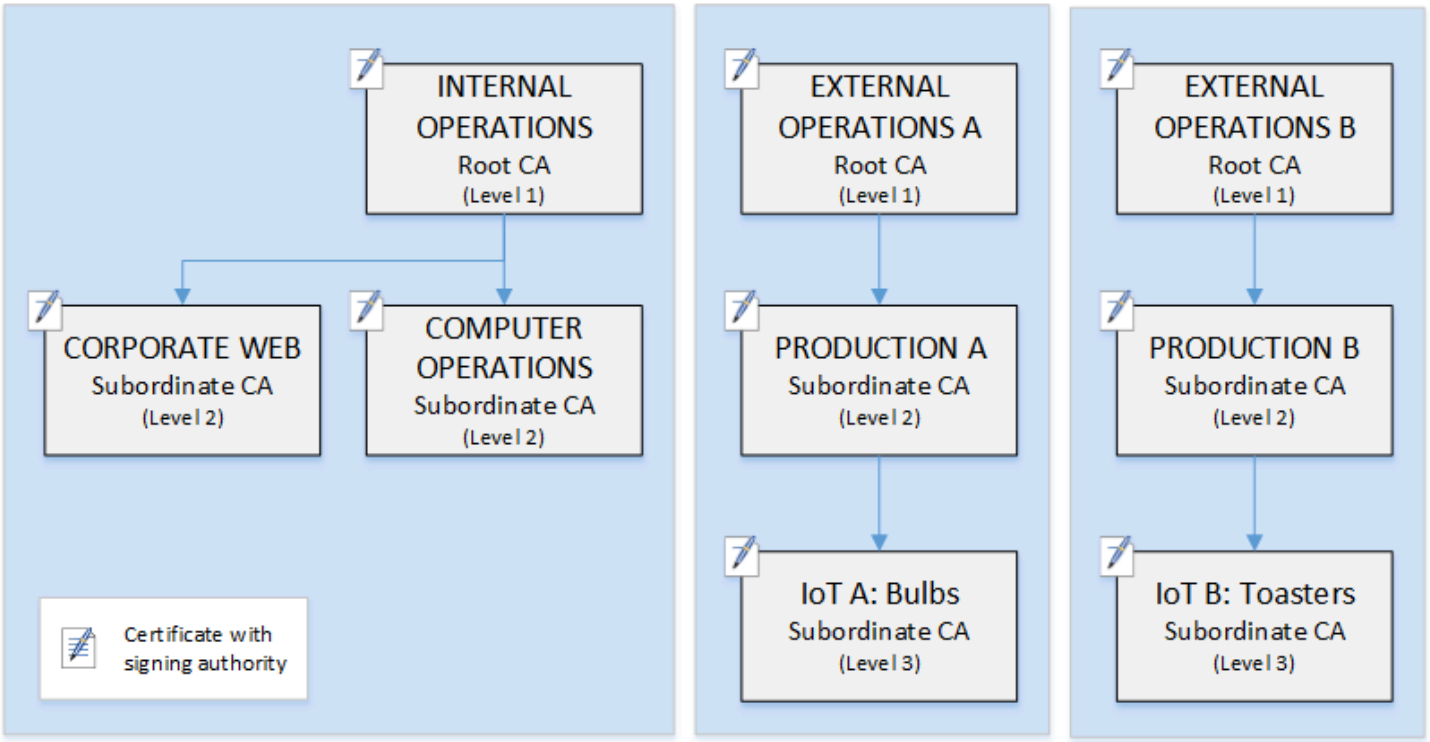
  Toutefois, si vous émettez déjà des certificats directement depuis une autorité de certification racine, vous pouvez effectuer la migration vers Autorité de certification privée AWS. Cela offre des avantages en termes de sécurité et de contrôle par rapport à l'utilisation d'une autorité de certification racine gérée avec OpenSSL ou un autre logiciel.

Exemple de PKI privé pour un fabricant

Dans cet exemple, une société de technologie hypothétique fabrique deux produits de l'Internet des objets (IoT), une ampoule intelligente et un grille-pain intelligent. Au cours de la production, chaque appareil reçoit un certificat d'entité finale afin qu'il puisse communiquer en toute sécurité sur

Internet avec le fabricant. La PKI de la société sécurise également son infrastructure informatique, y compris le site web interne et divers services informatiques auto-hébergés qui gèrent les opérations financières et commerciales.

Par conséquent, la hiérarchie de CA modélise étroitement ces aspects administratifs et opérationnels de l'entreprise.



Cette hiérarchie contient trois racines, une pour les opérations internes et deux pour les opérations externes (une autorité de certification racine pour chaque ligne de produits). Il illustre également la longueur de plusieurs parcours de certification, avec deux niveaux de CA pour les opérations internes et trois niveaux pour les opérations externes.

L'utilisation d'autorités de certification racines séparées et de couches d'autorités de certification subordonnées supplémentaires du côté des opérations externes est une décision de conception répondant aux besoins commerciaux et de sécurité. Avec plusieurs arborescences de CA, la PKI est à l'épreuve de l'avenir contre les réorganisations, les dessaisissements ou les acquisitions d'entreprises. Lorsque des modifications se produisent, une hiérarchie d'autorité de certification racine entière peut se déplacer proprement avec la division qu'elle sécurise. Et avec deux niveaux de CA subordonnées, les CA racine ont un niveau élevé d'isolement par rapport aux CA de niveau 3 qui sont responsables de la signature en bloc des certificats pour des milliers ou des millions d'articles manufacturés.

Sur le plan interne, les opérations web et informatiques internes de l'entreprise complètent une hiérarchie à deux niveaux. Ces niveaux permettent aux administrateurs web et aux ingénieurs des opérations de gérer indépendamment l'émission de certificats pour leurs propres domaines de travail. La compartimentalisation de la PKI en domaines fonctionnels distincts est une bonne pratique en matière de sécurité et protège chacun contre un compromis susceptible d'affecter l'autre. Les administrateurs web émettent des certificats d'entité finale à utiliser par les navigateurs web dans toute l'entreprise, pour authentifier et chiffrer les communications sur le site web interne. Les ingénieurs des opérations émettent des certificats d'entité finale qui authentifient les hôtes de centre de données et les services informatiques les uns auprès des autres. Ce système permet de sécuriser les données sensibles en les chiffrant sur le réseau local.

Définition des contraintes de longueur sur le parcours de certification

La structure d'une hiérarchie d'autorité de certification est définie et appliquée par l'extension de contraintes de base que chaque certificat contient. L'extension définit deux contraintes :

- `cA`— Si le certificat définit une autorité de certification. Si cette valeur est false (valeur par défaut), le certificat est un certificat d'entité finale.
- `pathLenConstraint`— Le nombre maximum d'autorités de certification subordonnées de niveau inférieur pouvant exister dans une chaîne de confiance valide. Le certificat d'entité finale n'est pas pris en compte car il ne s'agit pas d'un certificat CA.

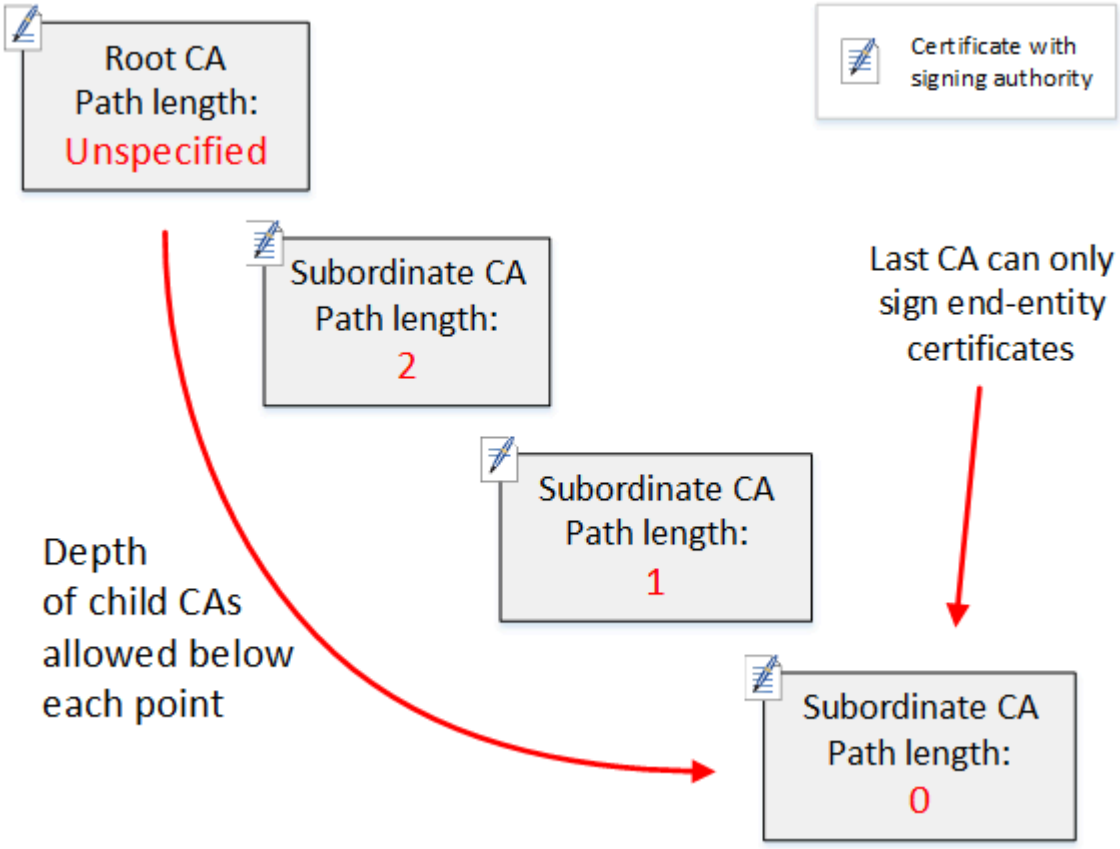
Un certificat d'une autorité de certification racine nécessite une flexibilité maximale et n'inclut pas de contrainte de longueur de chemin. Cela permet à la racine de définir un chemin de certification de n'importe quelle longueur.

> **Note**
>
> Autorité de certification privée AWS limite le parcours de certification à cinq niveaux.

Les autorités de certification subordonnées ont des valeurs `pathLenConstraint` égales ou supérieures à zéro, en fonction de l'emplacement dans la hiérarchie et des entités souhaitées. Par exemple, dans une hiérarchie avec trois autorités de certification, aucune contrainte de chemin n'est spécifiée pour l'autorité de certification racine. La première autorité de certification subordonnée a une longueur de chemin égale à 1 et peut donc signer des autorités de certification enfants. Chacune de ces autorités de certifications enfants doit nécessairement avoir une valeur `pathLenConstraint`

de zéro. Cela signifie qu'elles peuvent signer des certificats d'entité finale, mais ne peuvent pas émettre de certificats d'une autorité de certification supplémentaires. Limiter le pouvoir de créer de nouvelles autorités de certification est un contrôle de sécurité important.

Le diagramme suivant illustre cette propagation de l'autorité limitée vers le bas de la hiérarchie.



Dans cette hiérarchie à quatre niveaux, la racine est sans contrainte (comme toujours). Mais la première autorité de certification subordonnée a une valeur `pathLenConstraint` de 2, ce qui empêche ses autorités de certification enfants d'aller plus de deux niveaux plus loin. Par conséquent, pour un chemin de certification valide, la valeur de contrainte doit être décrémentée à zéro dans les deux niveaux suivants. Si un navigateur web rencontre un certificat d'entité finale de cette branche dont la longueur de chemin est supérieure à quatre, la validation échoue. Un tel certificat peut être le résultat d'une autorité de certification créée accidentellement, d'une autorité de certification mal configurée ou d'une émission non autorisée.

Gestion de la longueur des chemins à l'aide de modèles

Autorité de certification privée AWS fournit des modèles pour l'émission de certificats d'entité racine, subordonnée et finale. Ces modèles encapsulent les bonnes pratiques pour les valeurs de

contraintes de base, y compris la longueur du chemin d'accès. Les modèles incluent les éléments suivants :

- RootCACertificate/V1
- Certificat CA subordonné _ 0/V1 PathLen
- Certificat CA subordonné _ 1/V1 PathLen
- Certificat CA subordonné _ 2/V1 PathLen
- Certificat CA subordonné _ 3/V1 PathLen
- EndEntityCertificate/V1

L'API `IssueCertificate` renvoie une erreur si vous tentez de créer une autorité de certification avec une longueur de chemin supérieure ou égale à la longueur de chemin de son certificat d'une autorité de certification émettrice.

Pour de plus amples informations sur les modèles de certificats, veuillez consulter Comprendre les modèles de certificats.

Automatiser la configuration de la hiérarchie CA avec AWS CloudFormation

Une fois que vous avez défini une conception pour votre hiérarchie d'autorité de certification, vous pouvez la tester et la mettre en production à l'aide d'un AWS CloudFormation modèle. Pour obtenir un exemple de modèle de ce type, veuillez consulter Déclaration d'une hiérarchie d'autorité de certification privée dans le Guide de l'utilisateur AWS CloudFormation .

Gestion du cycle de vie de l'autorité de certification privée

Les certificats d'une autorité de certification ont une durée de vie ou une période de validité fixe. Lorsqu'un certificat d'une autorité de certification expire, tous les certificats émis directement ou indirectement par les autorités de certification subordonnées situées en dessous de celui-ci dans la hiérarchie de l'autorité de certification deviennent invalides. Vous pouvez éviter l'expiration du certificat d'une autorité de certification en planifiant à l'avance.

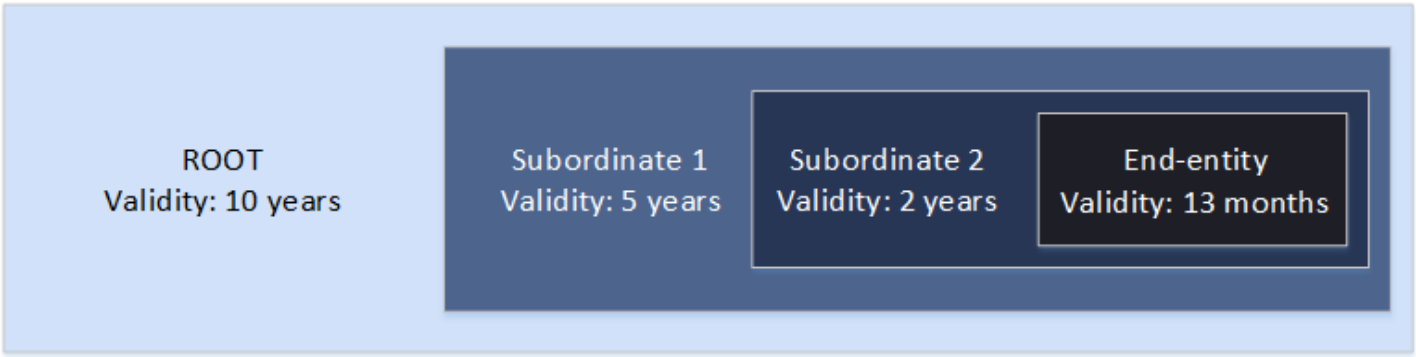
Choix des périodes de validité

La période de validité d'un certificat X.509 est un champ de certificat de base obligatoire. Elle détermine la période pendant laquelle l'autorité de certification émettrice certifie que le certificat peut être approuvé, sauf révocation. (Un certificat racine, auto-signé, certifie sa propre période de validité.)

Autorité de certification privée AWS et AWS Certificate Manager aider à configurer les périodes de validité des certificats sous réserve des contraintes suivantes :

- Un certificat géré par Autorité de certification privée AWS doit avoir une période de validité inférieure ou égale à la période de validité de l'autorité de certification qui l'a émis. En d'autres termes, les autorités de certification enfants et les certificats d'entité finale ne peuvent pas survivre à leurs certificats parents. La tentative d'utilisation de l'API `IssueCertificate` pour émettre un certificat d'une autorité de certification dont la période de validité est supérieure ou égale à l'autorité de certification du parent échoue.
- Les certificats émis et gérés par AWS Certificate Manager (ceux pour lesquels ACM génère la clé privée) ont une durée de validité de 13 mois (395 jours). ACM gère le processus de renouvellement de ces certificats. Si vous émettez Autorité de certification privée AWS des certificats directement, vous pouvez choisir n'importe quelle période de validité.

Le diagramme suivant montre une configuration type des périodes de validité imbriquées. Le certificat racine est la durée de vie la plus longue ; les certificats d'entité finale ont une durée de vie relativement courte ; et les autorités de certification subordonnées varient entre ces extrêmes.



Lorsque vous planifiez votre hiérarchie d'autorité de certification, déterminez la durée de vie optimale de vos certificats d'une autorité de certification. Travaillez en arrière à partir de la durée de vie souhaitée des certificats d'entité finale que vous souhaitez émettre.

Certificats d'entité finale

Les certificats d'entité finale doivent avoir une période de validité appropriée au cas d'utilisation. Une courte durée de vie minimise l'exposition d'un certificat en cas de perte ou de vol de sa clé privée. Cependant, des durées de vie courtes signifient des renouvellements fréquents. Le non-renouvellement d'un certificat expirant peut entraîner des temps d'arrêt.

L'utilisation distribuée de certificats d'entité finale peut également poser des problèmes logistiques en cas d'atteinte à la sécurité. Votre planification doit prendre en compte les certificats de renouvellement et de distribution, la révocation des certificats compromis et la rapidité avec laquelle les révocations se propagent aux clients qui s'appuient sur les certificats.

La période de validité par défaut d'un certificat d'entité finale délivré via ACM est de 13 mois (395 jours). Dans Autorité de certification privée AWS, vous pouvez utiliser l'`IssueCertificate`API pour appliquer n'importe quelle période de validité, à condition qu'elle soit inférieure à celle de l'autorité de certification émettrice.

Certificats d'une autorité de certification subordonnée

Les certificats d'une autorité de certification subordonnée doivent avoir des périodes de validité beaucoup plus longues que les certificats qu'ils délivrent. Une bonne fourchette pour la validité d'un certificat d'une autorité de certification est de deux à cinq fois la période de validité d'un certificat d'une autorité de certification enfant ou d'un certificat d'entité finale qu'il émet. Par exemple, supposons que vous disposez d'une hiérarchie d'autorité de certification à deux niveaux (autorité de certification racine et une autorité de certification subordonnée). Si vous souhaitez émettre des certificats d'entité finale d'une durée de vie d'un an, vous pouvez configurer la durée de vie de l'autorité de certification émettrice subordonnée à trois ans. Il s'agit de la période de validité par défaut pour un certificat CA subordonné dans Autorité de certification privée AWS. Les certificats d'une autorité de certification subordonnée peuvent être modifiés sans remplacer le certificat d'une autorité de certification racine.

Certificats racines d’autorité de certification

Les modifications apportées à un certificat d'une autorité de certification racine affectent l'ensemble de l'infrastructure à clé publique (infrastructure à clé publique) et vous obligent à mettre à jour tous les magasins d'approbation du système d'exploitation client dépendant et du navigateur. Pour minimiser l'impact opérationnel, vous devez choisir une longue période de validité pour le certificat racine. La durée Autorité de certification privée AWS par défaut pour les certificats racine est de dix ans.

Gestion de la succession des CA

Vous disposez de deux façons de gérer la succession d'autorité de certification : remplacer l'ancienne autorité de certification ou réémettre l'autorité de certification avec une nouvelle période de validité.

Remplacement d'un ancien CA

Pour remplacer une ancienne autorité de certification, vous créez une nouvelle autorité de certification et vous la chaînez à la même autorité de certification parent. Ensuite, vous émettez des certificats à partir de la nouvelle autorité de certification.

Les certificats émis à partir de la nouvelle autorité de certification ont une nouvelle chaîne d'autorité de certification. Une fois la nouvelle autorité de certification établie, vous pouvez désactiver l'ancienne autorité de certification pour l'empêcher d'émettre de nouveaux certificats. Lorsqu'elle est désactivée, l'ancienne autorité de certification prend en charge la révocation des anciens certificats émis par l'autorité de certification et, si elle est configurée pour ce faire, elle continue à valider les certificats au moyen d'OCSP et/ou de listes de révocation de certificats (CRL). Lorsque le dernier certificat émis à partir de l'ancienne autorité de certification expire, vous pouvez supprimer l'ancienne autorité de certification. Vous pouvez générer un rapport d'audit pour tous les certificats émis par l'autorité de certification afin de confirmer que tous les certificats émis ont expiré. Si l'ancienne autorité de certification a des autorités de certification subordonnées, vous devez également les remplacer, car les autorités de certification subordonnées expirent en même temps ou avant leur autorité de certification parent. Commencez par remplacer l'autorité de certification la plus élevée dans la hiérarchie qui doit être remplacée. Créez ensuite de nouvelles autorités de certification subordonnées de remplacement à chaque niveau inférieur suivant.

AWS recommande d'inclure un identifiant de génération de CA dans les noms des autorités de certification selon les besoins. Supposons, par exemple, que vous nommez l'autorité de certification de première génération « Corporate Root CA ». Lorsque vous créez l'autorité de certification de deuxième génération, nommez-la « Corporate Root CA G2 ». Cette convention de dénomination simple peut aider à éviter toute confusion lorsque les deux autorités de certification n'ont pas expiré.

Cette méthode de succession de CA est préférée car elle fait pivoter la clé privée de l'autorité de certification. La rotation de la clé privée est une bonne pratique pour les clés de CA. La fréquence de rotation devrait être proportionnelle à la fréquence d'utilisation des clés : les autorités de certification qui délivrent plus de certificats devraient être tournées plus fréquemment.

> **Note**
>
> Les certificats privés émis via ACM ne peuvent pas être renouvelés si vous remplacez l'autorité de certification. Si vous utilisez ACM pour l'émission et le renouvellement, vous devez réémettre le certificat CA pour prolonger la durée de vie de l'AC.

Réédition d'une ancienne CA

Lorsqu'une autorité de certification est sur le point d'expirer, une autre méthode pour prolonger sa durée de vie consiste à réémettre le certificat de l'autorité de certification avec une nouvelle date d'expiration. La réémission laisse toutes les métadonnées de l'autorité de certification en place et préserve les clés privées et publiques existantes. Dans ce scénario, la chaîne de certificats existante et les certificats d'entité finale non expirés émis par l'autorité de certification restent valides jusqu'à leur expiration. L'émission de nouveaux certificats peut également se poursuivre sans interruption. Pour mettre à jour une autorité de certification avec un certificat réémis, suivez les procédures d'installation habituelles décrites dansCréation et installation du certificat CA.

> **Note**
>
> Nous recommandons de remplacer une autorité de certification qui arrive à expiration plutôt que de réémettre son certificat en raison des avantages en termes de sécurité liés à la rotation vers une nouvelle paire de clés.

Révocation d'une autorité de certification

Vous révoquez une autorité de certification en révoquant son certificat sous-jacent. Cela révoque également efficacement tous les certificats émis par l'autorité de certification. Les informations de révocation sont distribuées aux clients au moyen d'un OCSP ou d'une CRL. Vous ne devez révoquer un certificat d'autorité de certification que si vous souhaitez révoquer tous les certificats d'entité finale et d'autorité de certification subordonnée qu'il a émis.

Configuration d'une méthode de révocation des certificats

Lorsque vous planifiez votre PKI privée Autorité de certification privée AWS, vous devez réfléchir à la manière de gérer les situations dans lesquelles vous ne souhaitez plus que les points de terminaison fassent confiance à un certificat émis, par exemple lorsque la clé privée d'un point de terminaison est révélée. Les approches les plus courantes pour résoudre ce problème consistent à utiliser des certificats de courte durée ou à configurer la révocation des certificats. Les certificats de courte durée expirent en si peu de temps, en heures ou en jours, que la révocation n'a aucun sens, le certificat devenant invalide à peu près au même moment qu'il faut pour informer un terminal de révocation. Cette section décrit les options de révocation pour les Autorité de certification privée AWS clients, y compris la configuration et les meilleures pratiques.

Les clients qui recherchent une méthode de révocation peuvent choisir le protocole OCSP (Online Certificate Status Protocol), les listes de révocation de certificats (CRL) ou les deux.

> **Note**
>
> Si vous créez votre autorité de certification sans configurer la révocation, vous pourrez toujours la configurer ultérieurement. Pour plus d'informations, consultez Mettre à jour votre CA privée.

- Protocole d'état des certificats en ligne (OCSP)

  Autorité de certification privée AWS fournit une solution OCSP entièrement gérée pour informer les points de terminaison que les certificats ont été révoqués sans que les clients aient à exploiter eux-mêmes l'infrastructure. Les clients peuvent activer l'OCSP sur des autorités de certification nouvelles ou existantes en une seule opération à l'aide de la Autorité de certification privée AWS console, de l'API, de la CLI ou via AWS CloudFormation. Alors que les CRL sont stockées et traitées sur le terminal et peuvent devenir obsolètes, les exigences de stockage et de traitement OCSP sont gérées de manière synchrone sur le backend du répondeur.

  Lorsque vous activez l'OCSP pour une autorité de certification, Autorité de certification privée AWS incluez l'URL du répondeur OCSP dans l'extension Authority Information Access (AIA) de chaque nouveau certificat émis. L'extension permet aux clients tels que les navigateurs Web d'interroger le répondeur et de déterminer si un certificat d'entité finale ou d'autorité de certification subordonnée est fiable. Le répondeur renvoie un message d'état signé cryptographiquement pour garantir son authenticité.

  Le répondeur Autorité de certification privée AWS OCSP est conforme à la RFC 5019.

  Considérations relatives à l'OCSP

  - Les messages d'état OCSP sont signés à l'aide du même algorithme de signature que celui pour lequel l'autorité de certification émettrice a été configurée. Les autorités de certification créées dans la Autorité de certification privée AWS console utilisent l'algorithme de signature SHA256WITHRSA par défaut. Les autres algorithmes pris en charge sont disponibles dans la documentation de CertificateAuthorityConfigurationl'API.
  - Les modèles de certificats APIPassthrough et CSRPassthrough ne fonctionneront pas avec l'extension AIA si le répondeur OCSP est activé.

  - Le point de terminaison du service OCSP géré est accessible sur l'Internet public. Les clients qui veulent un OCSP mais préfèrent ne pas avoir de point de terminaison public devront exploiter leur propre infrastructure OCSP.
- Listes de révocation de certificats (CRL)

  Une CRL contient une liste de certificats révoqués. Lorsque vous configurez une autorité de certification pour générer des CRL, Autorité de certification privée AWS incluez l'extension CRL Distribution Points dans chaque nouveau certificat émis. Cette extension fournit l'URL de la CRL. L'extension permet aux clients tels que les navigateurs Web d'interroger la CRL et de déterminer si un certificat d'entité finale ou de CA subordonné est fiable.

Comme un client doit télécharger les CRL et les traiter localement, leur utilisation est plus gourmande en mémoire que l'OCSP. Les CRL peuvent consommer moins de bande passante réseau car la liste des CRL est téléchargée et mise en cache, par rapport à l'OCSP qui vérifie l'état de révocation à chaque nouvelle tentative de connexion.

> **Note**
>
> L'OCSP et les CRL présentent un certain délai entre la révocation et la disponibilité du changement de statut.
>
> - Les réponses OCSP peuvent prendre jusqu'à 60 minutes pour refléter le nouveau statut lorsque vous révoquez un certificat. En général, l'OCSP a tendance à accélérer la distribution des informations de révocation car, contrairement aux CRL qui peuvent être mises en cache par les clients pendant des jours, les réponses OCSP ne sont généralement pas mises en cache par les clients.
> - Une liste de révocation de certificats est généralement mise à jour environ 30 minutes après la révocation d'un certificat. Si, pour une raison quelconque, une mise à jour de la CRL échoue, Autorité de certification privée AWS effectuez de nouvelles tentatives toutes les 15 minutes.

Exigences générales relatives aux configurations de révocation

Les exigences suivantes s'appliquent à toutes les configurations de révocation.

- Une configuration désactivant les CRL ou OCSP doit contenir uniquement le paramètre `Enabled=False` et échouera si d'autres paramètres tels que `CustomCname` ou `ExpirationInDays` sont inclus.
- Dans une configuration CRL, le `S3BucketName` paramètre doit être conforme aux règles de dénomination des compartiments Amazon Simple Storage Service.
- Une configuration contenant un paramètre de nom canonique (CNAME) personnalisé pour les CRL ou les OCSP doit être conforme aux restrictions de la RFC7230 relatives à l'utilisation de caractères spéciaux dans un CNAME.
- Dans une configuration CRL ou OCSP, la valeur d'un paramètre CNAME ne doit pas inclure de préfixe de protocole tel que « http:// » ou « https:// ».

Rubriques

Planification d'une liste de révocation de certificats (CRL)

Avant de pouvoir configurer une CRL dans le cadre du processus de création de l'autorité de certification, une configuration préalable peut être nécessaire. Cette section explique les prérequis et les options que vous devez comprendre avant de créer une autorité de certification associée à une CRL.

Pour plus d'informations sur l'utilisation du protocole OCSP (Online Certificate Status Protocol) comme alternative ou complément à une CRL, consultez Options de révocation des certificats et. Configuration d'une URL personnalisée pour Autorité de certification privée AWS OCSP

Rubriques

Structure CRL

Chaque liste est un fichier codé DER. Pour télécharger le fichier et utiliser OpenSSL pour l'afficher, utilisez une commande similaire à la suivante :

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

Les listes de révocation de certificats sont au format suivant :

```
Certificate Revocation List (CRL):
          Version 2 (0x1)
      Signature Algorithm: sha256WithRSAEncryption
          Issuer: /C=US/ST=WA/L=Seattle/O=Example Company CA/OU=Corporate/
CN=www.example.com
          Last Update: Feb 26 19:28:25 2018 GMT
          Next Update: Feb 26 20:28:25 2019 GMT
          CRL extensions:
              X509v3 Authority Key Identifier:
                  keyid:AA:6E:C1:8A:EC:2F:8F:21:BC:BE:80:3D:C5:65:93:79:99:E7:71:65

              X509v3 CRL Number:
                  1519676905984
  Revoked Certificates:
      Serial Number: E8CBD2BEDB122329F97706BCFEC990F8
          Revocation Date: Feb 26 20:00:36 2018 GMT
          CRL entry extensions:
              X509v3 CRL Reason Code:
                  Key Compromise
      Serial Number: F7D7A3FD88B82C6776483467BBF0B38C
          Revocation Date: Jan 30 21:21:31 2018 GMT
          CRL entry extensions:
              X509v3 CRL Reason Code:
                  Key Compromise
      Signature Algorithm: sha256WithRSAEncryption
           82:9a:40:76:86:a5:f5:4e:1e:43:e2:ea:83:ac:89:07:49:bf:
           c2:fd:45:7d:15:d0:76:fe:64:ce:7b:3d:bb:4c:a0:6c:4b:4f:
           9e:1d:27:f8:69:5e:d1:93:5b:95:da:78:50:6d:a8:59:bb:6f:
           49:9b:04:fa:38:f2:fc:4c:0d:97:ac:02:51:26:7d:3e:fe:a6:
           c6:83:34:b4:84:0b:5d:b1:c4:25:2f:66:0a:2e:30:f6:52:88:
           e8:d2:05:78:84:09:01:e8:9d:c2:9e:b5:83:bd:8a:3a:e4:94:
           62:ed:92:e0:be:ea:d2:59:5b:c7:c3:61:35:dc:a9:98:9d:80:
           1c:2a:f7:23:9b:fe:ad:6f:16:7e:22:09:9a:79:8f:44:69:89:
           2a:78:ae:92:a4:32:46:8d:76:ee:68:25:63:5c:bd:41:a5:5a:
```

```
        57:18:d7:71:35:85:5c:cd:20:28:c6:d5:59:88:47:c9:36:44:
        53:55:28:4d:6b:f8:6a:00:eb:b4:62:de:15:56:c8:9c:45:d7:
        83:83:07:21:84:b4:eb:0b:23:f2:61:dd:95:03:02:df:0d:0f:
        97:32:e0:9d:38:de:7c:15:e4:36:66:7a:18:da:ce:a3:34:94:
        58:a6:5d:5c:04:90:35:f1:8b:55:a9:3c:dd:72:a2:d7:5f:73:
        5a:2c:88:85
```

> **Note**
>
> La CRL ne sera déposée dans Amazon S3 qu'après l'émission d'un certificat y faisant référence. Avant cela, seul un `acm-pca-permission-test-key` fichier sera visible dans le compartiment Amazon S3.

Politiques d'accès pour les CRL dans Amazon S3

Si vous envisagez de créer une CRL, vous devez préparer un compartiment Amazon S3 dans lequel la stocker. Autorité de certification privée AWS dépose automatiquement la CRL dans le compartiment Amazon S3 que vous désignez et la met à jour régulièrement. Pour de plus amples informations, veuillez consulter Créer un compartiment.

Votre compartiment S3 doit être sécurisé par une politique d'autorisation IAM attachée. Les utilisateurs autorisés et les responsables du service doivent être `Put` autorisés Autorité de certification privée AWS à placer des objets dans le compartiment et `Get` à les récupérer. Au cours de la procédure de création d'une autorité de certification dans la console, vous pouvez choisir d'autoriser la Autorité de certification privée AWS création d'un nouveau compartiment et d'appliquer une politique d'autorisation par défaut.

> **Note**
>
> La configuration de la politique IAM dépend de l'acteur Régions AWS concerné. Les régions se répartissent en deux catégories :
>
> - Régions activées par défaut : régions activées par défaut pour tous. Comptes AWS
> - Régions désactivées par défaut : régions désactivées par défaut, mais qui peuvent être activées manuellement par le client.

Pour plus d'informations et une liste des régions désactivées par défaut, consultez la section Gestion. Régions AWS Pour une discussion sur les principes de service dans le contexte de l'IAM, voir les principes de AWS service dans les régions optionnelles.
Lorsque vous configurez les CRL comme méthode de révocation des certificats, vous Autorité de certification privée AWS créez une CRL et la publiez dans un compartiment S3. Le compartiment S3 nécessite une politique IAM qui permet au principal du Autorité de certification privée AWS service d'écrire dans le compartiment. Le nom du principal de service varie en fonction des régions utilisées, et toutes les possibilités ne sont pas prises en charge.

PCA	S3	Principal du service
Les deux dans la même région		`acm-pca.amazonaws.com`
Activées	Activé	`acm-pca.amazonaws.com`
Désactivées	Activées	`acm-pca.Region.amazonaws.com`
Activé	Désactivées	Non pris en charge

La politique par défaut n'applique aucune `SourceArn` restriction à l'autorité de certification. Nous vous recommandons d'appliquer manuellement la politique moins permissive décrite ci-dessous, qui restreint l'accès à un AWS compte spécifique et à une autorité de certification privée spécifique. Pour plus d'informations, consultez Ajouter une politique de compartiment à l'aide de la console Amazon S3.

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

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

Si vous choisissez d'autoriser la politique par défaut, vous pourrez toujours la modifier ultérieurement.

Activation de l'accès public par blocs S3 (BPA) avec CloudFront

Les nouveaux compartiments Amazon S3 sont configurés par défaut avec la fonctionnalité Block Public Access (BPA) activée. Inclus dans les meilleures pratiques de sécurité d'Amazon S3, le BPA est un ensemble de contrôles d'accès que les clients peuvent utiliser pour affiner l'accès aux objets de leurs compartiments S3 et aux compartiments dans leur ensemble. Lorsque le BPA est actif et correctement configuré, seuls les AWS utilisateurs autorisés et authentifiés ont accès à un bucket et à son contenu.

AWS recommande l'utilisation du BPA sur tous les compartiments S3 afin d'éviter d'exposer des informations sensibles à des adversaires potentiels. Cependant, une planification supplémentaire est requise si vos clients PKI récupèrent des CRL sur Internet public (c'est-à-dire lorsqu'ils ne sont pas connectés à un AWS compte). Cette section décrit comment configurer une solution PKI privée à l'aide d'Amazon CloudFront, un réseau de diffusion de contenu (CDN), pour servir des CRL sans avoir besoin d'un accès client authentifié à un compartiment S3.

> **Note**
>
> L'utilisation CloudFront entraîne des frais supplémentaires sur votre AWS compte. Pour plus d'informations, consultez Amazon CloudFront Pricing.
> Si vous choisissez de stocker votre CRL dans un compartiment S3 où le BPA est activé et que vous ne l'utilisez pas CloudFront, vous devez créer une autre solution CDN pour garantir que votre client PKI a accès à votre CRL.

Configurer Amazon S3 avec le BPA

Dans S3, créez un nouveau compartiment pour votre CRL, comme d'habitude, puis activez le BPA dessus.

Pour configurer un compartiment Amazon S3 qui bloque l'accès public à votre CRL

1. Créez un nouveau compartiment S3 en suivant la procédure décrite dans Création d'un compartiment. Au cours de la procédure, sélectionnez l'option Bloquer tout accès public.

   Pour plus d'informations, consultez Blocage de l'accès public à votre espace de stockage Amazon S3.
2. Une fois le compartiment créé, choisissez son nom dans la liste, accédez à l'onglet Autorisations, choisissez Modifier dans la section Propriété de l'objet, puis sélectionnez Propriétaire du compartiment préféré.
3. Également dans l'onglet Autorisations, ajoutez une politique IAM au compartiment, comme décrit dansPolitiques d'accès pour les CRL dans Amazon S3 .

Configuration CloudFront pour le BPA

Créez une CloudFront distribution qui aura accès à votre compartiment S3 privé et pourra servir des CRL à des clients non authentifiés.

Pour configurer une CloudFront distribution pour la CRL

1. Créez une nouvelle CloudFront distribution en suivant la procédure décrite dans la section Création d'une distribution du manuel Amazon CloudFront Developer Guide.

   Au cours de la procédure, appliquez les paramètres suivants :

- Dans Origin Domain Name, choisissez votre compartiment S3.
- Choisissez Oui pour Restreindre l'accès au bucket.
- Choisissez Créer une nouvelle identité pour Origin Access Identity.
- Choisissez Yes, Update Bucket Policy sous Accorder les autorisations de lecture sur le bucket.

> **Note**
>
> Dans cette procédure, CloudFront modifie votre politique de compartiment pour lui permettre d'accéder aux objets du compartiment. Envisagez de modifier cette politique afin de n'autoriser l'accès qu'aux objets situés `crl` dans le dossier.

2. Une fois la distribution initialisée, recherchez son nom de domaine dans la CloudFront console et enregistrez-le pour la procédure suivante.

> **Note**
>
> Si votre compartiment S3 vient d'être créé dans une région autre que us-east-1, vous risquez de recevoir une erreur de redirection temporaire HTTP 307 lorsque vous accédez à votre application publiée via. CloudFront La propagation de l'adresse du bucket peut prendre plusieurs heures.

Configurez votre CA pour le BPA

Lors de la configuration de votre nouvelle autorité de certification, incluez l'alias dans votre CloudFront distribution.

Pour configurer votre autorité de certification avec un CNAME pour CloudFront

- Créez votre CA à l'aide deProcédure de création d'une autorité de certification (CLI) .

  Lorsque vous exécutez la procédure, le fichier de révocation `revoke_config.txt` doit inclure les lignes suivantes pour spécifier un objet CRL non public et pour fournir une URL vers le point de terminaison de distribution dans : CloudFront

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

Par la suite, lorsque vous émettrez des certificats avec cette autorité de certification, ils contiendront un bloc comme celui-ci :

```
X509v3 CRL Distribution Points:
 Full Name:
 URI:http://abcdef012345.cloudfront.net/crl/01234567-89ab-
cdef-0123-456789abcdef.crl
```

> **Note**
>
> Si vous possédez d'anciens certificats émis par cette autorité de certification, elle ne pourra pas accéder à la CRL.

Chiffrement de vos listes de révocation de certificats

Vous pouvez éventuellement configurer le chiffrement sur le compartiment Amazon S3 contenant vos CRL. Autorité de certification privée AWS prend en charge deux modes de chiffrement pour les actifs dans Amazon S3 :

- Chiffrement automatique côté serveur avec des clés AES-256 gérées par Amazon S3.
- Chiffrement géré par le client à l'aide AWS Key Management Service et AWS KMS key configuré selon vos spécifications.

> **Note**
>
> Autorité de certification privée AWS ne prend pas en charge l'utilisation de clés KMS par défaut générées automatiquement par S3.

Les procédures suivantes décrivent comment configurer chacune des options de chiffrement.

Pour configurer le chiffrement automatique

Procédez comme suit pour activer le chiffrement côté serveur S3.

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.

2. Dans le tableau Buckets, choisissez le compartiment qui contiendra vos Autorité de certification privée AWS actifs.
3. Sur la page de votre compartiment, choisissez l'onglet Propriétés .
4. Choisissez la carte de Chiffrement par défaut.
5. Sélectionnez Activer.
6. Choisissez la clé Amazon S3 (SSE-S3).
7. Choisissez Save Changes (Enregistrer les modifications).

Pour configurer le chiffrement personnalisé

Procédez comme suit pour activer le chiffrement à l'aide d'une clé personnalisée.

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Dans le tableau Buckets, choisissez le compartiment qui contiendra vos Autorité de certification privée AWS actifs.
3. Sur la page de votre compartiment, choisissez l'onglet Propriétés .
4. Choisissez la carte de Chiffrement par défaut.
5. Sélectionnez Activer.
6. Choisissez AWS Key Management Service la clé (SSE-KMS).
7. Choisissez Choisir parmi vos AWS KMS clés ou Entrer un AWS KMS key ARN.
8. Choisissez Save Changes (Enregistrer les modifications).
9. (Facultatif) Si vous ne possédez pas encore de clé KMS, créez-en une à l'aide de la commande AWS CLI create-key suivante :

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

La sortie contient l'ID de clé et le nom de ressource Amazon (ARN) de la clé KMS. Voici un exemple de résultat :

```
{
    "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. En suivant les étapes suivantes, vous autorisez le principal du Autorité de certification privée AWS service à utiliser la clé KMS. Par défaut, toutes les clés KMS sont privées ; seul le propriétaire de la ressource peut utiliser une clé KMS pour chiffrer et déchiffrer les données. Cependant, le propriétaire de la ressource peut accorder à d'autres utilisateurs et ressources des autorisations d'accès à la clé KMS. Le principal de service doit se trouver dans la même région que celle où la clé KMS est stockée.

    a. Tout d'abord, enregistrez la politique par défaut pour votre clé KMS à `policy.json` l'aide de la get-key-policycommande suivante :

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

    b. Ouvrez le fichier `policy.json` dans un éditeur de texte. Sélectionnez l'une des déclarations de politique suivantes et ajoutez-la à la politique existante.

    Si votre clé de compartiment Amazon S3 est activée, utilisez l'instruction suivante :

    ```
    {
       "Sid":"Allow ACM-PCA use of the key",
       "Effect":"Allow",
       "Principal":{
          "Service":"acm-pca.amazonaws.com"
       },
       "Action":[
          "kms:GenerateDataKey",
          "kms:Decrypt"
       ],
       "Resource":"*",
       "Condition":{
          "StringLike":{
             "kms:EncryptionContext:aws:s3:arn":"arn:aws:s3:::bucket-name"
          }
       }
    ```

```
}
```

Si votre clé de compartiment Amazon S3 est désactivée, utilisez l'instruction suivante :

```
{
   "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. Enfin, appliquez la politique mise à jour à l'aide de la put-key-policycommande suivante :

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

Déterminer l'URI du point de distribution CRL (CDP)

Si vous utilisez le compartiment S3 comme CDP pour votre autorité de certification, l'URI du CDP peut être dans l'un des formats suivants.

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

Si vous avez configuré votre autorité de certification avec un CNAME personnalisé, l'URI du CDP inclura le CNAME, par exemple, `http://alternative.example.com/crl/CA-ID.crl`

Configuration d'une URL personnalisée pour Autorité de certification privée AWS OCSP

> **Note**
>
> Cette rubrique s'adresse aux clients qui souhaitent personnaliser l'URL publique du point de terminaison du répondeur OCSP à des fins de marque ou à d'autres fins. Si vous envisagez d'utiliser la configuration par défaut de l'OCSP Autorité de certification privée AWS géré, vous pouvez ignorer cette rubrique et suivre les instructions de configuration de la section Configurer la révocation.

Par défaut, lorsque vous activez OCSP pour Autorité de certification privée AWS, chaque certificat que vous émettez contient l'URL du répondeur AWS OCSP. Cela permet aux clients demandant une connexion sécurisée par chiffrement d'envoyer directement des requêtes de validation OCSP à. AWS Cependant, dans certains cas, il peut être préférable d'indiquer une URL différente dans vos certificats tout en soumettant des requêtes OCSP à AWS.

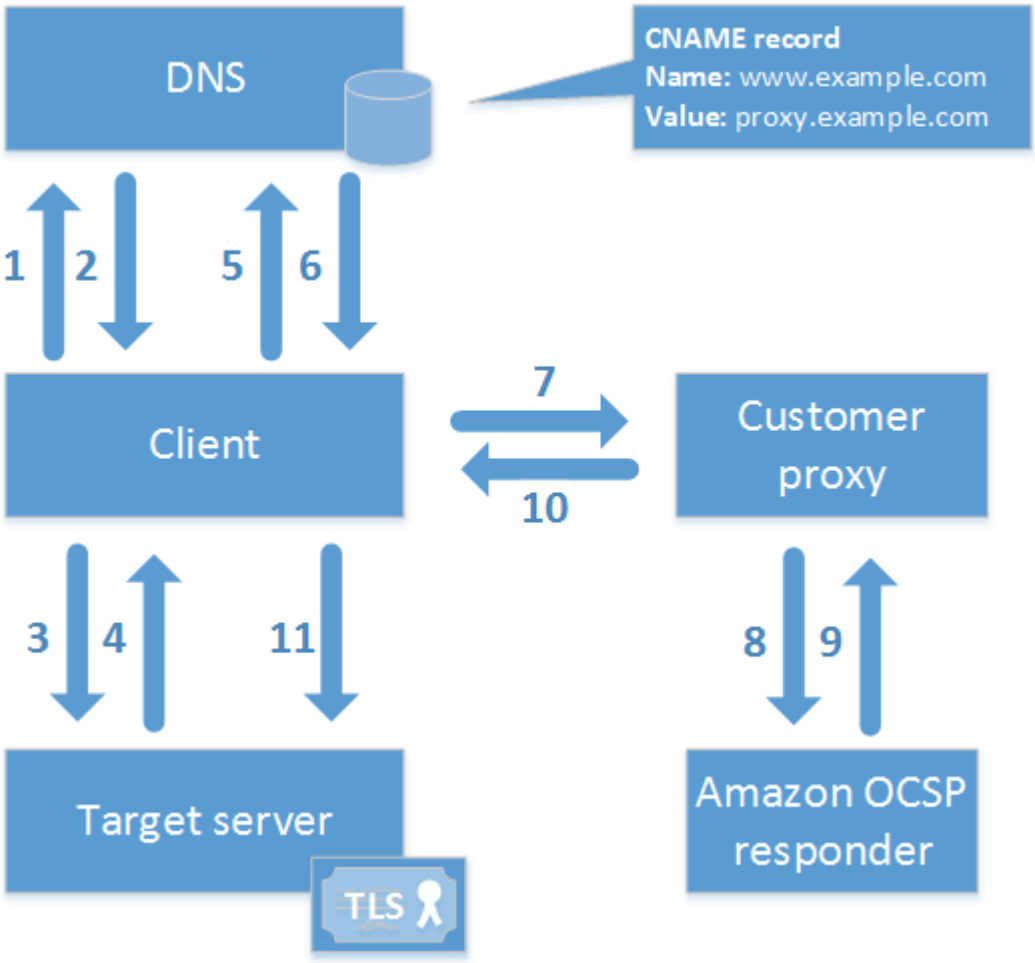
> **Note**
>
> Pour plus d'informations sur l'utilisation d'une liste de révocation de certificats (CRL) comme alternative ou complément à l'OCSP, voir Configurer la révocation et Planifier une liste de révocation de certificats (CRL).

Trois éléments sont impliqués dans la configuration d'une URL personnalisée pour OCSP.

- Configuration de l'autorité de certification — Spécifiez une URL OCSP personnalisée dans le `RevocationConfiguration` pour votre autorité de certification, comme décrit Exemple 2 : créer une autorité de certification avec OCSP et un CNAME personnalisé activés dansProcédure de création d'une autorité de certification (CLI) .
- DNS — Ajoutez un enregistrement CNAME à la configuration de votre domaine pour mapper l'URL figurant dans les certificats à l'URL d'un serveur proxy. Pour plus d'informations, consultez Exemple 2 : créer une autorité de certification avec OCSP et un CNAME personnalisé activés dans Procédure de création d'une autorité de certification (CLI) .

- Transfert de serveur proxy : configurez un serveur proxy capable de transférer de manière transparente le trafic OCSP qu'il reçoit au répondeur AWS OCSP.

Le schéma suivant illustre la façon dont ces éléments fonctionnent ensemble.



Comme le montre le schéma, le processus de validation OCSP personnalisé comprend les étapes suivantes :

1. Le client interroge le DNS pour le domaine cible.
2. Le client reçoit l'adresse IP cible.
3. Le client ouvre une connexion TCP avec la cible.
4. Le client reçoit le certificat TLS cible.
5. Le client interroge le DNS pour le domaine OCSP répertorié dans le certificat.
6. Le client reçoit l'adresse IP du proxy.
7. Le client envoie une requête OCSP au proxy.

8. Le proxy transmet la requête au répondeur OCSP.
9. Le répondeur renvoie l'état du certificat au proxy.
10. Le proxy transmet l'état du certificat au client.
11. Si le certificat est valide, le client lance le handshake TLS.

> **Tip**
>
> Cet exemple peut être implémenté à l'aide d'Amazon CloudFront et d'Amazon Route 53 une fois que vous avez configuré une autorité de certification comme décrit ci-dessus.
>
> 1. Dans CloudFront, créez une distribution et configurez-la comme suit :
>   - Créez un autre nom correspondant à votre CNAME personnalisé.
>   - Liez votre certificat à celui-ci.
>   - Définissez ocsp.acm-pca. <region>.amazonaws.com comme origine.
>   - Appliquez la `Managed-CachingDisabled` politique.
>   - Définissez la politique du protocole Viewer sur HTTP et HTTPS.
>   - Définissez les méthodes HTTP autorisées sur GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE.
> 2. Dans Route 53, créez un enregistrement DNS qui fait correspondre votre CNAME personnalisé à l'URL de la CloudFront distribution.

Modes d'autorité de certification

Autorité de certification privée AWS prend en charge la création d'une autorité de certification dans l'un des deux modes. Les modes GENERAL_PURPOSE et SHORT_LIVED_CERTIFICATE affectent la période de validité autorisée des certificats émis par l'autorité de certification.

> **Note**
>
> Autorité de certification privée AWS n'effectue pas de contrôles de validité sur les certificats de l'autorité de certification racine.

GENERAL_PURPOSE (par défaut)

Ce mode permet à l'autorité de certification de délivrer des certificats avec n'importe quelle période de validité. La plupart des applications utilisent des certificats de ce type. Généralement, l'autorité de certification spécifie également un mécanisme de révocation.

CERTIFICAT_ÉPHÉMÈRE

Ce mode définit une autorité de certification qui émet exclusivement des certificats dont la durée de validité maximale est de sept jours. Ces certificats de courte durée expirent si rapidement qu'ils peuvent être déployés sans qu'un mécanisme de révocation soit en place. Pour certaines applications, il est plus judicieux de déployer fréquemment des certificats de courte durée plutôt que d'encourir les frais de réseau et de traitement liés à la révocation.

Les autorités de certification dotées du mode SHORT_LIVED_CERTIFICATE coûtent moins cher que les autorités de certification à usage général. Pour plus d'informations, consultez la section AWS Private Certificate Authority Tarification.

Pour créer une autorité de certification qui émet des certificats de courte durée, définissez le `UsageMode` paramètre sur SHORT_LIVED_CERTIFICATE en suivant la AWS CLIprocédure de création d'une autorité de certification.

> **Note**
>
> AWS Certificate Manager ne peut pas émettre de certificats signés par une autorité de certification privée en mode éphémère.

L'utilisation de certificats de courte durée est prise en charge par les AWS services suivants :

- Amazon AppStream
- Amazon WorkSpaces

Planification de la résilience

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones

de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Redondance et reprise après sinistre

Tenez compte de la redondance et de la reprise après sinistre lors de la planification de votre hiérarchie d'autorité de certification. Autorité de certification privée AWS est disponible dans plusieurs régions, ce qui vous permet de créer des autorités de certification redondantes dans plusieurs régions. Le Autorité de certification privée AWS service fonctionne avec un accord de niveau de service (SLA) garantissant une disponibilité de 99,9 %. Il existe au moins deux approches que vous pouvez envisager pour la redondance et la reprise après sinistre. Vous pouvez configurer la redondance à l'autorité de certification racine ou à l'autorité de certification subordonnée la plus élevée. Chaque approche a des avantages et des inconvénients.

1. Vous pouvez créer deux autorités de certification racines dans deux AWS régions différentes à des fins de redondance et de reprise après sinistre. Avec cette configuration, chaque autorité de certification racine fonctionne de manière indépendante dans une AWS région, vous protégeant ainsi en cas de sinistre survenant dans une seule région. La création d'autorités de certification racine redondantes augmente toutefois la complexité opérationnelle : vous devrez distribuer les deux certificats d'une autorité de certification racine aux magasins de confiance des navigateurs et des systèmes d'exploitation de votre environnement.
2. Vous pouvez également créer des autorités de certification subordonnées redondantes à déployer dans chacune de vos AWS régions et les associer à la même autorité de certification racine unique dans une seule AWS région. L'avantage de cette approche est que vous devez distribuer un seul certificat d'une autorité de certification racine aux magasins de confiance de votre environnement. La limite est que vous ne disposez pas d'une autorité de certification racine redondante en cas de sinistre affectant la AWS région dans laquelle se trouve votre autorité de certification racine.

Bonnes pratiques Autorité de certification privée AWS

Les bonnes pratiques sont des recommandations qui peuvent vous aider à utiliser efficacement Autorité de certification privée AWS. Les bonnes pratiques suivantes reposent sur l'expérience réelle de clients actuels AWS Certificate Manager et Autorité de certification privée AWS.

Documenter la structure et les politiques de l'AC

AWS recommande de documenter toutes vos politiques et pratiques pour l'exploitation de votre autorité de certification. Cela pourrait inclure :

- Raisonnement de vos décisions sur la structure de l'autorité de certification
- Un diagramme montrant vos autorités de certification et leurs relations
- Politiques sur les périodes de validité de l'autorité de certification
- Planification de la relève de l'autorité de certification
- Stratégies sur la longueur du chemin d'accès
- Catalogue des autorisations
- Description des structures de contrôle administratif
- Sécurité

Vous pouvez saisir ces informations dans deux documents, appelés Politique de certification (CP) et Déclaration des pratiques de certification (CPS). Reportez-vous à RFC 3647 pour obtenir un cadre permettant de capturer des informations importantes sur vos opérations de l'autorité de certification.

Minimisez l'utilisation de l'autorité de certification racine si possible

En général, une autorité de certification racine ne doit être utilisée que pour émettre des certificats pour les autorités de certification intermédiaires. Cela permet à l'autorité de certification racine d'être stockée hors du danger, tandis que les autorités de certification intermédiaires effectuent la tâche quotidienne d'émettre des certificats d'entité finale.

Toutefois, si la pratique actuelle de votre organisation consiste à émettre des certificats d'entité finale directement à partir d'une autorité de certification racine, Autorité de certification privée AWS peut prendre en charge ce flux de travail tout en améliorant les contrôles de sécurité et d'exploitation. L'émission de certificats d'entité finale dans ce scénario exige une stratégie d'autorisations IAM qui

permet à votre autorité de certification racine d'utiliser un modèle de certificat d'entité finale. Pour obtenir des informations sur les stratégies IAM, veuillez consulter Identity and Access Management (IAM) pour AWS Private Certificate Authority.

> **Note**
>
> Cette configuration impose des limites qui peuvent entraîner des problèmes opérationnels. Par exemple, si votre autorité de certification racine est compromise ou perdue, vous devez créer une nouvelle autorité de certification racine et la distribuer à tous les clients de votre environnement. Tant que ce processus de récupération n'est pas terminé, vous ne pourrez pas émettre de nouveaux certificats. L'émission de certificats directement à partir d'une autorité de certification racine vous empêche également de restreindre l'accès et de limiter le nombre de certificats émis à partir de votre racine, qui sont toutes deux considérées comme des bonnes pratiques pour la gestion d'une autorité de certification racine.

Donnez à l'autorité de certification racine la sienne Compte AWS

La création d'une autorité de certification racine et d'une autorité de certification subordonnée dans deux AWS comptes différents est une bonne pratique recommandée. Cela peut vous fournir une protection et des contrôles d'accès supplémentaires pour votre autorité de certification racine. Vous pouvez le faire en exportant la demande de signature de certificat à partir de l'autorité de certification subordonnée dans un compte et en la signant avec une autorité de certification racine dans un autre compte. L'avantage de cette approche est que vous pouvez séparer le contrôle de vos autorités de certification par compte. L'inconvénient est que vous ne pouvez pas utiliser l'AWS Management Consoleassistant pour simplifier le processus de signature du certificat de l'autorité de certification d'une autorité de certification subordonnée à partir de votre autorité de certification racine.

> **Important**
>
> Nous vous recommandons vivement d'utiliser l'authentification multifactorielle (MFA) à chaque fois que vous y accédez. Autorité de certification privée AWS

Rôles d'administrateur et d'émetteur distincts

Le rôle d'administrateur de l'autorité de certification doit être distinct de celui des utilisateurs qui doivent uniquement émettre des certificats d'entité finale. Si l'administrateur de l'autorité de

certification et l'émetteur du certificat résident dans le même Compte AWS établissement, vous pouvez limiter les autorisations de l'émetteur en créant un utilisateur IAM spécialement à cette fin.

Mettre en œuvre la révocation gérée des certificats

La révocation gérée informe automatiquement les clients du certificat lorsqu'un certificat a été révoqué. Vous devrez peut-être révoquer un certificat si ses informations cryptographiques ont été compromises ou s'il a été émis par erreur. Les clients refusent généralement d'accepter les certificats révoqués. Autorité de certification privée AWSpropose deux options standard pour la gestion des révocations : le protocole OCSP (Online Certificate Status Protocol) et les listes de révocation de certificats (CRL). Pour plus d'informations, consultez Configuration d'une méthode de révocation des certificats.

Activer AWS CloudTrail

Activez la CloudTrail journalisation avant de créer et de commencer à exploiter une autorité de certification privée. Vous pouvez ainsi récupérer l'historique des appels d'AWSAPI pour votre compte afin de surveiller vos AWS déploiements. CloudTrail Cet historique inclut les appels d'API effectués à partir desAWS Management Console, des AWS SDK, des services et des AWS Command Line Interface services de niveau supérieurAWS. Vous pouvez également identifier les utilisateurs et les comptes qui ont appelé les opérations d'API PCA, l'adresse IP source d'origine des appels, ainsi que le moment où les appels ont eu lieu. Vous pouvez CloudTrail intégrer des applications à l'aide de l'API, automatiser la création de traces pour votre organisation, vérifier l'état de vos pistes et contrôler la manière dont les administrateurs activent et désactivent la CloudTrail connexion. Pour plus d'informations, consultez Création d'un journal d'activité. Accédez à En utilisant CloudTrail pour consulter des exemples de journal de suivi pour des opérations Autorité de certification privée AWS.

Faites pivoter la clé privée CA

Il est recommandé de mettre à jour régulièrement la clé privée de votre autorité de certification privée. Vous pouvez mettre à jour une clé en important un nouveau certificat d'une autorité de certification ou en remplaçant l'autorité de certification privée par une nouvelle autorité de certification.

> **Note**
>
> Si vous remplacez l'autorité de certification elle-même, sachez que l'ARN de l'autorité de certification change. Cela entraînerait l'échec de l'automatisation basée sur un ARN codé en dur.

Supprimer les autorités de certification non utilisées

Vous pouvez supprimer définitivement une autorité de certification privée. Notamment si vous n'avez plus besoin de l'autorité de certification ou si vous souhaitez la remplacer par une autorité de certification avec une clé privée plus récente. Pour supprimer en toute sécurité une autorité de certification, nous vous conseillons de suivre le processus décrit dans Supprimer votre CA privée.

> **Note**
>
> AWS vous facture pour une autorité de certification jusqu'à sa suppression.

Bloquez l'accès public à vos CRL

Autorité de certification privée AWSrecommande d'utiliser la fonctionnalité Block Public Access (BPA) d'Amazon S3 sur les compartiments contenant des CRL. Cela évite d'exposer inutilement les détails de votre PKI privée à des adversaires potentiels. Le BPA est une bonne pratique S3 et est activé par défaut sur les nouveaux buckets. Une configuration supplémentaire est nécessaire dans certains cas. Pour plus d'informations, consultez Activation de l'accès public par blocs S3 (BPA) avec CloudFront.

Bonnes pratiques relatives aux applications Amazon EKS

Lorsque vous l'utilisez Autorité de certification privée AWS pour approvisionner Amazon EKS avec des certificats X.509, suivez les recommandations relatives à la sécurisation des environnements à locataires multiples figurant dans les guides des meilleures pratiques Amazon EKS. Pour des informations générales sur l'intégration Autorité de certification privée AWS à Kubernetes, consultez. Sécuriser Kubernetes avec Autorité de certification privée AWS

Administration privée de CA

Vous pouvez ainsi créer une hiérarchie entièrement AWS hébergée d'autorités de certification (CA) racines et subordonnées à usage interne par votre organisation. Autorité de certification privée AWS Pour gérer la révocation des certificats, vous pouvez activer le protocole OCSP (Online Certificate Status Protocol), les listes de révocation de certificats (CRL) ou les deux. Autorité de certification privée AWS stocke et gère vos certificats CA, vos CRL et vos réponses OCSP, et les clés privées de vos autorités racines sont stockées de manière sécurisée par. AWS

> **Note**
>
> L'implémentation OCSP dans Autorité de certification privée AWS ne prend pas en charge les extensions de requête OCSP. Si vous soumettez une requête par lots OCSP contenant plusieurs certificats, le répondeur AWS OCSP traite uniquement le premier certificat de la file d'attente et supprime les autres. Une révocation peut prendre jusqu'à une heure pour apparaître dans les réponses de l'OCSP.

Vous pouvez y accéder Autorité de certification privée AWS en utilisant le AWS Management Console, le AWS CLI, et l' Autorité de certification privée AWS API. Les rubriques suivantes vous montrent comment utiliser la console et l'interface de ligne de commande. Pour en savoir plus sur l'API, consultez la référence des AWS Private Certificate Authority API. Pour obtenir des exemples Java montrant comment utiliser l'API, consultez Utilisation de l'Autorité de certification privée AWSAPI (exemples Java).

Rubriques

Création d'une autorité de certification privée

Vous pouvez utiliser les procédures décrites dans cette section pour créer des autorités de certification racines ou des autorités de certification subordonnées, afin d'obtenir une hiérarchie vérifiable des relations de confiance correspondant aux besoins de votre organisation. Vous pouvez créer une autorité de certification à AWS Management Console l'aide de la partie PCA du AWS CLI ou AWS CloudFormation.

Pour plus d'informations sur la mise à jour de la configuration d'une autorité de certification que vous avez déjà créée, consultezMettre à jour votre CA privée.

Pour plus d'informations sur l'utilisation d'une autorité de certification pour signer des certificats d'entité finale pour vos utilisateurs, appareils et applications, consultezÉmission de certificats d'entité finale privés.

> **Note**
>
> Un tarif mensuel est facturé à votre compte pour chaque autorité de certification privée à partir du moment où vous l'avez créée.
> Pour obtenir les dernières informations Autorité de certification privée AWS tarifaires, consultez la section AWS Private Certificate Authority Tarification. Vous pouvez également utiliser le calculateur de AWS prix pour estimer les coûts.

Rubriques

- Procédure de création d'une autorité de certification (console)
- Procédure de création d'une autorité de certification (CLI)
- Utilisation AWS CloudFormation pour créer une autorité de certification

Procédure de création d'une autorité de certification (console)

Procédez comme suit pour créer une autorité de certification privée à l'aide du AWS Management Console.

Pour commencer à utiliser la console

Connectez-vous à votre AWS compte et ouvrez la Autorité de certification privée AWS console à l'adresse**`https://console.aws.amazon.com/acm-pca/home`**.

- Si vous ouvrez la console dans une région où vous n'avez aucune autorité de certification privée, la page d'introduction apparaît. Choisissez Create a private CA.
- Si vous ouvrez la console dans une région où vous avez déjà créé une autorité de certification, la page Autorités de certification privées s'ouvre avec la liste de vos autorités de certification. Choisissez Create CA.

Options de mode

Dans la section Options du mode de la console, choisissez le mode d'expiration des certificats émis par votre autorité de certification.

- Usage général : émet des certificats qui peuvent être configurés avec n'importe quelle date d'expiration. Il s'agit de l'option par défaut.
- Certificat de courte durée — Délivre des certificats dont la durée de validité maximale est de sept jours. Une courte période de validité peut remplacer dans certains cas un mécanisme de révocation.

Options de type de CA

Dans la section Options de type de la console, choisissez le type d'autorité de certification privée que vous souhaitez créer.

- Le choix de Root établit une nouvelle hiérarchie CA. Cette autorité de certification est basée sur un certificat auto-signé. Il constitue l'autorité de signature ultime pour les autres autorités de certification et certificats d'entité finale de la hiérarchie.
- Le choix de Subordinate crée une autorité de certification qui doit être signée par une autorité de certification parente située au-dessus de cette autorité dans la hiérarchie. Les autorités de certification subordonnées sont généralement utilisées pour créer d'autres autorités de certification subordonnées ou pour délivrer des certificats d'entité finale aux utilisateurs, aux ordinateurs et aux applications.

> **Note**
>
> Autorité de certification privée AWS fournit un processus de signature automatique lorsque l'autorité de certification parent de votre autorité de certification subordonnée est également hébergée par Autorité de certification privée AWS. Il vous suffit de choisir l'autorité de certification parent à utiliser.

Votre autorité de certification subordonnée devra peut-être être signée par un fournisseur de services de confiance externe. Si tel est le cas, vous Autorité de certification privée AWS fournit une demande de signature de certificat (CSR) que vous devez télécharger et utiliser pour obtenir un certificat CA signé. Pour plus d'informations, consultez Installation d'un certificat d'autorité de certification subordonnée signé par une autorité de certification parent externe.

Options de nom distinctif du sujet

Sous Options de nom distinctif du sujet, configurez le nom du sujet de votre autorité de certification privée. Vous devez saisir une valeur pour au moins l'une des options suivantes :

- Organisation (O) — Par exemple, le nom d'une entreprise
- Unité organisationnelle (UO) — Par exemple, une division au sein d'une entreprise
- Nom du pays (C) — Code de pays à deux lettres
- Nom de l'État ou de la province — Nom complet d'un État ou d'une province
- Nom de la localité — Le nom d'une ville
- Nom commun (CN) : chaîne lisible par l'homme pour identifier l'autorité de certification.

Note

Vous pouvez personnaliser davantage le nom du sujet d'un certificat en appliquant un modèle APIPassThrough au moment de l'émission. Pour plus d'informations et un exemple détaillé, consultezÉmettre un certificat avec un nom de sujet personnalisé à l'aide d'un modèle APIPassThrough.

Le certificat de sauvegarde étant auto-signé, les informations relatives au sujet que vous fournissez à une autorité de certification privée sont probablement plus limitées que celles que contiendrait une autorité de certification publique. Pour plus d'informations sur chacune des valeurs qui constituent le nom distinctif d'un sujet, consultez la RFC 5280.

Principales options de l'algorithme

Sous Options de l'algorithme clé, choisissez l'algorithme clé et la taille en bits de la clé. La valeur par défaut est un algorithme RSA avec une longueur de clé de 2 048 bits. Vous pouvez choisir parmi les algorithmes suivants :

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

Options de révocation des certificats

Dans la section Options de révocation des certificats, vous pouvez choisir entre deux méthodes pour partager le statut de révocation avec les clients qui utilisent vos certificats :

- Activer la distribution CRL
- Activez OCSP

Vous pouvez configurer l'une ou l'autre de ces options de révocation, ou les deux, pour votre autorité de certification. Bien que facultative, la gestion des révocations est recommandée en tant que bonne pratique. Avant de terminer cette étape, consultez Configuration d'une méthode de révocation des certificats les informations relatives aux avantages de chaque méthode, à la configuration préliminaire qui pourrait être requise et aux fonctionnalités de révocation supplémentaires.

> **Note**
>
> Si vous créez votre autorité de certification sans configurer la révocation, vous pourrez toujours la configurer ultérieurement. Pour plus d'informations, consultez Mettre à jour votre CA privée.

Pour configurer une CRL

1. Sous Options de révocation des certificats, choisissez Activer la distribution CRL.
2. Pour créer un compartiment Amazon S3 pour vos entrées CRL, choisissez Create a new S3 bucket et saisissez un nom de bucket unique. (Vous n'avez pas besoin d'inclure le chemin

d'accès au compartiment.) Sinon, sous URI du compartiment S3, choisissez un compartiment existant dans la liste.

Lorsque vous créez un nouveau compartiment par le biais de la console Autorité de certification privée AWS , vous essayez d'associer la politique d'accès requise au compartiment et de désactiver le paramètre BPA (Block Public Access) par défaut de S3 sur celui-ci. Si vous spécifiez plutôt un bucket existant, vous devez vous assurer que le BPA est désactivé pour le compte et pour le bucket. Dans le cas contraire, l'opération de création de l'autorité de certification échouera. Si l'autorité de certification est créée avec succès, vous devez toujours y associer manuellement une politique avant de pouvoir commencer à générer des CRL. Utilisez l'un des modèles de politique décrits dansPolitiques d'accès pour les CRL dans Amazon S3 . Pour plus d'informations, consultez Ajouter une politique de compartiment à l'aide de la console Amazon S3.

> **Important**
>
> Une tentative de création d'une autorité de certification à l'aide de la Autorité de certification privée AWS console échoue si toutes les conditions suivantes sont réunies :
>
> - Vous êtes en train de configurer une CRL.
> - Vous demandez Autorité de certification privée AWS à créer automatiquement un compartiment S3.
> - Vous appliquez les paramètres BPA dans S3.
>
> Dans ce cas, la console crée un compartiment, mais tente de le rendre accessible au public sans succès. Vérifiez vos paramètres Amazon S3 si cela se produit, désactivez le BPA si nécessaire, puis répétez la procédure de création d'une autorité de certification. Pour plus d'informations, consultez Blocage de l'accès public à votre espace de stockage Amazon S3.

3. Développez les paramètres CRL pour obtenir des options de configuration supplémentaires.

   - Ajoutez un nom CRL personnalisé pour créer un alias pour votre compartiment Amazon S3. Ce nom est contenu dans les certificats émis par l'autorité de certification dans l'extension « CRL Distribution Points » définie par la RFC 5280.
   - Entrez la durée de validité en jours pendant laquelle votre CRL restera valide. La valeur par défaut est 7 jours. Pour les CRL en ligne, une période de validité de 2 à 7 jours est courante.

Autorité de certification privée AWS essaie de régénérer la CRL au milieu de la période spécifiée.

4. Étendez les paramètres S3 pour la configuration facultative du versionnement des compartiments et de la journalisation des accès aux compartiments.

Pour configurer OCSP

1. Sous Options de révocation des certificats, choisissez Activer l'OCSP.
2. Dans le champ Point de terminaison OCSP personnalisé - facultatif, vous pouvez fournir un nom de domaine complet (FQDN) pour un point de terminaison autre qu'Amazon OCSP.

   Lorsque vous fournissez un FQDN dans ce champ, Autorité de certification privée AWS insère le FQDN dans l'extension Authority Information Access de chaque certificat émis à la place de l'URL par défaut du répondeur AWS OCSP. Lorsqu'un point de terminaison reçoit un certificat contenant le nom de domaine complet personnalisé, il demande à cette adresse une réponse OCSP. Pour que ce mécanisme fonctionne, vous devez effectuer deux actions supplémentaires :

   - Utilisez un serveur proxy pour transférer le trafic qui arrive à votre nom de domaine complet personnalisé vers le répondeur AWS OCSP.
   - Ajoutez un enregistrement CNAME correspondant à votre base de données DNS.

   > **Tip**
   >
   > Pour plus d'informations sur la mise en œuvre d'une solution OCSP complète à l'aide d'un CNAME personnalisé, consultez. Configuration d'une URL personnalisée pour Autorité de certification privée AWS OCSP

   Par exemple, voici un enregistrement CNAME pour un OCSP personnalisé tel qu'il apparaîtrait dans Amazon Route 53.

Nom de l'enregistrement	Type	Stratégie de routage	Différenciateur	Valeur/acheminer le trafic vers
alternative.exemple.com	CNAME	Simplicité	-	proxy.exemple.com

> **Note**
>
> La valeur du CNAME ne doit pas inclure de préfixe de protocole tel que « http ://» ou « https ://».

Ajout de balises

Sous Ajouter des balises, vous pouvez éventuellement étiqueter votre autorité de certification. Les balises sont des paires clé-valeur qui servent de métadonnées pour identifier et organiser des ressources AWS . Pour obtenir la liste des paramètres des Autorité de certification privée AWS balises et des instructions sur la façon d'ajouter des balises aux autorités de certification après leur création, consultezGestion des tags pour votre autorité de certification privée.

> **Note**
>
> Pour associer des balises à une autorité de certification privée au cours de la procédure de création, un administrateur de l'autorité de certification doit d'abord associer une politique IAM intégrée à l'`CreateCertificateAuthority`action et autoriser explicitement le balisage. Pour plus d'informations, consultez T ag-on-create : Attacher des tags à une autorité de certification au moment de sa création.

Options d'autorisations CA

Dans les options d'autorisation de l'autorité de certification, vous pouvez éventuellement déléguer les autorisations de renouvellement automatique au principal du AWS Certificate Manager service. ACM ne peut renouveler automatiquement les certificats d'entité finale privés générés par cette autorité de certification que si cette autorisation est accordée. Vous pouvez attribuer des autorisations de

renouvellement à tout moment à l'aide de l' Autorité de certification privée AWS CreatePermissionAPI ou de la commande create-permission CLI.

La valeur par défaut est d'activer ces autorisations.

> **Note**
>
> AWS Certificate Manager ne prend pas en charge le renouvellement automatique des certificats de courte durée.

Tarification

Sous Tarification, confirmez que vous comprenez les tarifs applicables à une autorité de certification privée.

> **Note**
>
> Pour obtenir les dernières informations Autorité de certification privée AWS tarifaires, consultez la section AWS Private Certificate Authority Tarification. Vous pouvez également utiliser le calculateur de AWS prix pour estimer les coûts.

Création d'une CA

Choisissez Create CA après avoir vérifié l'exactitude de toutes les informations saisies. La page de détails de l'autorité de certification s'ouvre et affiche son statut en tant que certificat en attente.

> **Note**
>
> Sur la page de détails, vous pouvez terminer la configuration de votre autorité de certification en choisissant Actions, Installer le certificat de l'autorité de certification, ou vous pouvez revenir ultérieurement à la liste des autorités de certification privées et terminer la procédure d'installation qui s'applique à votre cas :
>
> - Installation d'un certificat CA racine
> - Installation d'un certificat CA subordonné hébergé par Autorité de certification privée AWS

- Installation d'un certificat d'autorité de certification subordonnée signé par une autorité de certification parent externe

Procédure de création d'une autorité de certification (CLI)

Utilisez la commande create-certificate-authority pour créer une autorité de certification privée. Vous devez spécifier la configuration de l'autorité de certification (contenant les informations relatives à l'algorithme et au nom du sujet), la configuration de révocation (si vous prévoyez d'utiliser un OCSP et/ou une CRL) et le type d'autorité de certification (racine ou subordonnée). Les détails de configuration et de révocation sont contenus dans deux fichiers que vous fournissez en tant qu'arguments de la commande. Facultativement, vous pouvez également configurer le mode d'utilisation de l'autorité de certification (pour l'émission de certificats standard ou de courte durée), joindre des balises et fournir un jeton d'idempuissance.

Si vous configurez une CRL, vous devez disposer d'un compartiment Amazon S3 sécurisé avant d'émettre la create-certificate-authority commande. Pour plus d'informations, consultez Politiques d'accès pour les CRL dans Amazon S3 .

Le fichier de configuration de l'autorité de certification indique les informations suivantes :

- Le nom de l'algorithme
- La taille de la clé à utiliser pour créer la clé privée de l'autorité de certification
- Le type de l'algorithme de signature que l'autorité de certification utilise pour signer
- Les informations sur l'objet X.500

La configuration de révocation pour OCSP définit un `OcspConfiguration` objet avec les informations suivantes :

- Le `Enabled` drapeau est réglé sur « vrai ».
- (Facultatif) Un CNAME personnalisé déclaré en tant que valeur pour`OcspCustomCname`.

La configuration de révocation d'une CRL définit un `CrlConfiguration` objet avec les informations suivantes :

- Le `Enabled` drapeau est réglé sur « vrai ».
- Période d'expiration de la CRL en jours (période de validité de la CRL).

- Le compartiment Amazon S3 qui contiendra la CRL.
- (Facultatif) Une ObjectAcl valeur S3 qui détermine si la CRL est accessible au public. Dans l'exemple présenté ici, l'accès public est bloqué. Pour plus d'informations, consultez Activation de l'accès public par blocs S3 (BPA) avec CloudFront.
- (Facultatif) Un alias CNAME pour le compartiment S3 inclus dans les certificats émis par l'autorité de certification. Si la CRL n'est pas accessible au public, cela indiquera un mécanisme de distribution tel qu'Amazon CloudFront.
- (Facultatif) Un `CrlDistributionPointExtensionConfiguration` objet contenant les informations suivantes :
  - Le `OmitExtension` drapeau est défini sur « vrai » ou « faux ». Cela permet de contrôler si la valeur par défaut de l'extension CDP sera écrite sur un certificat émis par l'autorité de certification. Pour plus d'informations sur l'extension CDP, consultezDéterminer l'URI du point de distribution CRL (CDP) . A CustomCname ne peut pas être défini s' OmitExtension il est « vrai ».

> **Note**
>
> Vous pouvez activer les deux mécanismes de révocation sur la même autorité de certification en définissant à la fois un `OcspConfiguration` objet et un `CrlConfiguration` objet. Si vous ne fournissez aucun --revocation-configuration paramètre, les deux mécanismes sont désactivés par défaut. Si vous avez besoin d'une assistance pour la validation de la révocation ultérieurement, consultezMettre à jour une autorité de certification (CLI).

Les exemples suivants supposent que vous avez configuré votre répertoire de `.aws` configuration avec une région par défaut, un point de terminaison et des informations d'identification valides. Pour plus d'informations sur la configuration de votre AWS CLI environnement, consultez Configuration et paramètres des fichiers d'identification. Pour des raisons de lisibilité, nous fournissons les entrées de configuration et de révocation de l'autorité de certification sous forme de fichiers JSON dans les exemples de commandes. Modifiez les fichiers d'exemple en fonction de vos besoins.

Tous les exemples utilisent le fichier de `ca_config.txt` configuration suivant, sauf indication contraire.

Fichier : ca_config.txt

```
{
```

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

Exemple 1 : créer une autorité de certification avec OCSP activé

Dans cet exemple, le fichier de révocation active le support OCSP par défaut, qui utilise le Autorité de certification privée AWS répondeur pour vérifier l'état du certificat.

Fichier : revoke_config.txt pour OCSP

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

Commande

```
$ 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
```

En cas de succès, cette commande affiche l'Amazon Resource Name (ARN) de la nouvelle autorité de certification.

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

Commande

```
$ 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
```

En cas de succès, cette commande affiche l'Amazon Resource Name (ARN) de l'autorité de certification.

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

Utilisez la commande suivante pour inspecter la configuration de votre autorité de certification.

```
$ 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
```

Cette description doit contenir la section suivante.

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

Exemple 2 : créer une autorité de certification avec OCSP et un CNAME personnalisé activés

Dans cet exemple, le fichier de révocation permet un support OCSP personnalisé. Le `OcspCustomCname` paramètre prend un nom de domaine complet (FQDN) comme valeur.

Lorsque vous fournissez un FQDN dans ce champ, Autorité de certification privée AWS insère le FQDN dans l'extension Authority Information Access de chaque certificat émis à la place de l'URL par défaut du répondeur AWS OCSP. Lorsqu'un point de terminaison reçoit un certificat contenant le nom de domaine complet personnalisé, il demande à cette adresse une réponse OCSP. Pour que ce mécanisme fonctionne, vous devez effectuer deux actions supplémentaires :

- Utilisez un serveur proxy pour transférer le trafic qui arrive à votre nom de domaine complet personnalisé vers le répondeur AWS OCSP.
- Ajoutez un enregistrement CNAME correspondant à votre base de données DNS.

> **Tip**
>
> Pour plus d'informations sur la mise en œuvre d'une solution OCSP complète à l'aide d'un CNAME personnalisé, consultez. Configuration d'une URL personnalisée pour Autorité de certification privée AWS OCSP

Par exemple, voici un enregistrement CNAME pour un OCSP personnalisé tel qu'il apparaîtrait dans Amazon Route 53.

Nom de l'enregistrement	Type	Stratégie de routage	Différenciateur	Valeur/acheminer le trafic vers
alternative.exemple.com	CNAME	Simplicité	-	proxy.exemple.com

> **Note**
>
> La valeur du CNAME ne doit pas inclure de préfixe de protocole tel que « http ://» ou « https ://».

Fichier : revoke_config.txt pour OCSP

```
{
```

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

Commande

```
$ 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
```

En cas de succès, cette commande affiche l'Amazon Resource Name (ARN) de l'autorité de certification.

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

Utilisez la commande suivante pour inspecter la configuration de votre autorité de certification.

```
$ 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
```

Cette description doit contenir la section suivante.

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

Exemple 3 : créer une autorité de certification avec une CRL attachée

Dans cet exemple, la configuration de révocation définit les paramètres CRL.

Fichier : revoke_config.txt

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

Commande

```
$ 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
```

En cas de succès, cette commande affiche l'Amazon Resource Name (ARN) de l'autorité de certification.

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

Utilisez la commande suivante pour inspecter la configuration de votre autorité de certification.

```
$ 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
```

Cette description doit contenir la section suivante.

```
"RevocationConfiguration": {
```

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

Exemple 4 : créer une autorité de certification avec une CRL attachée et un CNAME personnalisé activé

Dans cet exemple, la configuration de révocation définit les paramètres CRL qui incluent un CNAME personnalisé.

Fichier : revoke_config.txt

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

Commande

```
$ 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
```

En cas de succès, cette commande affiche l'Amazon Resource Name (ARN) de l'autorité de certification.

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

```
}
```

Utilisez la commande suivante pour inspecter la configuration de votre autorité de certification.

```
$ 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
```

Cette description doit contenir la section suivante.

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

Exemple 5 : créer une autorité de certification et spécifier le mode d'utilisation

Dans cet exemple, le mode d'utilisation de l'autorité de certification est spécifié lors de la création d'une autorité de certification. S'il n'est pas spécifié, le paramètre du mode d'utilisation prend par défaut la valeur GENERAL_PURPOSE. Dans cet exemple, le paramètre est défini sur SHORT_LIVED_CERTIFICATE, ce qui signifie que l'autorité de certification délivrera des certificats dont la durée de validité maximale est de sept jours. Dans les situations où il n'est pas pratique de configurer la révocation, un certificat de courte durée qui a été compromis expire rapidement dans le cadre des opérations normales. Par conséquent, cet exemple d'autorité de certification ne dispose pas d'un mécanisme de révocation.

> **Note**
>
> Autorité de certification privée AWS n'effectue pas de contrôles de validité sur les certificats de l'autorité de certification racine.

```
$ 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
```

Utilisez la describe-certificate-authoritycommande dans le AWS CLI pour afficher des détails sur l'autorité de certification obtenue, comme indiqué dans la commande suivante :

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

Exemple 6 : créer une autorité de certification pour la connexion à Active Directory

Vous pouvez créer une autorité de certification privée adaptée à une utilisation dans le magasin Enterprise NTAuth de Microsoft Active Directory (AD), où elle peut émettre des certificats d'ouverture de session par carte ou de contrôleur de domaine. Pour plus d'informations sur l'importation d'un certificat CA dans AD, consultez Comment importer des certificats d'autorité de certification (CA) tiers dans le magasin Enterprise NTAuth.

L'outil Microsoft certutil peut être utilisé pour publier des certificats CA dans AD en invoquant l'option. -dspublish Un certificat publié sur AD avec certutil est fiable dans l'ensemble de la forêt. À l'aide de la stratégie de groupe, vous pouvez également limiter la confiance à un sous-ensemble de la forêt entière, par exemple un seul domaine ou un groupe d'ordinateurs dans un domaine. Pour que la connexion fonctionne, l'autorité de certification émettrice doit également être publiée dans le magasin NTAuth. Pour plus d'informations, voir Distribuer des certificats aux ordinateurs clients à l'aide d'une stratégie de groupe.

Cet exemple utilise le fichier `ca_config_AD.txt` de configuration suivant.

Fichier : 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"
         }
      ]
   }
}
```

Commande

```
$ aws acm-pca create-certificate-authority \
      --certificate-authority-configuration file://ca_config_AD.txt \
      --certificate-authority-type "ROOT" \
      --tags Key=application,Value=ActiveDirectory
```

En cas de succès, cette commande affiche l'Amazon Resource Name (ARN) de l'autorité de certification.

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

Utilisez la commande suivante pour inspecter la configuration de votre autorité de certification.

```
$ 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
```

Cette description doit contenir la section suivante.

```
...

"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"
      }
   ]
}
...
```

Exemple 7 : créer une autorité de certification Matter avec une CRL attachée et l'extension CDP omise dans les certificats émis

Vous pouvez créer une autorité de certification privée adaptée à la délivrance de certificats pour la norme de maison intelligente Matter. Dans cet exemple, la configuration CA `ca_config_PAA.txt` définit une autorité d'attestation de produit (PAA) Matter Product Attestation Authority (PAA) dont l'ID de fournisseur (VID) est défini sur FFF1.

Fichier : ca_config_PAA.txt

```
{
   "KeyAlgorithm":"EC_prime256v1",
   "SigningAlgorithm":"SHA256WITHECDSA",
   "Subject":{
      "Country":"US",
      "Organization":"Example Corp",
      "OrganizationalUnit":"SmartHome",
      "State":"WA",
      "Locality":"Seattle",
      "CommonName":"Example Corp Matter PAA",
   "CustomAttributes":[
      {
        "ObjectIdentifier":"1.3.6.1.4.1.37244.2.1",
        "Value":"FFF1"
      }
    ]
  }
}
```

La configuration de révocation active les CRL et configure l'autorité de certification pour omettre l'URL CDP par défaut de tous les certificats émis.

Fichier : revoke_config.txt

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

```
   }
}
```

Commande

```
$ 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
```

En cas de succès, cette commande affiche l'Amazon Resource Name (ARN) de l'autorité de certification.

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

Utilisez la commande suivante pour inspecter la configuration de votre autorité de certification.

```
$ 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
```

Cette description doit contenir la section suivante.

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

```
...
```

Utilisation AWS CloudFormation pour créer une autorité de certification

Pour plus d'informations sur la création d'une autorité de certification privée à l'aide de AWS CloudFormation, reportez-vous à la section Référence Autorité de certification privée AWS des types de ressources dans le guide de AWS CloudFormation l'utilisateur.

Création et installation du certificat CA

Suivez les procédures suivantes pour créer et installer votre certificat d'une autorité de certification privée. Votre autorité de certification sera alors prête à l'emploi.

Autorité de certification privée AWS prend en charge trois scénarios d'installation d'un certificat CA :

- Installation d'un certificat pour une autorité de certification racine hébergée par Autorité de certification privée AWS
- Installation d'un certificat d'une autorité de certification subordonnée dont l'autorité parente est hébergée par Autorité de certification privée AWS
- Installation d'un certificat d'une autorité de certification subordonnée dont l'autorité parent est hébergée en externe

Les sections suivantes décrivent les procédures pour chaque scénario. Les procédures de console commencent sur la page de la console Autorités de certification privées.

Algorithmes de signature compatibles

La prise en charge des algorithmes de signature pour les certificats CA dépend de l'algorithme de signature de l'autorité de certification parent et du Région AWS. Les contraintes suivantes s'appliquent à la fois à la console et AWS CLI aux opérations.

- Une autorité de certification parent utilisant l'algorithme de signature RSA peut émettre des certificats avec les algorithmes suivants :
  - SHA256 RSA
  - SHA384 RSA
  - SHA512 RSA

- Dans une ancienne version Région AWS, une autorité de certification parent utilisant l'algorithme de signature EDCSA peut émettre des certificats avec les algorithmes suivants :
  - SHA256 ECDSA
  - SHA384 ECDSA
  - SHA512 ECDSA

  L'héritage Régions AWS inclut :

Nom de la région	Situation géographique
eu-north-1	Europe (Stockholm)
me-south-1	Moyen-Orient (Bahreïn)
ap-south-1	Asie-Pacifique (Mumbai)
eu-west-3	Europe (Paris)
us-east-2	USA Est (Ohio)
af-south-1	Afrique (Le Cap)
eu-west-1	Europe (Irlande)
eu-central-1	Europe (Francfort)
sa-east-1	Amérique du Sud (São Paulo)
ap-east-1	Asie-Pacifique (Hong Kong)
us-east-1	USA Est (Virginie du Nord)

Nom de la région	Situation géographique
ap-northeast-2	Asie-Pacifique (Séoul)
eu-west-2	Europe (Londres)
ap-northeast-1	Asie-Pacifique (Tokyo)
us-gov-east-1	AWS GovCloud (USA Est)
us-gov-west-1	AWS GovCloud (US-Ouest)
us-west-2	USA Ouest (Oregon)
us-west-1	USA Ouest (Californie du Nord)
ap-southeast-1	Asie-Pacifique (Singapour)
ap-southeast-2	Asie-Pacifique (Sydney)

- Dans le cas d'un produit qui n'est pas un héritage Région AWS, les règles suivantes s'appliquent à l'EDCSA :
  - Une autorité de certification parent utilisant l'algorithme de signature EC_Prime256v1 peut émettre des certificats avec ECDSA P256.
  - Une autorité de certification parent utilisant l'algorithme de signature EC_SECP384r1 peut émettre des certificats avec ECDSA P384.

Installation d'un certificat CA racine

Vous pouvez installer un certificat CA racine à partir du AWS Management Console ou du AWS CLI.

Pour créer et installer un certificat pour votre autorité de certification racine privée (console)

1. (Facultatif) Si vous n'êtes pas encore sur la page de détails de l'autorité de certification, ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home. Sur la page Autorités de certification privées, choisissez une autorité de certification racine dont le statut est En attente de certificat ou Actif.
2. Choisissez Actions, Installer le certificat CA pour ouvrir la page Installer le certificat CA racine.
3. Sous Spécifier les paramètres du certificat de l'autorité de certification racine, spécifiez les paramètres de certificat suivants :

   - Validité — Spécifie la date et l'heure d'expiration du certificat CA. La période de validité Autorité de certification privée AWS par défaut d'un certificat CA racine est de 10 ans.
   - Algorithme de signature — Spécifie l'algorithme de signature à utiliser lorsque l'autorité de certification racine émet de nouveaux certificats. Les options disponibles varient en fonction de l' Région AWS endroit où vous créez l'autorité de certification. Pour plus d'informationsAlgorithmes de signature compatibles, consultezAlgorithmes cryptographiques pris en charge, et SigningAlgorithmdans CertificateAuthorityConfiguration.
     - SHA256 RSA
     - SHA384 RSA
     - SHA512 RSA

   Vérifiez que vos paramètres sont corrects, puis choisissez Confirmer et installer. Autorité de certification privée AWS exporte un CSR pour votre autorité de certification, génère un certificat à l'aide d'un modèle de certificat de l'autorité de certification racine et signe automatiquement le certificat. Autorité de certification privée AWS importe ensuite le certificat de l'autorité de certification racine auto-signé.
4. La page de détails de l'autorité de certification affiche l'état de l'installation (réussite ou échec) en haut de la page. Si l'installation a réussi, l'autorité de certification racine nouvellement terminée affiche le statut Actif dans le volet Général.

Pour créer et installer un certificat pour votre autorité de certification racine privée (AWS CLI)

1. Générez une demande de signature de certificat (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
```

Le fichier obtenu `ca.csr`, un fichier PEM codé au format base64, présente l'aspect suivant.

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

Vous pouvez utiliser OpenSSL pour afficher et vérifier le contenu du CSR.

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

Cela donne un résultat similaire à ce qui suit.

```
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. En utilisant le CSR de l'étape précédente comme argument pour le `--csr` paramètre, émettez le certificat racine.

> **Note**
>
> Si vous utilisez la AWS CLI version 1.6.3 ou une version ultérieure, utilisez le préfixe `fileb://` lorsque vous spécifiez le fichier d'entrée requis. Cela garantit que les données codées Autorité de certification privée AWS en Base64 sont correctement analysées.

```
$ 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. Récupérez le certificat racine.

```
$ 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
```

Le fichier obtenu`cert.pem`, un fichier PEM codé au format base64, présente l'aspect suivant.

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

Vous pouvez utiliser OpenSSL pour consulter et vérifier le contenu du certificat.

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

Cela donne un résultat similaire à ce qui suit.

```
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. Importez le certificat de l'autorité de certification racine pour l'installer sur l'autorité de certification.

> **Note**
>
> Si vous utilisez la AWS CLI version 1.6.3 ou une version ultérieure, utilisez le préfixe `fileb://` lorsque vous spécifiez le fichier d'entrée requis. Cela garantit que les

données codées Autorité de certification privée AWS en Base64 sont correctement analysées.

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

Vérifiez le nouveau statut de l'autorité de certification.

```
$ 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
```

Le statut apparaît désormais comme ACTIF.

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

Installation d'un certificat CA subordonné hébergé par Autorité de certification privée AWS

Vous pouvez utiliser le AWS Management Console pour créer et installer un certificat pour votre autorité de certification subordonnée Autorité de certification privée AWS hébergée.

Pour créer et installer un certificat pour votre autorité de certification subordonnée Autorité de certification privée AWS hébergée

1. (Facultatif) Si vous n'êtes pas encore sur la page de détails de l'autorité de certification, ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home. Sur la page Autorités de certification privées, choisissez une autorité de certification subordonnée dont le statut est En attente de certificat ou Active.
2. Choisissez Actions, Installer le certificat CA pour ouvrir la page Installer le certificat CA subordonné.
3. Sur la page Installer un certificat d'autorité de certification subordonnée, sous Sélectionner le type d'autorité AWS Private CAde certification, choisissez d'installer un certificat géré par Autorité de certification privée AWS.
4. Sous Sélectionner une autorité de certification parent, choisissez une autorité de certification dans la liste des autorités de certification privées parentes. La liste est filtrée pour afficher les autorités de certification répondant aux critères suivants :
   - Vous êtes autorisé à utiliser le CA.
   - Le CA ne signerait pas lui-même.

- Le CA est en étatACTIVE.
- Le mode CA estGENERAL_PURPOSE.

5. Sous Spécifier les paramètres de certificat de l'autorité de certification subordonnée, spécifiez les paramètres de certificat suivants :

   - Validité — Spécifie la date et l'heure d'expiration du certificat CA.
   - Algorithme de signature — Spécifie l'algorithme de signature à utiliser lorsque l'autorité de certification racine émet de nouveaux certificats. Les options sont :
     - SHA256 RSA
     - SHA384 RSA
     - SHA512 RSA
   - Longueur du chemin : nombre de couches de confiance que l'autorité de certification subordonnée peut ajouter lors de la signature de nouveaux certificats. Une longueur de chemin de zéro (valeur par défaut) signifie que seuls les certificats d'entité finale, et non les certificats CA, peuvent être créés. Une longueur de chemin supérieure ou égale à un signifie que l'autorité de certification subordonnée peut émettre des certificats pour créer des autorités de certification supplémentaires qui lui sont subordonnées.
   - ARN du modèle : affiche l'ARN du modèle de configuration pour ce certificat CA. Le modèle change si vous modifiez la Longueur de chemin d'accès spécifiée. Si vous créez un certificat à l'aide de la commande CLI issue-certificate ou de l'IssueCertificateaction API, vous devez spécifier l'ARN manuellement. Pour de plus amples informations sur les modèles de certificats d'une autorité de certification disponibles, veuillez consulter Comprendre les modèles de certificats.

6. Vérifiez que vos paramètres sont corrects, puis choisissez Confirmer et installer. Autorité de certification privée AWS exporte un CSR, génère un certificat à l'aide d'un modèle de certificat d'autorité de certification subordonnée et signe le certificat auprès de l'autorité de certification parent sélectionnée. Autorité de certification privée AWS importe ensuite le certificat CA subordonné signé.
7. La page de détails de l'autorité de certification affiche l'état de l'installation (réussite ou échec) en haut de la page. Si l'installation a réussi, l'autorité de certification subordonnée nouvellement terminée affiche le statut Actif dans le volet Général.

Installation d'un certificat d'autorité de certification subordonnée signé par une autorité de certification parent externe

Après avoir créé une autorité de certification privée subordonnée comme décrit dans Procédure de création d'une autorité de certification (console) ouProcédure de création d'une autorité de certification (CLI) , vous avez la possibilité de l'activer en installant un certificat d'autorité de certification signé par une autorité de signature externe. Pour signer le certificat de votre autorité de certification subordonnée auprès d'une autorité de certification externe, vous devez d'abord configurer un fournisseur de services de confiance externe comme autorité de signature, ou faire appel à un fournisseur tiers.

> **Note**
>
> Les procédures de création ou d'obtention d'un fournisseur de services de confiance externe n'entrent pas dans le cadre de ce guide.

Après avoir créé une autorité de certification subordonnée et avoir accès à une autorité de signature externe, effectuez les tâches suivantes :

1. Obtenez une demande de signature de certificat (CSR) auprès de Autorité de certification privée AWS.
2. Soumettez le CSR à votre autorité de signature externe et obtenez un certificat CA signé ainsi que tous les certificats de chaîne.
3. Importez le certificat de l'autorité de certification et sa chaîne Autorité de certification privée AWS pour activer votre autorité de certification subordonnée.

Pour connaître les procédures détaillées, consultez Certificats CA privés signés en externe .

Contrôle de l'accès à une autorité de certification privée

Tout utilisateur disposant des autorisations nécessaires sur une autorité de certification privée Autorité de certification privée AWS peut utiliser cette autorité de certification pour signer d'autres certificats. Le propriétaire de l'autorité de certification peut délivrer des certificats ou déléguer les autorisations requises pour l'émission de certificats à un utilisateur AWS Identity and Access Management (IAM) résidant dans le même Compte AWSétablissement. Un utilisateur résidant sur

un autre AWS compte peut également émettre des certificats s'il est autorisé par le propriétaire de l'autorité de certification par le biais d'une politique basée sur les ressources.

Les utilisateurs autorisés, qu'ils soient titulaires d'un compte unique ou multicompte, peuvent utiliser Autorité de certification privée AWS nos AWS Certificate Manager ressources lorsqu'ils émettent des certificats. Les certificats émis à partir de l' Autorité de certification privée AWS IssueCertificateAPI ou de la commande issue-certificate CLI ne sont pas gérés. Ces certificats nécessitent une installation manuelle sur les appareils cibles et un renouvellement manuel à leur expiration. Les certificats émis à partir de la console ACM, de l'RequestCertificateAPI ACM ou de la commande CLI request-certificate sont gérés. Ces certificats peuvent facilement être installés dans les services intégrés à ACM. Si l'administrateur de l'autorité de certification l'autorise et que le compte de l'émetteur dispose d'un rôle lié au service pour ACM, les certificats gérés sont renouvelés automatiquement à leur expiration.

Rubriques

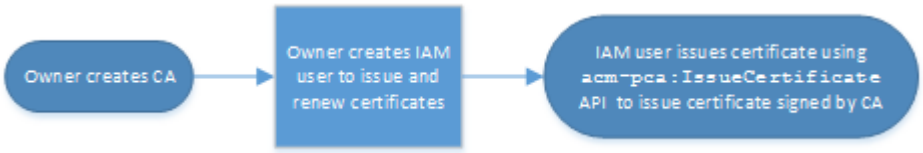
- Création d'autorisations de compte unique pour un utilisateur IAM
- Joindre une politique d'accès entre comptes

Création d'autorisations de compte unique pour un utilisateur IAM

Lorsque l'administrateur de l'autorité de certification (c'est-à-dire le propriétaire de l'autorité de certification) et l'émetteur du certificat résident dans un seul AWS compte, la meilleure pratique consiste à séparer les rôles d'émetteur et d'administrateur en créant un utilisateur AWS Identity and Access Management (IAM) doté d'autorisations limitées. Pour plus d'informations sur l'utilisation d'IAM avec Autorité de certification privée AWS, ainsi que des exemples d'autorisations, consultezIdentity and Access Management (IAM) pour AWS Private Certificate Authority.

Cas de compte unique 1 : émission d'un certificat non géré

Dans ce cas, le propriétaire du compte crée une autorité de certification privée, puis un utilisateur IAM autorisé à émettre des certificats signés par l'autorité de certification privée. L'utilisateur IAM émet un certificat en appelant l' Autorité de certification privée AWS `IssueCertificate`API.

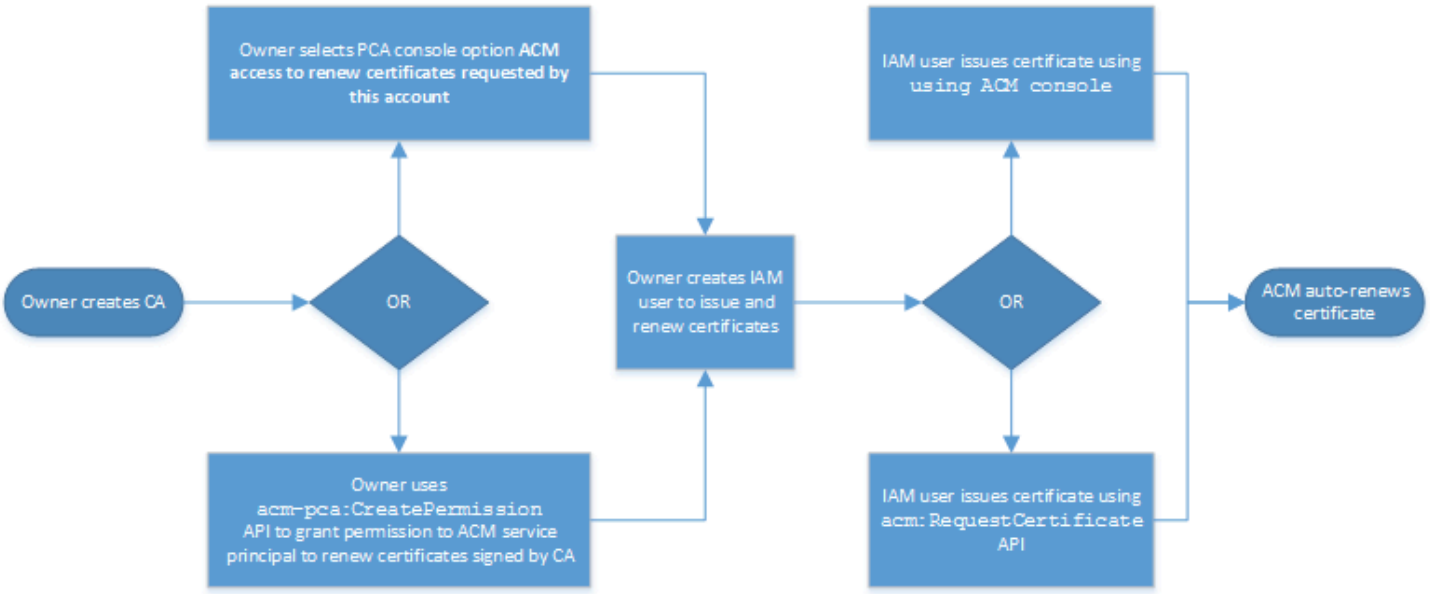


Les certificats émis de cette manière ne sont pas gérés, ce qui signifie qu'un administrateur doit les exporter et les installer sur les appareils sur lesquels ils sont destinés à être utilisés. Ils doivent

également être renouvelés manuellement lorsqu'ils expirent. L'émission d'un certificat à l'aide de cette API nécessite une demande de signature de certificat (CSR) et une paire de clés générées en dehors d' Autorité de certification privée AWS OpenSSL ou d'un programme similaire. Pour plus d'informations, consultez la `IssueCertificate`documentation https://docs.aws.amazon.com/privateca/latest/APIReference/API_IssueCertificate.html.

Cas de compte unique 2 : émission d'un certificat géré via ACM

Ce second cas concerne des opérations d'API provenant à la fois d'ACM et de PCA. Le propriétaire du compte crée un utilisateur CA et IAM privé comme auparavant. Le titulaire du compte autorise ensuite le principal du service ACM à renouveler automatiquement tous les certificats signés par cette autorité de certification. L'utilisateur IAM émet à nouveau le certificat, mais cette fois en appelant l'`RequestCertificate`API ACM, qui gère la CSR et la génération de clés. Lorsque le certificat expire, ACM automatise le processus de renouvellement.



Le titulaire du compte a la possibilité d'accorder une autorisation de renouvellement via la console de gestion pendant ou après la création de l'autorité de certification ou à l'aide de l'`CreatePermission`API PCA. Les certificats gérés créés à partir de ce flux de travail peuvent être utilisés avec AWS des services intégrés à ACM.

La section suivante décrit les procédures d'octroi des autorisations de renouvellement.

Attribuer des autorisations de renouvellement de certificat à ACM

Grâce au renouvellement géré AWS Certificate Manager intégré (ACM), vous pouvez automatiser le processus de renouvellement des certificats publics et privés. Pour qu'ACM renouvelle

automatiquement les certificats générés par une autorité de certification privée, le principal du service ACM doit recevoir toutes les autorisations possibles de la part de l'autorité de certification elle-même. Si ces autorisations de renouvellement ne sont pas présentes pour ACM, le propriétaire de l'autorité de certification (ou un représentant autorisé) doit réémettre manuellement chaque certificat privé à son expiration.

> **Important**
>
> Ces procédures d'attribution des autorisations de renouvellement s'appliquent uniquement lorsque le propriétaire de l'autorité de certification et l'émetteur du certificat résident sur le même AWS compte. Pour les scénarios entre comptes, voirJoindre une politique d'accès entre comptes.

Les autorisations de renouvellement peuvent être déléguées lors de la création de l'autorité de certification privée ou modifiées à tout moment, tant que l'état de l'autorité de certification est `ACTIVE`.

Vous pouvez gérer les autorisations d'une autorité de certification privée à partir de la console Autorité de certification privée AWS, de l'AWS Command Line Interface (AWS CLI) ou de l'API Autorité de certification privée AWS :

Pour attribuer des autorisations CA privées à ACM (console)

1. Connectez-vous à votre AWS compte et ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home.
2. Sur la page Autorités de certification privées, choisissez votre autorité de certification privée dans la liste.
3. Choisissez Actions, puis Configurez les autorisations CA.
4. Sélectionnez Autoriser l'accès à ACM pour renouveler les certificats demandés par ce compte.
5. Choisissez Enregistrer.

Pour gérer les autorisations ACM dans Autorité de certification privée AWS ()AWS CLI

Utilisez la commande create-permission pour attribuer des autorisations à ACM. Vous devez attribuer les autorisations nécessaires (`IssueCertificate`Get`Certificate`, et`ListPermissions`) pour qu'ACM renouvelle automatiquement vos certificats.

```
$ 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
```

Utilisez la commande list-permissions pour répertorier les autorisations déléguées par une autorité de certification.

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

Utilisez la commande delete-permission pour révoquer les autorisations attribuées par une autorité de certification à un AWS principal de service.

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

Joindre une politique d'accès entre comptes

Lorsque l'administrateur de l'autorité de certification et l'émetteur du certificat résident dans des AWS comptes différents, l'administrateur de l'autorité de certification doit partager l'accès à l'autorité de certification. Pour ce faire, une politique basée sur les ressources est attachée à l'autorité de certification. La politique accorde des autorisations d'émission à un principal spécifique, qui peut être un propriétaire de AWS compte, un utilisateur IAM, un AWS Organizations identifiant ou un identifiant d'unité organisationnelle.

Un administrateur de l'autorité de certification peut associer et gérer des politiques de la manière suivante :

- Dans la console de gestion, en utilisant AWS Resource Access Manager (RAM), qui est une méthode standard pour partager AWS des ressources entre comptes. Lorsque vous partagez une ressource CA AWS RAM avec un mandant d'un autre compte, la politique basée sur les ressources requise est automatiquement attachée à l'AC. Pour plus d'informations sur la RAM, consultez le guide de AWS RAM l'utilisateur.

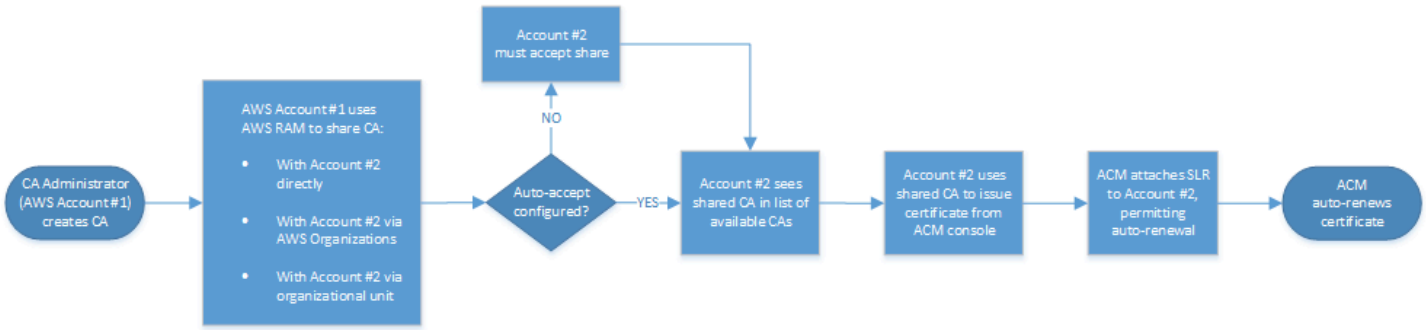
> **Note**
>
> Vous pouvez facilement ouvrir la console RAM en choisissant une autorité de certification, puis en choisissant Actions, Gérer les partages de ressources.

- Par programmation, en utilisant les API PCA PutPolicy, et GetPolicy. DeletePolicy
- Manuellement, à l'aide des commandes PCA put-policy, get-policyet delete-policy dans le. AWS CLI

Seule la méthode console nécessite un accès à la RAM.

Cas 1 entre comptes : émission d'un certificat géré depuis la console

Dans ce cas, l'administrateur de l'autorité de certification utilise AWS Resource Access Manager (AWS RAM) pour partager l'accès de l'autorité de certification avec un autre AWS compte, ce qui permet à ce compte d'émettre des certificats ACM gérés. Le schéma montre qu'il est AWS RAM possible de partager le CA directement avec le compte, ou indirectement par le biais d'un AWS Organizations identifiant dont le compte est membre.



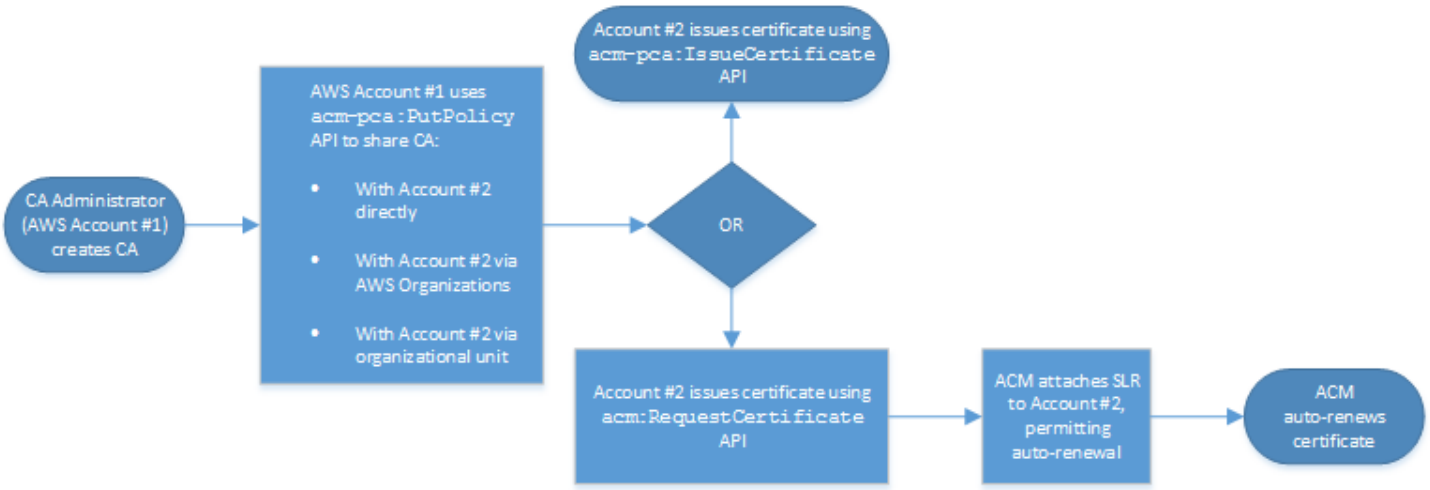
Une fois que la RAM a partagé une ressource AWS Organizations, le principal destinataire doit accepter la ressource pour qu'elle prenne effet. Le destinataire peut configurer AWS Organizations pour accepter automatiquement les actions proposées.

> **Note**
>
> Le compte du destinataire est responsable de la configuration du renouvellement automatique dans ACM. Généralement, lors de la première utilisation d'une autorité de certification partagée, ACM installe un rôle lié à un service qui lui permet de passer des appels de certificat sans surveillance. Autorité de certification privée AWS En cas d'échec (généralement en raison d'une autorisation manquante), les certificats de l'autorité de

certification ne sont pas renouvelés automatiquement. Seul l'utilisateur ACM peut résoudre le problème, pas l'administrateur de l'autorité de certification. Pour plus d'informations, voir Utilisation d'un rôle lié à un service (SLR) avec ACM.

Cas 2 entre comptes : émission de certificats gérés et non gérés à l'aide de l'API ou de la CLI

Ce second cas illustre les options de partage et d'émission possibles à l'aide de l' Autorité de certification privée AWS API AWS Certificate Manager and. Toutes ces opérations peuvent également être effectuées à l'aide des AWS CLI commandes correspondantes.



Dans la mesure où les opérations d'API sont utilisées directement dans cet exemple, l'émetteur du certificat a le choix entre deux opérations d'API pour émettre un certificat. L'action de l'API PCA `IssueCertificate` génère un certificat non géré qui ne sera pas automatiquement renouvelé et qui doit être exporté et installé manuellement. L'action de l'API ACM permet d'RequestCertificateobtenir un certificat géré qui peut être facilement installé sur les services intégrés d'ACM et qui se renouvelle automatiquement.

> **Note**
>
> Le compte du destinataire est responsable de la configuration du renouvellement automatique dans ACM. Généralement, lors de la première utilisation d'une autorité de certification partagée, ACM installe un rôle lié à un service qui lui permet de passer des appels de certificat sans surveillance. Autorité de certification privée AWS En cas d'échec (généralement en raison d'une autorisation manquante), les certificats de l'autorité de certification ne seront pas renouvelés automatiquement et seul l'utilisateur d'ACM pourra

résoudre le problème, et non l'administrateur de l'autorité de certification. Pour plus d'informations, voir Utilisation d'un rôle lié à un service (SLR) avec ACM.

Répertorier des CA privées

Vous pouvez utiliser la Autorité de certification privée AWS console ou AWS CLI répertorier les autorités de certification privées que vous possédez ou auxquelles vous avez accès.

Pour répertorier les autorités de certification disponibles à l'aide de la console

1. Connectez-vous à votre AWS compte et ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home.
2. Consultez les informations de la liste des autorités de certification privées. Vous pouvez parcourir plusieurs pages de CA à l'aide des numéros de page en haut à droite. Chaque CA occupe une ligne avec certaines ou toutes les colonnes suivantes affichées pour chacune d'entre elles :

- Objet — Résumé des informations relatives au nom distinctif de l'autorité de certification.
- Id — Identifiant unique hexadécimal de 32 octets de l'autorité de certification.
- État : statut CA. Les valeurs possibles sont Création, Certificat en attente, Actif, Supprimé, Désactivé, Expiré et Échoué.
- Type : type de CA. Les valeurs possibles sont Root et Subordinate.
- Mode — Le mode de l'autorité de certification. Les valeurs possibles sont les suivantes : usage général (émission de certificats pouvant être configurés avec n'importe quelle date d'expiration) et certificat de courte durée (émission de certificats d'une durée de validité maximale de sept jours). Une courte période de validité peut remplacer dans certains cas un mécanisme de révocation. La valeur par défaut est Usage général.
- Propriétaire : AWS compte auquel appartient l'autorité de certification. Il peut s'agir de votre compte ou d'un compte auquel des autorisations de gestion de CA vous ont été déléguées.
- Algorithme clé : algorithme à clé publique pris en charge par l'autorité de certification. Les valeurs possibles sont RSA_2048, RSA_4096, EC_Prime256v1 et EC_SECP384R1.
- Algorithme de signature : algorithme utilisé par l'autorité de certification pour signer les demandes de certificat. (À ne pas confondre avec le `SigningAlgorithm` paramètre utilisé pour signer les certificats lors de leur émission.) Les valeurs possibles sont SHA256WITHECDSA,

SHA384WITHECDSA, SHA512WITHECDSA, SHA256WITHRSA, SHA384WITHRSA et SHA512WITHRSA.

> **Note**
>
> Vous pouvez personnaliser les colonnes que vous souhaitez afficher, ainsi que d'autres paramètres, en choisissant l'icône des paramètres dans le coin supérieur droit de la console.

Pour répertorier les autorités de certification disponibles à l'aide du AWS CLI

Utilisez la commande list-certificate-authorities pour répertorier les autorités de certification disponibles, comme indiqué dans l'exemple suivant :

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

La commande renvoie des informations semblables à ce qui suit :

```
{
   "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
            }
         }
```

```
      }
  ...
    ]
}
```

Afficher une autorité de certification privée

Vous pouvez utiliser la console ACM ou le AWS CLI pour afficher les métadonnées détaillées d'une autorité de certification privée et modifier plusieurs valeurs selon vos besoins. Pour des informations détaillées sur la mise à jour des autorités de certification, consultezMettre à jour votre CA privée.

Pour afficher les détails de l'autorité de certification dans la console

1. Connectez-vous à votre AWS compte et ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home.
2. Consultez la liste des autorités de certification privées. Vous pouvez parcourir plusieurs pages de CA à l'aide des numéros de page en haut à droite.
3. Pour afficher les métadonnées détaillées d'une autorité de certification répertoriée, cliquez sur le bouton radio à côté de l'autorité de certification que vous souhaitez inspecter. Cela ouvre un volet de détails avec les vues à onglets suivantes :

   - Onglet Objet : informations sur le nom distinctif de l'autorité de certification. Pour plus d'informations, consultez Options de nom distinctif du sujet. Les champs affichés incluent :
     - Objet — Résumé des champs d'information relatifs au nom fournis
     - Organisation (O) — Par exemple, le nom d'une entreprise
     - Unité organisationnelle (UO) — Par exemple, une division au sein d'une entreprise
     - Nom du pays (C) — Code de pays à deux lettres
     - Nom de l'État ou de la province — Nom complet d'un État ou d'une province
     - Nom de la localité — Le nom d'une ville
     - Nom commun (CN) : chaîne lisible par l'homme pour identifier l'autorité de certification.
   - Onglet Certificat CA : informations sur la validité du certificat CA
     - Valable jusqu'à : date et heure de validité du certificat CA
     - Expire en : nombre de jours avant l'expiration
   - Onglet de configuration de révocation : vos sélections actuelles pour les options de révocation des certificats. Choisissez Modifier pour effectuer la mise à jour.

    - Distribution de listes de révocation de certificats (CRL) — État activé ou désactivé
    - Protocole OCSP (Online Certificate Status Protocol) — État activé ou désactivé
  - Onglet Autorisations : votre sélection actuelle d'autorisations de renouvellement de certificat pour ce CA via AWS Certificate Manager (ACM). Choisissez Modifier pour effectuer la mise à jour.
  - Autorisation ACM pour les renouvellements — État d'autorisation ou d'autorisation
  - Onglet Tags — Votre attribution actuelle d'étiquettes personnalisables pour cette autorité de certification. Choisissez l'option Gérer les balises à mettre à jour.
  - Onglet Partage des ressources — Votre attribution actuelle de partages de ressources pour cette autorité de certification via AWS Resource Access Manager (RAM). Choisissez Gérer les partages de ressources à mettre à jour.
    - Nom : nom du partage de ressources
    - État : état du partage des ressources
4. Choisissez le champ ID de l'autorité de certification que vous souhaitez inspecter pour ouvrir le volet Général. L'identifiant unique hexadécimal de 32 octets de l'autorité de certification apparaît en haut. Le volet fournit les informations supplémentaires suivantes :

   - État : statut CA. Les valeurs possibles sont Création, Certificat en attente, Actif, Supprimé, Désactivé, Expiré et Échoué.
   - ARN — Le nom de la ressource Amazon pour l'autorité de certification.
   - Propriétaire : AWS compte auquel appartient l'autorité de certification. Il peut s'agir de votre compte (Self) ou d'un compte auquel des autorisations de gestion de CA vous ont été déléguées.
   - Type de CA : type de CA. Les valeurs possibles sont Root et Subordinate.
   - Créé à : date et heure de création de l'autorité de certification.
   - Date d'expiration : date et heure d'expiration du certificat CA.
   - Mode — Le mode de l'autorité de certification. Les valeurs possibles sont les suivantes : certificats à usage général (certificats pouvant être configurés avec n'importe quelle date d'expiration) et certificat de courte durée (certificats dont la durée de validité maximale est de sept jours). Une courte période de validité peut remplacer dans certains cas un mécanisme de révocation. La valeur par défaut est Usage général.
   - Algorithme clé : algorithme à clé publique pris en charge par l'autorité de certification. Les valeurs possibles sont RSA 2048, RSA 4096, ECDSA P2567 et ECDSA P384.

- Algorithme de signature : algorithme utilisé par l'autorité de certification pour signer les demandes de certificat. (À ne pas confondre avec le `SigningAlgorithm` paramètre utilisé pour signer les certificats lors de leur émission.) Les valeurs possibles sont SHA256 ECDSA, SHA384 ECDSA, SHA512 ECDSA, SHA256 RSA, SHA384 RSA et SHA512 RSA
- Norme de sécurité du stockage des clés — Niveau de conformité aux normes fédérales de traitement de l'information. Les valeurs possibles sont FIPS 140-2 niveau 3 ou supérieur et FIPS 140-2 niveau 3 ou supérieur. Ce paramètre varie selon les AWS régions.

Pour afficher et modifier les détails de l'autorité de certification à l'aide du AWS CLI

Utilisez la describe-certificate-authoritycommande du AWS CLI pour afficher les détails d'une autorité de certification, comme indiqué dans la commande suivante :

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

La commande renvoie des informations semblables à ce qui suit :

```
{
   "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
         }
      }
   }
```

```
}
```

Pour plus d'informations sur la mise à jour d'une autorité de certification privée depuis la ligne de commande, consultezMettre à jour une autorité de certification (CLI).

Gestion des tags pour votre autorité de certification privée

Les balises sont des mots ou des expressions qui jouent le rôle de métadonnées pour identifier et organiser les ressources AWS . Chaque balise se compose d'une clé et d'une valeur. Vous pouvez utiliser la Autorité de certification privée AWS console, AWS Command Line Interface (AWS CLI) ou l'API PCA pour ajouter, afficher ou supprimer des balises pour les autorités de certification privées.

Vous pouvez ajouter ou supprimer des balises personnalisées pour votre autorité de certification privée à tout moment. Par exemple, vous pouvez étiqueter des autorités de certification privées avec des paires clé-valeur telles que `Environment=Prod` ou `Environment=Beta` pour identifier l'environnement auquel l'autorité de certification est destinée. Pour plus d'informations, voir Création d'une autorité de certification privée.

> **Note**
>
> Pour associer des balises à une autorité de certification privée au cours de la procédure de création, un administrateur de l'autorité de certification doit d'abord associer une politique IAM intégrée à l'`CreateCertificateAuthority`action et autoriser explicitement le balisage. Pour plus d'informations, consultez T ag-on-create : Attacher des tags à une autorité de certification au moment de sa création.

D'autres AWS ressources prennent également en charge le balisage. Vous pouvez attribuer le même tag à différentes ressources pour indiquer que ces ressources sont liées. Par exemple, vous pouvez attribuer une balise telle que `Website=example.com` à votre autorité de certification, à l'équilibreur de charge Elastic Load Balancing et à d'autres ressources connexes. Pour plus d'informations sur le balisage AWS des ressources, consultez la section Marquage de vos ressources Amazon EC2 dans le guide de l'utilisateur Amazon EC2.

Les restrictions de base suivantes s'appliquent aux Autorité de certification privée AWS tags :

- Le nombre maximal de balises par autorité de certification est de 50.
- La longueur maximale d'une clé de balise est de 128 caractères.

- La longueur maximale d'une valeur de balise est de 256 caractères.
- La clé et la valeur de la balise peuvent contenir les caractères suivants : A-Z, a-z et .:+=@_%-(trait d'union).
- Les clés et valeurs d'étiquette sont sensibles à la casse.
- Les préfixes `aws:` et `rds:` sont réservés à AWS . Vous ne pouvez pas ajouter, modifier ou supprimer des balises dont la clé commence par `aws:` ou `rds:`. Tags par défaut qui commencent par votre tags-per-resource quota `aws:` et `rds:` ne sont pas pris en compte dans celui-ci.
- Si vous prévoyez d'utiliser votre schéma de balisage pour plusieurs services et ressources, n'oubliez pas que d'autres services peuvent avoir des restrictions différentes en ce qui concerne les caractères autorisés. Reportez-vous à la documentation correspondant à ce service.
- Autorité de certification privée AWS les balises ne peuvent pas être utilisées dans les Resource Groups et l'éditeur de balises dans le AWS Management Console.

Vous pouvez baliser une autorité de certification privée à partir de la console Autorité de certification privée AWS, de l'AWS Command Line Interface (AWS CLI) ou de l'API Autorité de certification privée AWS.

Pour baliser une autorité de certification privée (console)

1. Connectez-vous à votre AWS compte et ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home.
2. Sur la page Autorités de certification privées, choisissez votre autorité de certification privée dans la liste.
3. Dans la zone de détails située sous la liste, choisissez l'onglet Tags. La liste des balises existantes s'affiche.
4. Choisissez Gérer les balises.
5. Choisissez Add new tag (Ajouter une nouvelle balise).
6. Saisissez une paire clé-valeur.
7. Choisissez Enregistrer.

Pour baliser une autorité de certification privée (AWS CLI)

Utilisez la commande tag-certificate-authority pour ajouter des balises à votre autorité de certification privée.

```
$ aws acm-pca tag-certificate-authority \
     --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-
authority/CA_ID \
     --tags Key=Admin,Value=Alice
```

Utilisez la commande list-tags pour afficher les balises pour une autorité de certification privée.

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

Utilisez la commande untag-certificate-authority pour supprimer des balises d'une autorité de certification privée.

```
$ aws acm-pca untag-certificate-authority \
     --certificate-authority-arn arn:aws:acm-pca:aregion:account:certificate-
authority/CA_ID \
     --tags Key=Purpose,Value=Website
```

Mettre à jour votre CA privée

Vous pouvez mettre à jour le statut d'une autorité de certification privée ou modifier sa configuration de révocation après l'avoir créée. Cette rubrique fournit des informations détaillées sur le statut de l'autorité de certification et le cycle de vie de l'autorité de certification, ainsi que des exemples de mises à jour de la console et de la CLI destinées aux autorités de certification.

Mettre à jour le statut de CA

Le statut d'une autorité de certification gérée par Autorité de certification privée AWS résulte d'une action de l'utilisateur ou, dans certains cas, d'une action de service. Par exemple, le statut d'une autorité de certification change lorsqu'il expire. Les options d'état disponibles pour les administrateurs de l'autorité de certification varient en fonction de l'état actuel de l'autorité de certification.

Autorité de certification privée AWS peut signaler les valeurs d'état suivantes. Le tableau indique les fonctionnalités de l'autorité de certification disponibles dans chaque État.

> **Note**
>
> Pour toutes les valeurs de statut, à l'exception de `DELETED` et`FAILED`, l'autorité de certification vous est facturée.

Statut	Émettre des certificats	Valider les certificats avec OCSP	Générer des CRL	Générer des audits	Vous pouvez mettre à jour le certificat CA	Les certificats peuvent être révoqués	Le CA vous est facturé
`CREATING`— Le CA est en cours de création.	Non	Non	Non	Non	Non	Non	Oui
`PENDING_CERTIFICATE` — L'autorité de certification a été créée et a besoin d'un certificat pour être opérationnelle. *	Non	Non	Non	Non	Non	Non	Oui
`ACTIVE`	Oui	Oui	Oui	Oui	Oui	Oui	Oui
`DISABLED`— Vous avez désactivé manuellement le CA.	Non	Oui	Oui	Oui	Non	Oui	Oui
`EXPIRED`— Le certificat CA a expiré. **	Non	Non	Non	Non	Oui	Non	Oui
`FAILED`	L'`CreateCertificateAuthority` action a échoué. Cela peut être dû à une panne réseau, à une AWS défaillance du						Non

Statut	Émettre des certificats	Valider les certificats avec OCSP	Génére des CRL	Génére des audits	Vous pouvez mettre à jour le certificat CA	Les certificats peuvent être révoqués	Le CA vous est facturé
	backend ou à d'autres erreurs. Une autorité de certification défaillante ne peut pas être récupérée. Supprimez l'autorité de certification et créez-en une nouvelle.						
`DELETED`	Votre CA est en période de restauration, qui peut durer de 7 à 30 jours. Après cette période, elle est définitivement supprimé. • Si vous appelez l'API `RestoreCertificateAuthority` sur une autorité de certification dont l'état est `DELETED` et dont l'un des certificats a expiré, l'autorité de certification sera définie sur `EXPIRED`. • Pour de plus amples informations sur la suppression d'une autorité de certification, veuillez consulter Supprimer votre CA privée.						Non

* Pour terminer l'activation, vous devez générer un CSR, obtenir un certificat d'autorité de certification signé auprès d'une autorité de certification et importer le certificat dans Autorité de certification privée AWS. Le CSR peut être soumis soit à votre nouvelle autorité de certification (pour l'auto-signature), soit à une autorité de certification racine ou subordonnée sur site. Pour plus d'informations, consultez Création et installation du certificat CA.

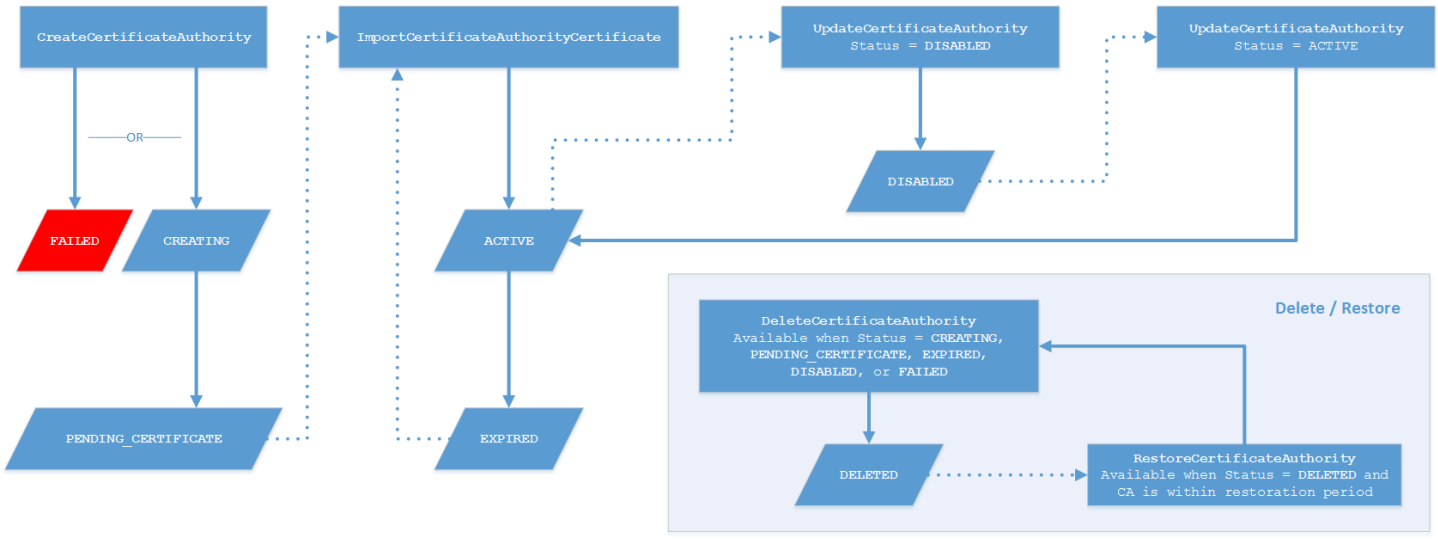
** Vous ne pouvez pas modifier directement le statut d'une autorité de certification expirée. Si vous importez un nouveau certificat pour l'autorité de certification, le statut Autorité de certification privée AWS est rétabli à `ACTIVE` moins qu'il n'ait été défini `DISABLED` avant l'expiration du certificat.

Considérations supplémentaires concernant les certificats CA expirés :

- Les certificats CA ne sont pas renouvelés automatiquement. Pour plus d'informations sur l'automatisation du renouvellement AWS Certificate Manager, consultezAttribuer des autorisations de renouvellement de certificat à ACM.
- Si vous tentez d'émettre un nouveau certificat avec une autorité de certification expirée, l'API `IssueCertificate` renvoie `InvalidStateException`. Une autorité de certification racine expirée doit auto-signer un nouveau certificat d'une autorité de certification racine avant de pouvoir émettre de nouveaux certificats subordonnés.
- The `ListCertificateAuthorities` et l'API `DescribeCertificateAuthority` renvoient l'état `EXPIRED` si le certificat d'une autorité de certification est expiré, que l'état de l'autorité de certification soit défini sur `ACTIVE` ou `DISABLED`. Toutefois, si l'autorité de certification expirée a été définie sur `DELETED`, l'état renvoyé est `DELETED`.
- L'API `UpdateCertificateAuthority` ne peut pas mettre à jour l'état d'une autorité de certification expirée.
- L'`RevokeCertificate`API ne peut pas être utilisée pour révoquer un certificat expiré, y compris un certificat CA.

État et cycle de vie de l'autorité de certification

Le diagramme suivant illustre le cycle de vie de l'autorité de certification en tant qu'interaction des actions de gestion avec l'état de l'autorité de certification.



Clé du diagramme

Action de gestion	Statut CA	L'action entraîne un changement d'état	Le nouvel état permet une nouvelle action

En haut du diagramme, les actions de gestion sont appliquées via la console Autorité de certification privée AWS , l'interface de ligne de commande ou l'API. Les actions font transiter l'autorité de certification via la création, l'activation, l'expiration et le renouvellement. L'état de l'autorité de certification change en réponse (comme indiqué par les lignes pleines) à des actions manuelles ou à des mises à jour automatisées. Dans la plupart des cas, un nouvel état entraîne une nouvelle action possible (représentée par une ligne pointillée) que l'administrateur de l'autorité de certification peut appliquer. L'encadré en bas à droite illustre les valeurs d'état possibles permettant les actions de suppression et de restauration.

Mettre à jour une autorité de certification (console)

Les procédures suivantes montrent comment mettre à jour les configurations CA existantes à l'aide du AWS Management Console.

Mettre à jour le statut de l'autorité de certification (console)

Dans cet exemple, le statut d'une autorité de certification activée devient désactivé.

Pour mettre à jour le statut d'une autorité de certification

1. Connectez-vous à votre AWS compte et ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home
2. Sur la page Autorités de certification privées, choisissez une autorité de certification privée actuellement active dans la liste.
3. Dans le menu Actions, choisissez Désactiver pour désactiver l'autorité de certification privée.

Mettre à jour la configuration de révocation d'une autorité de certification (console)

Vous pouvez mettre à jour la configuration de révocation de votre autorité de certification privée, par exemple en ajoutant ou en supprimant le support OCSP ou CRL, ou en modifiant leurs paramètres.

> **Note**
>
> Les modifications apportées à la configuration de révocation d'une autorité de certification n'affectent pas les certificats déjà émis. Pour que la révocation gérée fonctionne, les anciens certificats doivent être réémis.

Pour OCSP, vous modifiez les paramètres suivants :

- Activez ou désactivez OCSP.
- Activez ou désactivez un nom de domaine complet OCSP personnalisé (FQDN).
- Modifiez le FQDN.

Pour une CRL, vous pouvez modifier l'un des paramètres suivants :

- Si l'autorité de certification privée génère ou non une liste de révocation de certificats
- Le nombre de jours avant qu'une liste de révocation de certificats expire. Notez que la tentative de régénération de la CRL Autorité de certification privée AWS commence à la moitié du nombre de jours que vous spécifiez.
- Le nom du compartiment Amazon S3 dans lequel votre CRL est enregistrée.
- Un alias pour masquer le nom de votre compartiment Amazon S3 à la vue du public.

> **Important**
>
> La modification de l'un des paramètres précédents peut avoir des effets négatifs. Les exemples incluent la désactivation de la génération de CRL, la modification de la période de validité ou la modification du compartiment S3 une fois que vous avez placé votre autorité de certification privée en production. De telles modifications peuvent endommager les certificats existants qui dépendent de la CRL et de la configuration actuelle de la CRL. Il est possible de modifier l'alias en toute sécurité, à condition que l'ancien alias reste lié au compartiment correct.

Pour mettre à jour les paramètres de révocation

1. Connectez-vous à votre AWS compte et ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home.
2. Sur la page Autorités de certification privées, choisissez une autorité de certification privée dans la liste. Cela ouvre le panneau de détails de l'autorité de certification.
3. Choisissez l'onglet Configuration de la révocation, puis sélectionnez Modifier.
4. Sous Options de révocation du certificat, deux options sont affichées :

   - Activer la distribution CRL
   - Activez OCSP

   Vous pouvez configurer l'un ou l'autre de ces mécanismes de révocation, ou les deux, pour votre autorité de certification. Bien que facultative, la gestion des révocations est recommandée en tant que bonne pratique. Avant de terminer cette étape, consultez Configuration d'une méthode de révocation des certificats les informations sur les avantages de chaque méthode, la configuration préliminaire qui peut être requise et les fonctionnalités de révocation supplémentaires.

Pour configurer une CRL

1. Sélectionnez Activer la distribution CRL.
2. Pour créer un compartiment Amazon S3 pour vos entrées CRL, sélectionnez Créer un nouveau compartiment S3. Indiquez un nom de compartiment unique. (Vous n'avez pas besoin d'inclure le chemin d'accès au compartiment.) Sinon, laissez cette option désactivée et choisissez un bucket existant dans la liste des noms de bucket S3.

   Si vous créez un nouveau bucket, il Autorité de certification privée AWS crée et associe la politique d'accès requise à celui-ci. Si vous décidez d'utiliser un bucket existant, vous devez lui associer une politique d'accès avant de pouvoir commencer à générer des CRL. Utilisez l'un des modèles de politique décrits dansPolitiques d'accès pour les CRL dans Amazon S3 . Pour plus d'informations sur l'attachement d'une politique, consultez Ajouter une politique de compartiment à l'aide de la console Amazon S3.

> **Note**
>
> Lorsque vous utilisez la Autorité de certification privée AWS console, toute tentative de création d'une autorité de certification échoue si les deux conditions suivantes s'appliquent :
>
> - Vous appliquez les paramètres de blocage de l'accès public sur votre compartiment ou compte Amazon S3.
> - Vous avez demandé Autorité de certification privée AWS de créer automatiquement un compartiment Amazon S3.
>
> Dans ce cas, la console tente, par défaut, de créer un compartiment accessible au public, et Amazon S3 rejette cette action. Vérifiez vos paramètres Amazon S3 dans ce cas. Pour plus d'informations, consultez Blocage de l'accès public à votre espace de stockage Amazon S3.

3. Développez Avancé pour obtenir des options de configuration supplémentaires.

   - Ajoutez un nom CRL personnalisé pour créer un alias pour votre compartiment Amazon S3. Ce nom est contenu dans les certificats émis par l'autorité de certification dans l'extension « CRL Distribution Points » définie par la RFC 5280.
   - Tapez le nombre de jours de validité de votre liste de révocation de certificats. La valeur par défaut est 7 jours. Pour les CRL en ligne, une période de validité de 2 à 7 jours est courante. Autorité de certification privée AWS essaie de régénérer la CRL au milieu de la période spécifiée.

4. Choisissez Enregistrer les modifications lorsque vous avez terminé.

Pour configurer OCSP

1. Sur la page de révocation du certificat, choisissez Activer l'OCSP.
2. (Facultatif) Dans le champ Point de terminaison OCSP personnalisé, fournissez un nom de domaine complet (FQDN) pour votre point de terminaison OCSP.

   Lorsque vous fournissez un FQDN dans ce champ, Autorité de certification privée AWS insère le FQDN dans l'extension Authority Information Access de chaque certificat émis à la place de l'URL par défaut du répondeur AWS OCSP. Lorsqu'un point de terminaison reçoit un certificat

contenant le nom de domaine complet personnalisé, il demande à cette adresse une réponse OCSP. Pour que ce mécanisme fonctionne, vous devez effectuer deux actions supplémentaires :

- Utilisez un serveur proxy pour transférer le trafic qui arrive à votre nom de domaine complet personnalisé vers le répondeur AWS OCSP.
- Ajoutez un enregistrement CNAME correspondant à votre base de données DNS.

> **Tip**
>
> Pour plus d'informations sur la mise en œuvre d'une solution OCSP complète à l'aide d'un CNAME personnalisé, consultez. Configuration d'une URL personnalisée pour Autorité de certification privée AWS OCSP

Par exemple, voici un enregistrement CNAME pour un OCSP personnalisé tel qu'il apparaîtrait dans Amazon Route 53.

Nom de l'enregistrement	Type	Stratégie de routage	Différenciateur	Valeur/acheminer le trafic vers
alternative.exemple.com	CNAME	Simplicité	-	proxy.exemple.com

> **Note**
>
> La valeur du CNAME ne doit pas inclure de préfixe de protocole tel que « http ://» ou « https ://».

3. Choisissez Enregistrer les modifications lorsque vous avez terminé.

Mettre à jour une autorité de certification (CLI)

Les procédures suivantes montrent comment mettre à jour le statut et la configuration de révocation d'une autorité de certification existante à l'aide du AWS CLI.

> **Note**
>
> Les modifications apportées à la configuration de révocation d'une autorité de certification n'affectent pas les certificats déjà émis. Pour que la révocation gérée fonctionne, les anciens certificats doivent être réémis.

Pour mettre à jour le statut de votre autorité de certification privée (AWS CLI)

Utilisez la commande update-certificate-authority.

Cela est utile lorsque vous avez une autorité de certification existante `DISABLED` dont vous souhaitez définir le statut`ACTIVE`. Pour commencer, confirmez le statut initial de l'autorité de certification à l'aide de la commande suivante.

```
$ 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
```

Il en résulte un résultat similaire à ce qui suit.

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

La commande suivante définit le statut de l'autorité de certification privée surACTIVE. Cela n'est possible que si un certificat valide est installé sur l'autorité de certification.

```
$ 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"
```

Vérifiez le nouveau statut de l'autorité de certification.

```
$ 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
```

Le statut apparaît désormais sous la formeACTIVE.

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

Dans certains cas, vous pouvez avoir une autorité de certification active sans mécanisme de révocation configuré. Si vous souhaitez commencer à utiliser une liste de révocation de certificats (CRL), suivez la procédure ci-dessous.

Pour ajouter une CRL à une autorité de certification existante ()AWS CLI

1. Utilisez la commande suivante pour vérifier l'état actuel de l'autorité de certification.

```
$ 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
```

La sortie confirme que l'autorité de certification a un statut `ACTIVE` mais qu'elle n'est pas configurée pour utiliser une 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. Créez et enregistrez un fichier avec un nom tel que `revoke_config.txt` pour définir vos paramètres de configuration CRL.

```
{
   "CrlConfiguration":{
```

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

> **Note**
>
> Lorsque vous mettez à jour une autorité de certification d'appareil Matter pour activer les CRL, vous devez la configurer pour omettre l'extension CDP des certificats émis afin de vous conformer à la norme Matter actuelle. Pour ce faire, définissez vos paramètres de configuration CRL comme illustré ci-dessous :
>
> ```
> {
>    "CrlConfiguration":{
>       "Enabled": true,
>       "ExpirationInDays": 7,
>       "S3BucketName": "bucket-name"
>       "CrlDistributionPointExtensionConfiguration":{
>          "OmitExtension": true
>       }
>    }
> }
> ```

3. Utilisez la commande update-certificate-authority et le fichier de configuration de révocation pour mettre à jour l'autorité de certification.

```
$ 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. Vérifiez à nouveau l'état de l'autorité de certification.

```
$ 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
```

Le résultat confirme que CA est désormais configuré pour utiliser une 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
            }
        }
    }
}
```

Dans certains cas, vous souhaiterez peut-être ajouter le support de révocation OCSP au lieu d'activer une CRL comme dans la procédure précédente. Dans ce cas, suivez les étapes ci-dessous.

Pour ajouter le support OCSP à une autorité de certification existante ()AWS CLI

1. Créez et enregistrez un fichier avec un nom tel que `revoke_config.txt` pour définir vos paramètres OCSP.

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

2. Utilisez la commande update-certificate-authority et le fichier de configuration de révocation pour mettre à jour l'autorité de certification.

```
$ 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. Vérifiez à nouveau l'état de l'autorité de certification.

```
$ 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
```

Le résultat confirme que CA est désormais configuré pour utiliser 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**
>
> Vous pouvez également configurer le support CRL et OCSP sur une autorité de certification.

Supprimer votre CA privée

Vous pouvez supprimer AWS CLI définitivement une autorité de certification privée du AWS Management Console ou. Vous pouvez par exemple en supprimer une pour la remplacer par une nouvelle autorité de certification possédant une nouvelle clé privée. Pour supprimer une autorité de certification en toute sécurité, procédez comme suit :

1. Créez l'autorité de certification de remplacement.
2. Une fois que la nouvelle autorité de certification privée est en production, désactivez l'ancienne, mais ne la supprimez pas immédiatement.
3. Conservez l'ancienne autorité de certification désactivée jusqu'à ce que tous les certificats qu'elle a émis aient expiré.
4. Supprimez l'ancienne autorité de certification.

Autorité de certification privée AWS ne vérifie pas que tous les certificats émis ont expiré avant de traiter une demande de suppression. Vous pouvez générer un rapport d'audit pour déterminer quels certificats ont expiré. Pendant la désactivation de l'autorité de certification, vous pouvez révoquer des certificats, mais vous ne pouvez pas en émettre de nouveaux.

Si vous devez supprimer une autorité de certification privée avant que tous les certificats qu'elle a émis aient expiré, nous vous recommandons de révoquer également le certificat de l'autorité de certification. Le certificat de l'autorité de certification sera répertorié dans la liste de révocation de certificats de l'autorité de certification parent, et l'autorité de certification privée sera non approuvée par les clients.

> **Important**
>
> Une autorité de certification privée peut être supprimée si son état est `PENDING_CERTIFICATE`, `CREATING`, `EXPIRED`, `DISABLED` ou `FAILED`. Pour supprimer une autorité de certification à l'état `ACTIVE`, vous devez tout d'abord la désactiver. Dans le cas contraire, la demande de suppression entraîne une exception. Si vous supprimez une autorité de certification privée à l'`DISABLED`état `PENDING_CERTIFICATE` ou, vous pouvez définir la durée de sa période de restauration de 7 à 30 jours, 30 étant la valeur par défaut. Pendant cette période, l'état est défini sur `DELETED` et l'autorité de certification peut être restaurée. Aucune période de restauration n'est attribuée à une autorité de certification privée supprimée alors qu'elle est dans `FAILED` l'état `CREATING` ou ne peut pas être restaurée. Pour plus d'informations, consultez Restauration d'une autorité de certification privée.
> Vous n'êtes pas facturé pour l'autorité de certification privée après sa suppression. Toutefois, si vous restaurez une autorité de certification privée supprimée, vous serez facturé pour la durée écoulée entre la suppression et la restauration. Pour plus d'informations, consultez Tarification.

Pour supprimer une autorité de certification privée (console)

1. Connectez-vous à votre AWS compte et ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home.
2. Sur la page Autorités de certification privées, choisissez votre autorité de certification privée dans la liste.

3. Si votre autorité de certification est dans `ACTIVE` cet État, vous devez d'abord la désactiver. Dans le menu Actions , sélectionnez Disable. Lorsque vous y êtes invité, choisissez Je comprends le risque, continuez.
4. Pour une autorité de certification qui n'est pas dans `ACTIVE` cet état, choisissez Actions, Supprimer.
5. Si votre autorité de certification est dans l'`PENDING_CERTIFICATE`état `DISABLED`, `EXPIRED`, ou, la page Supprimer l'autorité de certification vous permet de spécifier une période de restauration de 7 à 30 jours. Si votre autorité de certification privée ne se trouve pas dans l'un de ces états, elle ne pourra pas être restaurée ultérieurement et la suppression est définitive.
6. Sélectionnez Delete (Supprimer).
7. Si vous voulez vraiment supprimer l'autorité de certification privée, choisissez Permanently delete dans l'invite. L'état de l'autorité de certification privée passe à `DELETED`. Toutefois, vous pouvez restaurer l'autorité de certification privée avant la fin de la période de restauration. Pour vérifier la période de restauration d'une autorité de certification privée dans l'`DELETED`État, appelez l'opération DescribeCerticateAuthority ou ListCertificateAuthorities API.

Pour supprimer une autorité de certification privée (AWS CLI)

Utilisez la commande delete-certificate-authority pour supprimer une autorité de certification privée.

```
$ 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
```

Restauration d'une autorité de certification privée

Vous pouvez restaurer une autorité de certification privée qui a été supprimée tant qu'elle reste comprise dans la période de restauration que vous avez spécifiée au moment de la suppression. La période de restauration est de 7 à 30 jours. L'autorité de certification sera supprimée définitivement à la fin de cette période. Pour plus d'informations, consultez Supprimer votre CA privée. Vous ne pouvez pas restaurer une autorité de certification privée qui a été supprimée définitivement.

> **Note**
>
> Vous n'êtes pas facturé pour l'autorité de certification privée après sa suppression. Toutefois, si vous restaurez une autorité de certification privée supprimée, vous serez facturé pour la

durée écoulée entre la suppression et la restauration. Pour plus d'informations, consultez Tarification.

Restauration d'une autorité de certification privée (console)

Vous pouvez utiliser le AWS Management Console pour restaurer une autorité de certification privée.

Pour restaurer une autorité de certification privée (console)

1. Connectez-vous à votre AWS compte et ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-pca/home.
2. Sur la page Autorités de certification privées, choisissez votre autorité de certification privée supprimée dans la liste.
3. Dans le menu Actions , choisissez Restaurer.
4. Sur la page Restore CA, sélectionnez Restaurer à nouveau.
5. En cas de réussite, l'autorité de certification privée récupère l'état qu'elle avait avant sa suppression. Choisissez Actions, Activer et Activer à nouveau pour changer son statut en`ACTIVE`. Si l'état de l'autorité de certification privée était `PENDING_CERTIFICATE` au moment de la suppression, vous devez importer un certificat dans l'autorité de certification privée avant de pouvoir l'activer.

Restauration d'une autorité de certification privée (AWS CLI)

Utilisez la commande pour restore-certificate-authority pour restaurer une autorité de certification privée dont l'état correspond à `DELETED`. Les étapes suivantes présentent le processus requis pour supprimer, restaurer, puis réactiver une autorité de certification privée.

Pour supprimer, restaurer et réactiver une autorité de certification privée (AWS CLI)

1. Supprimez l'autorité de certification privée.

   Exécutez la commande delete-certificate-authority pour supprimer l'autorité de certification privée. Si le statut de l'autorité de certification privée est `DISABLED` ou`PENDING_CERTIFICATE`, vous pouvez définir le `--permanent-deletion-time-in-days` paramètre pour spécifier la période de restauration de l'autorité de certification privée comprise entre 7 et 30 jours. Si vous ne spécifiez pas une période de restauration, le nombre de jours par défaut est utilisé, à

savoir 30 jours. Si elle aboutit, cette commande définit l'état de l'autorité de certification privée sur `DELETED`.

> **Note**
>
> Pour pouvoir être restaurée, l'autorité de certification privée doit avoir l'état `DISABLED` ou `PENDING_CERTIFICATE` au moment de la suppression.

```
$ 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. Restaurez l'autorité de certification privée.

   Exécutez la commande restore-certificate-authority pour restaurer l'autorité de certification privée. Vous devez exécuter la commande avant que la période de restauration que vous définissez avec la commande delete-certificate-authority n'expire. En cas de réussite, la commande attribue à l'autorité de certification privée l'état qu'elle avait avant sa suppression.

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

3. Rendez l'autorité de certification privée `ACTIVE`.

   Exécutez la commande update-certificate-authority pour remplacer l'état de l'autorité de certification privée par `ACTIVE`.

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

Administration des certificats

Après avoir créé et activé une autorité de certification (CA) privée et configuré l'accès à celle-ci, vous ou vos utilisateurs autorisés pouvez effectuer les tâches décrites dans cette section. Si vous n'avez pas encore défini de politiques AWS Identity and Access Management (IAM) pour l'autorité de certification, vous pouvez en savoir plus sur leur configuration dans la section Identity and Access Management de ce guide. Pour plus d'informations sur la configuration de l'accès CA dans les scénarios à compte unique et multicompte, consultez. Contrôle de l'accès à une autorité de certification privée

Rubriques

Émission de certificats d'entité finale privés

Une fois qu'une autorité de certification privée est en place, vous pouvez demander des certificats d'entité finale privés à AWS Certificate Manager (ACM) ou à. Autorité de certification privée AWS Les fonctionnalités des deux services sont comparées dans le tableau suivant.

Capacité	ACM	Autorité de certification privée AWS
Émettre des certificats d'entité finale	✓ (à l'aide RequestCertificate de la console)	✓ (en utilisant IssueCertificate)
Association avec des équilibreurs de charge et des services connectés à Internet AWS	✓	Non pris en charge

Capacité	ACM	Autorité de certification privée AWS
Renouvellement géré des certificats	✓	Soutenu indirectement par ACM
Prise en charge de la console	✓	Non pris en charge
Prise en charge de l'API	✓	✓
Prise en charge de CLI	✓	✓

Lors de Autorité de certification privée AWS la création d'un certificat, il suit un modèle qui spécifie le type de certificat et la longueur du chemin. Si aucun ARN de modèle n'est fourni à l'API ou à l'instruction CLI créant le certificat, le modèle EndEntityCertificate/V1 est appliqué par défaut. Pour de plus amples informations sur les modèles de certificats disponibles, veuillez consulter Comprendre les modèles de certificats.

Bien que les certificats ACM soient conçus autour de la confiance du public, Autorité de certification privée AWS ils répondent aux besoins de votre PKI privée. Par conséquent, vous pouvez configurer les certificats à l'aide de l'Autorité de certification privée AWSAPI et de la CLI d'une manière non autorisée par ACM. Tel est le cas des éléments suivants :

- Création d'un certificat avec n'importe quel nom de sujet.
- En utilisant l'un des algorithmes de clé privée et des longueurs de clé pris en charge.
- En utilisant l'un des algorithmes de signature pris en charge.
- Spécifier la période de validité de votre autorité de certification privée et de vos certificats privés.

Après avoir créé un certificat TLS privé à l'aide deAutorité de certification privée AWS, vous pouvez l'importer dans ACM et l'utiliser avec un service pris en chargeAWS.

> **Note**
>
> Les certificats créés à l'aide de la procédure ci-dessous, à l'aide de la issue-certificate commande ou de l'action de l'IssueCertificateAPI ne peuvent pas être directement exportés pour être utilisés à l'extérieurAWS. Toutefois, vous pouvez utiliser votre autorité de certification privée pour signer les certificats émis via ACM, et ces certificats peuvent être

exportés avec leurs clés secrètes. Pour plus d'informations, consultez les sections Demande d'un certificat privé et Exportation d'un certificat privé dans le guide de l'utilisateur d'ACM.

Délivrance d'un certificat standard (AWS CLI)

Vous pouvez utiliser la commande issue-certificate de la Autorité de certification privée AWS CLI ou l'action API IssueCertificatepour demander un certificat d'entité finale. Cette commande nécessite l'Amazon Resource Name (ARN) de l'autorité de certification privée que vous souhaitez utiliser pour émettre le certificat. Vous devez également générer une demande de signature de certificat (CSR) à l'aide d'un programme tel qu'OpenSSL.

Si vous utilisez l'Autorité de certification privée AWSAPI ou AWS CLI pour émettre un certificat privé, le certificat n'est pas géré, ce qui signifie que vous ne pouvez pas utiliser la console ACM, la CLI ACM ou l'API ACM pour le consulter ou l'exporter, et le certificat n'est pas automatiquement renouvelé. Toutefois, vous pouvez utiliser la commande PCA get-certificate pour récupérer les détails du certificat, et si vous êtes le propriétaire de l'autorité de certification, vous pouvez créer un rapport d'audit.

Considérations relatives à la création de certificats

- Conformément à la RFC 5280, la longueur du nom de domaine (techniquement, le nom commun) que vous fournissez ne peut pas dépasser 64 octets (caractères), points compris. Pour ajouter un nom de domaine plus long, spécifiez-le dans le champ Nom alternatif du sujet, qui prend en charge les noms d'une longueur maximale de 253 octets.
- Si vous utilisez la AWS CLI version 1.6.3 ou une version ultérieure, utilisez le préfixe `fileb://` lorsque vous spécifiez des fichiers d'entrée codés en base64 tels que les CSR. Cela garantit que les données Autorité de certification privée AWS sont correctement analysées.

La commande OpenSSL suivante génère un CSR et une clé privée pour un certificat :

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

Vous pouvez inspecter le contenu du CSR comme suit :

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

Le résultat obtenu doit ressembler à l'exemple abrégé suivant :

```
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
```

La commande suivante crée un certificat. Aucun modèle n'étant spécifié, un certificat d'entité finale de base est émis par défaut.

```
$ 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"
```

L'ARN du certificat émis est renvoyé :

```
{
```

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

> **Note**
>
> Autorité de certification privée AWSrenvoie immédiatement un ARN avec un numéro de série lorsqu'il reçoit la issue-certificate commande. Cependant, le traitement des certificats s'effectue de manière asynchrone et peut toujours échouer. Dans ce cas, une get-certificate commande utilisant le nouvel ARN échouera également.

Émettre un certificat avec un nom de sujet personnalisé à l'aide d'un modèle APIPassThrough

Dans cet exemple, un certificat contenant des éléments de nom de sujet personnalisés est émis. En plus de fournir un CSR comme celui fourni dansDélivrance d'un certificat standard (AWS CLI), vous transmettez deux arguments supplémentaires à la issue-certificate commande : l'ARN d'un modèle APIPassThrough et un fichier de configuration JSON qui spécifie les attributs personnalisés et leurs identifiants d'objet (OID). Vous ne pouvez pas `StandardAttributes` les utiliser conjointement avec`CustomAttributes`. Toutefois, vous pouvez transmettre des OID standard dans le cadre de`CustomAttributes`. Les OID du nom de sujet par défaut sont répertoriés dans le tableau suivant (informations issues de la RFC 4519 et de la base de données de référence Global OID) :

Nom du sujet	Abréviation	ID de l'objet
countryName	c	2.5.4.6
commonName	cn	2.5.4.3
DNQualifier [qualificatif de nom distinctif]		2.5.4.46
Qualificateur de génération		2.5.4.44
givenName		2.5.4.42
initiales		2.5.4.43

Nom du sujet	Abréviation	ID de l'objet
localité	l	2.5.4.7
Nom de l'organisation	o	2.5.4.10
organizationalUnitName	ou	2.5.4.11
pseudonyme		2.5.4.65
Numéro de série		2.5.4.5
st [état]		2.5.4.8
nom de famille	sn	2.5.4.4
title		2.5.4.12
Composant de domaine	dc	0,9,2342,19200300,10.1,25
userid		0,9,2342,19200300,10.1.1

L'exemple de fichier de configuration `api_passthrough_config.txt` contient le code suivant :

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

```
}
```

Utilisez la commande suivante pour émettre le certificat :

```
$ 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
```

L'ARN du certificat émis est renvoyé :

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

Récupérez le certificat localement comme suit :

```
$ 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
```

Vous pouvez inspecter le contenu du certificat à l'aide d'OpenSSL :

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

> **Note**
>
> Il est également possible de créer une autorité de certification privée qui transmet des attributs personnalisés à chaque certificat qu'elle émet.

Émettre un certificat avec des extensions personnalisées à l'aide d'un modèle APIPassThrough

Dans cet exemple, un certificat contenant des extensions personnalisées est émis. Pour cela, vous devez transmettre trois arguments à la issue-certificate commande : l'ARN d'un modèle APIPassThrough, un fichier de configuration JSON qui spécifie les extensions personnalisées et un CSR comme celui illustré dans. Délivrance d'un certificat standard (AWS CLI)

L'exemple de fichier de configuration `api_passthrough_config.txt` contient le code suivant :

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

Le certificat personnalisé est émis comme suit :

```
$ 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
```

L'ARN du certificat émis est renvoyé :

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

Récupérez le certificat localement comme suit :

```
$ 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
```

Vous pouvez inspecter le contenu du certificat à l'aide d'OpenSSL :

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

Récupération d'un certificat privé

Vous pouvez utiliser l'API Autorité de certification privée AWS et l'AWS CLI pour émettre un certificat privé. Le cas échéant, vous pouvez utiliser l'AWS CLI ou l'API Autorité de certification privée AWS pour récupérer ce certificat. Si vous avez utilisé ACM pour créer votre autorité de certification privée et pour demander des certificats, vous devez utiliser ACM pour exporter le certificat et la clé privée chiffrée. Pour plus d'informations, consultez Exportation d'un certificat privé.

Pour récupérer un certificat d'entité finale

Utilisez la AWS CLI commande get-certificate pour récupérer un certificat d'entité finale privé. Vous pouvez également utiliser l'opération GetCertificateAPI. Nous recommandons de formater la sortie avec jq, un analyseur de type sed.

> **Note**
>
> Si vous souhaitez révoquer un certificat, vous pouvez utiliser la commande get-certificate pour récupérer le numéro de série au format hexadécimal. Vous pouvez également créer un rapport d'audit pour récupérer le numéro de série hexadécimal. Pour plus d'informations, consultez Utilisation de rapports d'audit avec votre autorité de certification privée.

```
$ 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'
```

Cette commande génère le certificat et la chaîne de certificats dans le format standard suivant.

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

Pour récupérer un certificat CA

Vous pouvez utiliser l'API Autorité de certification privée AWS et l'AWS CLI pour récupérer le certificat d'une autorité de certification privée pour votre autorité de certification privée. Exécutez la commande get-certificate-authority-certificate. Vous pouvez également appeler l'opération GetCertificateAuthorityCertificate. Nous recommandons de formater la sortie avec jq, un analyseur de type 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'
```

Cette commande produit le certificat CA dans le format standard suivant.

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

Répertorier les certificats privés

Pour répertorier vos certificats privés, générez un rapport d'audit, extrayez-le de son compartiment S3 et analysez le contenu du rapport selon les besoins. Pour plus d'informations sur la création de rapports Autorité de certification privée AWS d'audit, consultezUtilisation de rapports d'audit avec

votre autorité de certification privée. Pour plus d'informations sur la récupération d'un objet depuis un compartiment S3, consultez la section Téléchargement d'un objet dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Les exemples suivants illustrent les approches permettant de créer des rapports d'audit et de les analyser pour obtenir des données utiles. Les résultats sont formatés en JSON et les données sont filtrées à l'aide de jq, un analyseur de type sed.

1. Créez un rapport d'audit.

La commande suivante génère un rapport d'audit pour une autorité de certification spécifiée.

```
$ 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
```

En cas de succès, la commande renvoie l'ID et l'emplacement du nouveau rapport d'audit.

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

2. Récupérez et formatez un rapport d'audit.

Cette commande récupère un rapport d'audit, affiche son contenu dans une sortie standard et filtre les résultats pour n'afficher que les certificats émis le 2020-12-01 ou après cette date.

```
$ 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")'
```

Les articles retournés ressemblent à ce qui suit :

```
{
   "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. Enregistrez un rapport d'audit localement.

Si vous souhaitez effectuer plusieurs requêtes, il est pratique d'enregistrer un rapport d'audit dans un fichier local.

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

Le même filtre que précédemment produit le même résultat :

```
$ 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. Requête dans une plage de dates

Vous pouvez rechercher des certificats émis dans une plage de dates comme suit :

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

Le contenu filtré est affiché dans la sortie standard :

```
{
   "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. Recherchez des certificats selon un modèle spécifié.

La commande suivante filtre le contenu du rapport à l'aide d'un modèle d'ARN :

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

La sortie affiche les enregistrements de certificats correspondants :

```
{
   "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. Filtre pour les certificats révoqués

Pour trouver tous les certificats révoqués, utilisez la commande suivante :

```
$ cat my_local_audit_report.json | jq '.[] | select(.revokedAt != null)'
```

Un certificat révoqué s'affiche comme suit :

```
{
   "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. Filtrez à l'aide d'une expression régulière.

La commande suivante recherche les noms de sujets contenant la chaîne « leaf » :

```
$ cat my_local_audit_report.json | jq '.[] | select(.subject|test("leaf"))'
```

Les enregistrements de certificats correspondants sont renvoyés comme suit :

```
{
   "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"
}
```

Exporter un certificat privé et sa clé secrète

Autorité de certification privée AWSne peut pas exporter directement un certificat privé qu'il a signé et émis. Cependant, vous pouvez l'utiliser AWS Certificate Manager pour exporter un tel certificat avec sa clé secrète cryptée. Le certificat est alors entièrement portable pour être déployé n'importe où

dans votre PKI privée. Pour plus d'informations, consultez la section Exportation d'un certificat privé dans le guide de AWS Certificate Manager l'utilisateur.

Autre avantage : AWS Certificate Manager permet de gérer le renouvellement des certificats privés émis à l'aide de la console ACM, de l'`RequestCertificate`action de l'API ACM ou de la request-certificate commande figurant dans la section ACM du. AWS CLI Pour plus d'informations sur les renouvellements, consultez la section Renouvellement des certificats dans une PKI privée.

Révocation d'un certificat privé

Vous pouvez révoquer un Autorité de certification privée AWS certificat à l'aide de la AWS CLI commande revoke-certificate ou de l'RevokeCertificateaction API. Il peut être nécessaire de révoquer un certificat avant son expiration prévue si, par exemple, sa clé secrète est compromise ou si le domaine associé devient invalide. Pour que la révocation soit effective, le client utilisant le certificat doit pouvoir vérifier l'état de la révocation chaque fois qu'il tente d'établir une connexion réseau sécurisée.

Autorité de certification privée AWSfournit deux mécanismes entièrement gérés pour prendre en charge le contrôle de l'état de révocation : le protocole OCSP (Online Certificate Status Protocol) et les listes de révocation de certificats (CRL). Avec OCSP, le client interroge une base de données de révocation faisant autorité qui renvoie un statut en temps réel. Avec une CRL, le client compare le certificat à une liste de certificats révoqués qu'il télécharge et stocke régulièrement. Les clients refusent d'accepter les certificats révoqués.

L'OCSP et les CRL dépendent des informations de validation intégrées dans les certificats. Pour cette raison, une autorité de certification émettrice doit être configurée pour prendre en charge l'un de ces mécanismes ou les deux avant l'émission. Pour plus d'informations sur la sélection et la mise en œuvre de la révocation gérée viaAutorité de certification privée AWS, consultezConfiguration d'une méthode de révocation des certificats.

Les certificats révoqués sont toujours enregistrés dans les rapports Autorité de certification privée AWS d'audit.

> **Note**
>
> Les émetteurs de certificats multicomptes ont besoin d'autorisations supplémentaires pour révoquer les certificats qu'ils émettent ; dans le cas contraire, le propriétaire de l'autorité de certification doit procéder à la révocation. Pour permettre la révocation par les émetteurs

multicomptes, l'administrateur de l'autorité de certification doit créer deux partages de RAM, tous deux pointant vers la même autorité de certification :

1. Un partage avec l'`AWSRAMRevokeCertificateCertificateAuthority`autorisation.
2. Un partage avec l'`AWSRAMDefaultPermissionCertificateAuthority`autorisation.

Pour révoquer un certificat

Utilisez l'action RevokeCertificateAPI ou la commande revoke-certificate pour révoquer un certificat PKI privé. Le numéro de série doit être au format hexadécimal. Vous pouvez récupérer le numéro de série en appelant la commande get-certificate. La commande `revoke-certificate` ne renvoie aucune réponse.

```
$ 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"
```

Certificats révoqués et OCSP

Les réponses OCSP peuvent prendre jusqu'à 60 minutes pour refléter le nouveau statut lorsque vous révoquez un certificat. En général, l'OCSP a tendance à accélérer la distribution des informations de révocation car, contrairement aux CRL qui peuvent être mises en cache par les clients pendant des jours, les réponses OCSP ne sont généralement pas mises en cache par les clients.

Certificats révoqués dans une liste de révocation de certificats

Une liste de révocation de certificats est généralement mise à jour environ 30 minutes après la révocation d'un certificat. Si, pour une raison quelconque, une mise à jour de la CRL échoue, Autorité de certification privée AWS effectuez de nouvelles tentatives toutes les 15 minutes.

Avec Amazon CloudWatch, vous pouvez créer des alarmes pour les métriques `CRLGenerated` et`MisconfiguredCRLBucket`. Pour plus d'informations, consultez la section CloudWatchMesures prises en charge. Pour plus d'informations sur la création et la configuration des listes de révocation de certificats, consultez Planification d'une liste de révocation de certificats (CRL).

L'exemple suivant illustre un certificat révoqué dans une liste de révocation de certificats.

```
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
```

Certificats révoqués dans un rapport d'audit

Tous les certificats, y compris les certificats révoqués, sont inclus dans le rapport d'audit d'une autorité de certification privée. L'exemple suivant illustre un rapport d'audit avec un certificat émis et un certificat révoqué. Pour plus d'informations, consultez Utilisation de rapports d'audit avec votre autorité de certification privée.

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


Automatiser l'exportation d'un certificat renouvelé

Lorsque vous créez une autorité Autorité de certification privée AWS de certification, vous pouvez importer cette autorité de certification AWS Certificate Manager et laisser ACM gérer l'émission et le renouvellement des certificats. Si un certificat en cours de renouvellement est associé à un service intégré, le service applique le nouveau certificat de manière fluide. Toutefois, si le certificat a été initialement exporté pour être utilisé ailleurs dans votre environnement PKI (par exemple, dans un serveur ou une appliance sur site), vous devez l'exporter à nouveau après le renouvellement.

Pour un exemple de solution qui automatise le processus d'exportation ACM à l'aide d'Amazon et EventBridge AWS Lambda, consultez Automatisation de l'exportation de certificats renouvelés.

Comprendre les modèles de certificats

Autorité de certification privée AWSutilise des modèles de configuration pour émettre à la fois des certificats CA et des certificats d'entité finale. Lorsque vous émettez un certificat CA depuis la console PCA, le modèle de certificat CA racine ou subordonné approprié est automatiquement appliqué.

Si vous utilisez la CLI ou l'API pour émettre un certificat, vous pouvez fournir un modèle d'ARN en tant que paramètre de l'`IssueCertificate`action. Si vous ne fournissez aucun ARN, le `EndEntityCertificate/V1` modèle est appliqué par défaut. Pour plus d'informations, consultez la documentation de l'IssueCertificateAPI et de la commande issue-certificate.

> **Note**
>
> AWS Certificate Manager(ACM) disposant d'un accès partagé entre comptes à une autorité de certification privée peuvent émettre des certificats gérés signés par l'autorité de certification. Les émetteurs multicomptes sont limités par une politique basée sur les ressources et n'ont accès qu'aux modèles de certificats d'entité finale suivants :
>
> - EndEntityCertificate/V1
> - EndEntityClientAuthCertificate/V1
> - EndEntityServerAuthCertificate/V1
> - BlankEndEntityCertificate_APIPassThrough/V1
> - BlankEndEntityCertificate_APICSR PassThrough/V1
> - Certificat CA subordonné _ 0/V1 PathLen
>
> Pour plus d'informations, consultez Politiques basées sur les ressources.

Rubriques

Variétés de modèles

Autorité de certification privée AWSprend en charge quatre types de modèles.

- Modèles de base

  Modèles prédéfinis dans lesquels aucun paramètre de transmission n'est autorisé.

- Modèles CSR Passthrough

  Modèles qui étendent leurs versions de modèles de base correspondantes en autorisant le transfert CSR. Les extensions du CSR utilisé pour délivrer le certificat sont copiées sur le certificat émis. Dans les cas où le CSR contient des valeurs d'extension en conflit avec la définition du modèle, la définition du modèle aura toujours la priorité la plus élevée. Pour plus de détails sur la priorité, voirModèle d'ordre des opérations.

- Modèles APIPassthrough

  Modèles qui étendent leurs versions de modèles de base correspondantes en autorisant le transfert d'API. Les valeurs dynamiques connues de l'administrateur ou d'autres systèmes intermédiaires peuvent ne pas être connues par l'entité demandant le certificat, être impossibles à définir dans un modèle ou ne pas être disponibles dans le CSR. L'administrateur de l'autorité de certification peut toutefois récupérer des informations supplémentaires à partir d'une autre source de données, telle qu'un Active Directory, pour terminer la demande. Par exemple, si une machine ne sait pas à quelle unité organisationnelle elle appartient, l'administrateur peut rechercher les informations dans Active Directory et les ajouter à la demande de certificat en les incluant dans une structure JSON.

  Les valeurs du `ApiPassthrough` paramètre de l'`IssueCertificate`action sont copiées sur le certificat émis. Dans les cas où le `ApiPassthrough` paramètre contient des informations en conflit avec la définition du modèle, la définition du modèle aura toujours la priorité la plus élevée. Pour plus de détails sur la priorité, voirModèle d'ordre des opérations.

- Modèles APICSR Passthrough

  Des modèles qui étendent leurs versions de modèles de base correspondantes en autorisant à la fois le transfert d'API et de CSR. Les extensions du CSR utilisé pour délivrer le certificat sont copiées sur le certificat émis, et les valeurs du `ApiPassthrough` paramètre de l'`IssueCertificate`action sont également copiées. Dans les cas où la définition du modèle, les valeurs de transmission d'API et les extensions de transmission CSR présentent un conflit, la définition du modèle a la priorité la plus élevée, suivie des valeurs de transmission d'API, suivies des extensions de transmission CSR. Pour plus de détails sur la priorité, voirModèle d'ordre des opérations.

Les tableaux ci-dessous répertorient tous les types de modèles pris en charge par, Autorité de certification privée AWS avec des liens vers leurs définitions.

> **Note**
>
> Pour plus d'informations sur les modèles ARN dans GovCloud les régions, consultez AWS Private Certificate Authorityle guide de l'AWS GovCloud (US)utilisateur.

Modèles de base

Nom du modèle	ARN du modèle	Type de certificat
CodeSigningCertificate/V1	`arn:aws:acm-pca:::template/CodeSigningCertificate/V1`	Signature de code
EndEntityCertificate/V1	`arn:aws:acm-pca:::template/EndEntityCertificate/V1`	Entité finale
EndEntityClientAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate/V1`	Entité finale
EndEntityServerAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate/V1`	Entité finale
OCSP/V1 SigningCertificate	`arn:aws:acm-pca:::template/OCSPSigningCertificate/V1`	Signature OCSP
RootCACertificate/V1	`arn:aws:acm-pca:::template/RootCACertificate/V1`	CA

Nom du modèle	ARN du modèle	Type de certificat
Certificat CA subordonné _ 0/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1`	CA
Certificat CA subordonné _ 1/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1/V1`	CA
Certificat CA subordonné _ 2/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2/V1`	CA
Certificat CA subordonné _ 3/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3/V1`	CA

Modèles CSR Passthrough

Nom du modèle	ARN du modèle	Type de certificat
BlankEndEntityCertificate_CSRPassThrough/V1	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_CSRPassthrough/V1`	Entité finale
BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassThrough/V1	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassthrough/V1`	Entité finale

Nom du modèle	ARN du modèle	Type de certificat
BlankSubordinateCertificat CA_0_CSRPassThrough/V1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
BlankSubordinateCertificat CA_1_CSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
BlankSubordinateCertificat CA_2_CSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
BlankSubordinateCertificat CA_3_CSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA
CodeSigningCertificate_CSRPassThrough/V1	arn:aws:acm-pca:::template/CodeSigningCertificate_CSRPassthrough/V1	Signature de code
EndEntityCertificate_CSRPassThrough/V1	arn:aws:acm-pca:::template/EndEntityCertificate_CSRPassthrough/V1	Entité finale

Nom du modèle	ARN du modèle	Type de certificat
EndEntityClientAuthCertificate_CSRPassThrough/V1	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_CSRPassthrough/V1	Entité finale
EndEntityServerAuthCertificate_CSRPassThrough/V1	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_CSRPassthrough/V1	Entité finale
OCSP _CSRPassThrough/V1 SigningCertificate	arn:aws:acm-pca:::template/OCSPSigningCertificate_CSRPassthrough/V1	Signature OCSP
Certificat CA subordonné _0_CSRPassThrough/V1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
Certificat CA subordonné _1_CSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
Certificat CA subordonné _2_CSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA

Nom du modèle	ARN du modèle	Type de certificat
Certificat CA subordonné _ 3_CSRPassThrough/v1 PathLen	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen3_CSRPassthrough/ V1	CA

Modèles APIPassthrough

Nom du modèle	ARN du modèle	Type de certificat
BlankEndEntityCertificate_A PIPassThrough/V1	arn:aws:acm-pca::: template/BlankEndE ntityCertificate_A PIPassthrough/V1	Entité finale
BlankEndEntityCertificate_ CriticalBasicConstraints _APIPassThrough/V1	arn:aws:acm-pca::: template/BlankEndE ntityCertificate_C riticalBasicConstr aints_APIPassthrough/ V1	Entité finale
CodeSigningCertificate_APIP assThrough/V1	arn:aws:acm-pca::: template/CodeSigni ngCertificate_APIP assthrough/V1	Signature de code
EndEntityCertificate_APIPas sThrough/V1	arn:aws:acm-pca::: template/EndEntity Certificate_APIPas sthrough/V1	Entité finale
EndEntityClientAuthCertific ate_APIPassThrough/V1	arn:aws:acm-pca::: template/EndEntity	Entité finale

Nom du modèle	ARN du modèle	Type de certificat
	ClientAuthCertificate_APIPassthrough/V1	
EndEntityServerAuthCertificate_APIPassThrough/V1	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APIPassthrough/V1	Entité finale
OCSP _APIPassThrough/v1 SigningCertificate	arn:aws:acm-pca:::template/OCSPSigningCertificate_APIPassthrough/V1	Signature OCSP
RootcaCertificate_APIPassThrough/v1	arn:aws:acm-pca:::template/RootCACertificate_APIPassthrough/V1	CA
BlankRootCACertificate_APIPassThrough/v1	arn:aws:acm-pca:::template/BlankRootCACertificate_APIPassthrough/V1	CA
BlankRootCertificat CA_0_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen0_APIPassthrough/V1	CA
BlankRootCertificat CA_1_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen1_APIPassthrough/V1	CA

Nom du modèle	ARN du modèle	Type de certificat
BlankRootCertificat CA_ 2_APIPassThrough/v1 PathLen	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len2_APIPassthrough/V1	CA
BlankRootCertificat CA_ 3_APIPassThrough/v1 PathLen	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len3_APIPassthrough/V1	CA
Certificat CA subordonné _ 0_APIPassThrough/v1 PathLen	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen0_APIPassthrough/ V1	CA
BlankSubordinateCertificat CA_ 0_APIPassThrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen0_APIPas sthrough/V1	CA
Certificat CA subordonné _ 1_APIPassThrough/v1 PathLen	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen1_APIPassthrough/ V1	CA
BlankSubordinateCertificat CA_ 1_APIPassThrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen1_APIPas sthrough/V1	CA

Nom du modèle	ARN du modèle	Type de certificat
Certificat CA subordonné _ 2_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
BlankSubordinateCertificat CA_ 2_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
Certificat CA subordonné _ 3_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APIPassthrough/V1	CA
BlankSubordinateCertificat CA_ 3_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1	CA

Modèles APICSR Passthrough

Nom du modèle	ARN du modèle	Type de certificat
BlankEndEntityCertificate_APICSR PassThrough/V1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_APICSRPassthrough/V1	Entité finale

Nom du modèle	ARN du modèle	Type de certificat
BlankEndEntityCertificate_CriticalBasicConstraints_APICSRPassThrough/V1	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APICSRPassthrough/V1`	Entité finale
CodeSigningCertificate_APICSRPassThrough/V1	`arn:aws:acm-pca:::template/CodeSigningCertificate_APICSRPassthrough/V1`	Signature de code
EndEntityCertificate_APICSRPassThrough/V1	`arn:aws:acm-pca:::template/EndEntityCertificate_APICSRPassthrough/V1`	Entité finale
EndEntityClientAuthCertificate_APICSR PassThrough/V1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_APICSRPassthrough/V1`	Entité finale
EndEntityServerAuthCertificate_APICSR PassThrough/V1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APICSRPassthrough/V1`	Entité finale
OCSP _APICSRPassThrough/V1 SigningCertificate	`arn:aws:acm-pca:::template/OCSPSigningCertificate_APICSRPassthrough/V1`	Signature OCSP

Nom du modèle	ARN du modèle	Type de certificat
Certificat CA subordonné _ 0_APICSRPassThrough/V1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
BlankSubordinateCertificat CA_ 0_APICSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
Certificat CA subordonné _ 1_apicsrpassthrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA
BlankSubordinateCertificat CA_ 1_APICSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA
Certificat CA subordonné _ 2_APICSRPassThrough/ 3_APIPassThrough V1 PathLen PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APICSRPassthrough/V1	CA
BlankSubordinateCertificat CA_ 2_APICSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1	CA

Nom du modèle	ARN du modèle	Type de certificat
Certificat CA subordonné _ 3_apicsrpassthrough/v1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA
BlankSubordinateCertificat CA_ 3_APICSRPassThrough/v1 PathLen	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA

Modèle d'ordre des opérations

Les informations contenues dans un certificat émis peuvent provenir de quatre sources : la définition du modèle, le transfert d'API, le transfert CSR et la configuration de l'autorité de certification.

Les valeurs de transmission d'API ne sont respectées que lorsque vous utilisez un modèle de transmission d'API ou un modèle de transmission APICSR. La transmission CSR n'est respectée que lorsque vous utilisez un modèle de transmission CSRPassthrough ou APICSR. Lorsque ces sources d'informations sont en conflit, une règle générale s'applique généralement : pour chaque valeur d'extension, la définition du modèle a la priorité la plus élevée, suivie des valeurs intermédiaires d'API, suivies des extensions de transmission CSR.

Exemples

1. La définition du modèle pour EndEntityClientAuthCertificate_APIPassThrough définit l' ExtendedKeyUsage extension avec la valeur « authentification du serveur Web TLS, authentification du client Web TLS ». S'il ExtendedKeyUsage est défini dans le CSR ou dans le `IssueCertificate` `ApiPassthrough` paramètre, la `ApiPassthrough` valeur pour ExtendedKeyUsage sera ignorée car la définition du modèle est prioritaire, et la valeur CSR pour la ExtendedKeyUsage valeur sera ignorée car le modèle n'est pas une variété CSR passthrough.

> **Note**
>
> La définition du modèle copie néanmoins d'autres valeurs du CSR, telles que le sujet et le nom alternatif du sujet. Ces valeurs sont toujours extraites du CSR même si le modèle n'est pas une variété de CSR passthrough, car la définition du modèle a toujours la plus haute priorité.

2. La définition du modèle pour EndEntityClientAuthCertificate_apicSRPassthrough définit l'extension Subject Alternative Name (SAN) comme étant copiée depuis l'API ou le CSR. Si l'extension SAN est définie dans le CSR et fournie dans le `IssueCertificate ApiPassthrough` paramètre, la valeur de transmission d'API sera prioritaire car les valeurs de transmission d'API ont priorité sur les valeurs de transmission de la CSR.

Définitions de modèles

Les sections suivantes fournissent des informations de configuration sur les modèles de Autorité de certification privée AWS certificats pris en charge.

BlankEndEntityCertificateDéfinition de _APIPassThrough/v1

Avec des modèles de certificats d'entité finale vierges, vous pouvez émettre des certificats d'entité finale avec uniquement les contraintes X.509 Basic présentes. Il s'agit du certificat d'entité finale le plus simple qui Autorité de certification privée AWS puisse être émis, mais il peut être personnalisé à l'aide de la structure de l'API. L'extension Basic constraints définit si le certificat est un certificat CA ou non. Un modèle de certificat d'entité finale vide applique la valeur FALSE pour les contraintes de base afin de garantir qu'un certificat d'entité finale est émis et non un certificat CA.

Vous pouvez utiliser des modèles d'échange vierges pour émettre des certificats de carte à puce qui nécessitent des valeurs spécifiques pour l'utilisation des clés (KU) et pour l'utilisation étendue des clés (EKU). Par exemple, l'utilisation étendue des clés peut nécessiter l'authentification du client et l'ouverture de session par carte à puce, tandis que l'utilisation des clés peut nécessiter une signature numérique, la non-répudiation et le chiffrement des clés. Contrairement aux autres modèles intermédiaires, les modèles de certificats d'entité finale vides permettent de configurer les extensions KU et EKU, où KU peut être l'une des neuf valeurs prises en charge (DigitalSignature, NonRepudiation, KeyEncipherment, DataEncipherment, KeyAgreement, CrlSign, EncipherOnly et DecipherOnly) et EKU peut être n'importe laquelle des valeurs prises en charge (ServerAuth,

ClientAuth, codesign, EmailProtection, timestam*). ping et OCSPSigning) ainsi que des extensions personnalisées. keyCertSign

BlankEndEntityCertificate_APIPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	CA:FALSE
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankEndEntityCertificateDéfinition de _APICSRPassThrough/v1

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankEndEntityCertificate_APICSR PassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	CA:FALSE
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankEndEntityCertificate_ CriticalBasicConstraints _Définition de APICSRPassThrough/V1

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _APICSRPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critical, CA : FALSE
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA, l'API ou le CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankEndEntityCertificate_ CriticalBasicConstraints _Définition de l'APIPassThrough/V1

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _APIPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]

Paramètre X509v3	Valeur
Contraintes de base	Critical, CA : FALSE
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration ou l'API CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankEndEntityCertificate_ CriticalBasicConstraints _Définition de CSRPassThrough/V1

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _CSRPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critical, CA : FALSE
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankEndEntityCertificateDéfinition de _CSRPassThrough/V1

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankEndEntityCertificate_CSRPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	CA:FALSE
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateCACertificate_ 0_Définition de CSRPassThrough/v1 PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 0_CSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 0`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateCACertificate_ 0_APICSRPassThrough/v1 définition PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 0_APICSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 0`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateDéfinition de CACertificate_ 0_APIPassThrough/v1 PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 0_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 0`

Paramètre X509v3	Valeur
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA]

BlankSubordinateCACertificate_ 1_APIPassThrough/v1 définition PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 1_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 1`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateCACertificate_ 1_Définition de CSRPassThrough/v1 PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 1_CSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 1`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateCACertificate_ 1_APICSRPassThrough/v1 définition PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 1_APICSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 1`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateDéfinition de CACertificate_ 2_APIPassThrough/v1 PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 2_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 2`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateDéfinition de CACertificate_ 2_CSRPassThrough/v1 PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 2_CSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 2`

Paramètre X509v3	Valeur
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateDéfinition de CACertificate_ 2_APICSRPassThrough/v1 PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 2_APICSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 2`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateDéfinition de CACertificate_ 3_APIPassthrough/v1 PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 3_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 3`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateCACertificate_ 3_Définition de CSRPassThrough/v1 PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 3_CSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 3`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

BlankSubordinateCACertificate_ 3_APICSRPassThrough/v1 définition PathLen

Pour des informations générales sur les modèles vierges, consultezBlankEndEntityCertificateDéfinition de _APIPassThrough/v1.

BlankSubordinateCertificat CA_ 3_APICSRPassthrough PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 3`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

CodeSigningCertificateDéfinition /V1

Ce modèle est utilisé pour créer des certificats pour la signature de code. Vous pouvez utiliser des certificats de signature de code à partir d'Autorité de certification privée AWS avec n'importe quelle solution de signature de code basée sur une infrastructure d'une autorité de certification privée. Par exemple, les clients qui utilisent la signature de code pour AWS IoT peuvent générer un certificat de signature de code avec Autorité de certification privée AWS et l'importer dans AWS Certificate Manager. Pour plus d'informations, voir À quoi sert la signature de code AWS IoT ? et obtenez et importez un certificat de signature de code.

CodeSigningCertificate/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	`CA:FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique
Utilisation étendue des clés	Critique, signature de code
Points de distribution CRL*	[Transfert depuis la configuration CA]

*Les points de distribution d'une demande de signature de certificat sont inclus dans le modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

CodeSigningCertificateDéfinition de _APICSRPassThrough/v1

Ce modèle étend CodeSigningCertificate /V1 pour prendre en charge les valeurs de transmission d'API et de CSR.

CodeSigningCertificate_APICSR PassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	`CA:FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]

Paramètre X509v3	Valeur
Utilisation de la clé	Signature numérique critique
Utilisation étendue des clés	Critique, signature de code
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

CodeSigningCertificateDéfinition de _APIPassThrough/v1

Ce modèle est identique au `CodeSigningCertificate` modèle, à une différence près : dans ce modèle, il Autorité de certification privée AWS transmet des extensions supplémentaires via l'API au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions de l'API.

CodeSigningCertificate_APIPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	`CA:FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique
Utilisation étendue des clés	Critique, signature de code
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

CodeSigningCertificateDéfinition de _CSRPassThrough/V1

Ce modèle est identique au modèle `CodeSigningCertificate` avec une différence : dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires à partir de la demande de signature de certificat (CSR) au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions dans la demande de signature de certificat.

CodeSigningCertificate_CSRPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	`CA:FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique
Utilisation étendue des clés	Critique, signature de code
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

*Les points de distribution d'une demande de signature de certificat sont inclus dans le modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

EndEntityCertificateDéfinition /V1

Ce modèle est utilisé pour créer des certificats pour des entités finales, telles que les systèmes d'exploitation ou les serveurs web.

EndEntityCertificate/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Autorité de certification :FALSE
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du serveur web TLS, authentification du client web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA]

*Les points de distribution d'une demande de signature de certificat sont inclus dans le modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

EndEntityCertificateDéfinition de _APICSRPassThrough/v1

Ce modèle étend EndEntityCertificate /V1 pour prendre en charge les valeurs de transmission d'API et de CSR.

EndEntityCertificate_APICSR PassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Autorité de certification :FALSE

Paramètre X509v3	Valeur
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du serveur web TLS, authentification du client web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

EndEntityCertificateDéfinition de _APIPassThrough/v1

Ce modèle est identique au `EndEntityCertificate` modèle, à une différence près : dans ce modèle, il Autorité de certification privée AWS transmet des extensions supplémentaires via l'API au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions de l'API.

EndEntityCertificate_APIPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Autorité de certification :`FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés

Paramètre X509v3	Valeur
Utilisation étendue des clés	Authentification du serveur web TLS, authentification du client web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

EndEntityCertificateDéfinition de _CSRPassThrough/V1

Ce modèle est identique au modèle `EndEntityCertificate` avec une différence : dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires à partir de la demande de signature de certificat (CSR) au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions dans la demande de signature de certificat.

EndEntityCertificate_CSRPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Autorité de certification :`FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du serveur web TLS, authentification du client web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

*Les points de distribution d'une demande de signature de certificat sont inclus dans le modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

EndEntityClientAuthCertificateDéfinition /V1

Ce modèle diffère du `EndEntityCertificate` seul par la valeur d'utilisation de la clé étendue, qui le limite à l'authentification du client Web TLS.

EndEntityClientAuthCertificate/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Autorité de certification :`FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du client web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

*Les points de distribution d'une demande de signature de certificat sont inclus dans le modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

EndEntityClientAuthCertificateDéfinition de _APICSRPassThrough/v1

Ce modèle étend EndEntityClientAuthCertificate /V1 pour prendre en charge les valeurs de transmission d'API et de CSR.

EndEntityClientAuthCertificate_APICSR PassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Autorité de certification :`FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du client web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

EndEntityClientAuthCertificateDéfinition de _APIPassThrough/v1

Ce modèle est identique au modèle `EndEntityClientAuthCertificate` avec une différence. Dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires via l'API au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions de l'API.

EndEntityClientAuthCertificate_APIPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Autorité de certification :`FALSE`

Paramètre X509v3	Valeur
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du client web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

EndEntityClientAuthCertificateDéfinition de _CSRPassThrough/V1

Ce modèle est identique au modèle `EndEntityClientAuthCertificate` avec une différence. Dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires de la demande de signature de certificat (CSR) au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions dans la demande de signature de certificat.

EndEntityClientAuthCertificate_CSRPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Autorité de certification :`FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du client web TLS

Paramètre X509v3	Valeur
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

*Les points de distribution d'une demande de signature de certificat sont inclus dans le modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

EndEntityServerAuthCertificateDéfinition /V1

Ce modèle diffère du `EndEntityCertificate` seul par la valeur d'utilisation de la clé étendue, qui le limite à l'authentification du serveur Web TLS.

EndEntityServerAuthCertificate/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Autorité de certification :`FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du serveur web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA]

*Les points de distribution d'une demande de signature de certificat sont inclus dans le modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

EndEntityServerAuthCertificateDéfinition de _APICSRPassThrough/v1

Ce modèle étend EndEntityServerAuthCertificate /V1 pour prendre en charge les valeurs de transmission d'API et de CSR.

EndEntityServerAuthCertificate_APICSR PassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Autorité de certification :`FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du serveur web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

EndEntityServerAuthCertificateDéfinition de _APIPassThrough/v1

Ce modèle est identique au modèle `EndEntityServerAuthCertificate` avec une différence. Dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires via l'API au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions de l'API.

EndEntityServerAuthCertificate_APIPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]

Paramètre X509v3	Valeur
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Autorité de certification :FALSE
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du serveur web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

EndEntityServerAuthCertificateDéfinition de _CSRPassThrough/V1

Ce modèle est identique au modèle `EndEntityServerAuthCertificate` avec une différence. Dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires de la demande de signature de certificat (CSR) au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions dans la demande de signature de certificat.

EndEntityServerAuthCertificate_CSRPassThrough/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Autorité de certification :FALSE
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]

Paramètre X509v3	Valeur
Utilisation de la clé	Signature numérique critique, chiffrement des clés
Utilisation étendue des clés	Authentification du serveur web TLS
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

*Les points de distribution d'une demande de signature de certificat sont inclus dans le modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

Définition SigningCertificate OCSP/V1

Ce modèle est utilisé pour créer des certificats pour signer des réponses OCSP. Le modèle est identique au `CodeSigningCertificate` modèle, sauf que la valeur d'utilisation de la clé étendue spécifie la signature OCSP au lieu de la signature de code.

OCSP/V1 SigningCertificate

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	`CA:FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique
Utilisation étendue des clés	Critique, signature OCSP
Points de distribution CRL*	[Transfert depuis la configuration CA]

*Les points de distribution d'une demande de signature de certificat sont inclus dans le modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

Définition OCSP SigningCertificate _APICSRPassThrough/v1

Ce modèle étend l'SigningCertificateOCSP/V1 pour prendre en charge les valeurs de transmission d'API et de CSR.

OCSP _APICSRPassThrough/V1 SigningCertificate

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	`CA:FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique
Utilisation étendue des clés	Critique, signature OCSP
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

Définition OCSP SigningCertificate _APIPassThrough/v1

Ce modèle est identique au modèle `OCSPSigningCertificate` avec une différence. Dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires via l'API au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions de l'API.

OCSP _APIPassThrough/v1 SigningCertificate

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	`CA:FALSE`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique
Utilisation étendue des clés	Critique, signature OCSP
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

Définition d'OCSP SigningCertificate _CSRPassThrough/v1

Ce modèle est identique au modèle `OCSPSigningCertificate` avec une différence. Dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires de la demande de signature de certificat (CSR) au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions dans la demande de signature de certificat.

OCSP _CSRPassThrough/V1 SigningCertificate

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	`CA:FALSE`

Paramètre X509v3	Valeur
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique
Utilisation étendue des clés	Critique, signature OCSP
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

*Les points de distribution d'une demande de signature de certificat sont inclus dans le modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

Définition du certificat RootCA/v1

Ce modèle est utilisé pour émettre des certificats d'une autorité de certification racine auto-signés. Les certificats d'une autorité de certification incluent une extension de contraintes basiques critiques avec le champ d'une autorité de certification défini sur `TRUE` pour indiquer que le certificat peut être utilisé pour émettre des certificats d'une autorité de certification. Le modèle ne spécifie pas de longueur de chemin (pathLenConstraint) car cela pourrait empêcher l'expansion future de la hiérarchie. L'utilisation étendue de la clé est exclue pour empêcher l'utilisation du certificat d'une autorité de certification en tant que certificat du client ou du serveur TLS. Aucune information de liste de révocation de certificats n'est spécifiée car un certificat auto-signé ne peut pas être révoqué.

RootCACertificate/V1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`
Identifiant clé du sujet	[Dérivé de CSR]

Paramètre X509v3	Valeur
Utilisation de la clé	Signature numérique critique keyCertSign, signe CRL
Points de distribution CRL	N/A

Définition de RootcaCertificate_APIPassthrough/v1

Ce modèle étend le certificat RootCA/v1 pour prendre en charge les valeurs de transmission des API.

RootcaCertificate_APIPassThrough/v1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`
Identifiant de clé d'autorité	[Transmission depuis l'API]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique keyCertSign, signe CRL
Points de distribution CRL*	N/A

BlankRootDéfinition de CACertificate_APIPassThrough/v1

Avec les modèles de certificat racine vierges, vous pouvez émettre des certificats racine en présence uniquement des contraintes de base X.509. Il s'agit du certificat racine le plus simple qui Autorité de certification privée AWS puisse être émis, mais il peut être personnalisé à l'aide de la structure de l'API. L'extension des contraintes de base définit si le certificat est un certificat CA ou non. Un modèle de certificat racine vide applique une valeur de `TRUE` pour les contraintes de base afin de garantir l'émission d'un certificat racine de l'autorité de certification.

Vous pouvez utiliser des modèles racine intermédiaires vierges pour émettre des certificats racines qui nécessitent des valeurs spécifiques pour l'utilisation des clés (KU). Par exemple, l'utilisation des clés peut nécessiter `keyCertSign` et`cRLSign`, mais pas`digitalSignature`. Contrairement aux autres modèles de certificats intermédiaires non vierges, les modèles de certificats racines vierges permettent de configurer l'extension KU, où KU peut être n'importe laquelle des neuf valeurs prises en charge (`digitalSignature`,`nonRepudiation`,`keyEncipherment`,`dataEncipherment`,`keyAgreement`, `keyCertSign` `cRLSign`,`encipherOnly`, et`decipherOnly`).

BlankRootCACertificate_APIPassThrough/v1

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`
Identifiant clé du sujet	[Dérivé de CSR]

BlankRootDéfinition de CACertificate_ 0_APIPassThrough/v1 PathLen

Pour des informations générales sur les modèles d'autorité de certification root vierges, consultez???.

BlankRootCertificat CA_ 0_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 0`
Identifiant clé du sujet	[Dérivé de CSR]

BlankRootCACertificate_ 1_APIPassThrough/v1 définition PathLen

Pour des informations générales sur les modèles d'autorité de certification root vierges, consultez???.

BlankRootCertificat CA_ 1_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 1`
Identifiant clé du sujet	[Dérivé de CSR]

BlankRootDéfinition de CACertificate_ 2_APIPassThrough/v1 PathLen

Pour des informations générales sur les modèles d'autorité de certification root vierges, consultez???.

BlankRootCertificat CA_ 2_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 2`
Identifiant clé du sujet	[Dérivé de CSR]

BlankRootDéfinition de CACertificate_ 3_APIPassthrough/v1 PathLen

Pour des informations générales sur les modèles d'autorité de certification root vierges, consultez???.

BlankRootCertificat CA_ 3_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 3`

Paramètre X509v3	Valeur
Identifiant clé du sujet	[Dérivé de CSR]

Définition de SubordinateCACertificate_ 0/V1 PathLen

Ce modèle est utilisé pour émettre des certificats CA subordonnés avec une longueur de chemin de`0`. Les certificats d'une autorité de certification incluent une extension de contraintes basiques critiques avec le champ d'une autorité de certification défini sur `TRUE` pour indiquer que le certificat peut être utilisé pour émettre des certificats d'une autorité de certification. L'utilisation étendue de la clé n'est pas incluse, ce qui empêche l'utilisation du certificat d'une autorité de certification en tant que certificat du client ou du serveur TLS.

Pour de plus amples informations sur les chemins de certification, veuillez consulter Définition des contraintes de longueur sur le chemin d'accès de certification.

Certificat CA subordonné _ 0/V1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 0`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA]

*Les points de distribution CRL sont inclus dans les certificats émis avec ce modèle uniquement si l'autorité de certification est configurée avec la génération de CRL activée.

SubordinateCACertificate_ 0_APICSRPassThrough/v1 définition PathLen

Ce modèle étend SubordinateCAertificate_ PathLen 0/V1 pour prendre en charge les valeurs de transmission API et CSR.

Certificat CA subordonné _ 0_APICSRPassThrough/V1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 0`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

SuborordinateCACertificate_ 0_APIPassThrough/v1 définition PathLen

Ce modèle étend SubordinateCAertificate_ PathLen 0/V1 pour prendre en charge les valeurs de transmission d'API.

Certificat CA subordonné _ 0_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 0`

Paramètre X509v3	Valeur
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

SubordinateCACertificate_ 0_Définition de CSRPassThrough/v1 PathLen

Ce modèle est identique au modèle `SubordinateCACertificate_PathLen0` avec une différence : dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires à partir de la demande de signature de certificat (CSR) au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions dans la demande de signature de certificat.

> **Note**
>
> Un CSR contenant des extensions supplémentaires personnalisées doit être créé en dehors deAutorité de certification privée AWS.

Certificat CA subordonné _ 0_CSRPassThrough/V1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 0`
Identifiant de clé d'autorité	[Certificat SKI from CA]

Paramètre X509v3	Valeur
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

*Les points de distribution d'une demande de signature de certificat sont inclus dans les certificats émis avec ce modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

Définition de SubordinateCACertificate_ 1/V1 PathLen

Ce modèle est utilisé pour émettre des certificats CA subordonnés avec une longueur de chemin de`1`. Les certificats CA incluent une extension critique des contraintes de base avec le champ CA défini `TRUE` pour indiquer que le certificat peut être utilisé pour émettre des certificats CA. L'utilisation étendue de la clé n'est pas incluse, ce qui empêche l'utilisation du certificat d'une autorité de certification en tant que certificat du client ou du serveur TLS.

Pour de plus amples informations sur les chemins de certification, veuillez consulter Définition des contraintes de longueur sur le chemin d'accès de certification.

Certificat CA subordonné _ 1/V1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 1`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL

Paramètre X509v3	Valeur
Points de distribution CRL*	[Transfert depuis la configuration CA]

*Les points de distribution d'une demande de signature de certificat sont inclus dans les certificats émis avec ce modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

Définition de SubordinateCACertificate_ 1_APICSRPassThrough/v1 PathLen

Ce modèle étend SubordinateCAertificate_ PathLen 1/V1 pour prendre en charge les valeurs de transmission API et CSR.

Certificat CA subordonné _ 1_apicsrpassthrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 1`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

Définition de SubordinateCACertificate_ 1_APIPassThrough/v1 PathLen

Ce modèle étend SubordinateCAertificate_ PathLen 0/V1 pour prendre en charge les valeurs de transmission d'API.

Certificat CA subordonné _ 1_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 1`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

Définition de SubordinateCACertificate_ 1_CSRPassThrough/v1 PathLen

Ce modèle est identique au modèle `SubordinateCACertificate_PathLen1` avec une différence : dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires à partir de la demande de signature de certificat (CSR) au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions dans la demande de signature de certificat.

> **Note**
>
> Un CSR contenant des extensions supplémentaires personnalisées doit être créé en dehors deAutorité de certification privée AWS.

Certificat CA subordonné _ 1_CSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 1`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

*Les points de distribution d'une demande de signature de certificat sont inclus dans les certificats émis avec ce modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

Définition de SubordinateCACertificate_ 2/V1 PathLen

Ce modèle est utilisé pour émettre des certificats d'une autorité de certification subordonnée avec une longueur de chemin d'accès de 2. Les certificats CA incluent une extension critique des contraintes de base avec le champ CA défini `TRUE` pour indiquer que le certificat peut être utilisé pour émettre des certificats CA. L'utilisation étendue de la clé n'est pas incluse, ce qui empêche l'utilisation du certificat d'une autorité de certification en tant que certificat du client ou du serveur TLS.

Pour de plus amples informations sur les chemins de certification, veuillez consulter Définition des contraintes de longueur sur le chemin d'accès de certification.

Certificat CA subordonné _ 2/V1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]

Paramètre X509v3	Valeur
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 2`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA]

*Les points de distribution d'une demande de signature de certificat sont inclus dans les certificats émis avec ce modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

Définition de SubordinateCACertificate_ 2_APICSRPassThrough/v1 PathLen

Ce modèle étend SubordinateCAertificate_ PathLen 2/V1 pour prendre en charge les valeurs de transmission des API et CSR.

Certificat CA subordonné _ 2_APICSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 2`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

Définition de SubordinateCACertificate_ 2_APIPassThrough/v1 PathLen

Ce modèle étend SubordinateCAertificate_ PathLen 2/V1 pour prendre en charge les valeurs de transmission d'API.

Certificat CA subordonné _ 2_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 2`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

Définition de SubordinateCACertificate_ 2_CSRPassThrough/v1 PathLen

Ce modèle est identique au modèle `SubordinateCACertificate_PathLen2` avec une différence : dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires à partir de la demande de signature de certificat (CSR) au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions dans la demande de signature de certificat.

> **Note**
>
> Un CSR contenant des extensions supplémentaires personnalisées doit être créé en dehors deAutorité de certification privée AWS.

Certificat CA subordonné _ 2_CSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 2`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

*Les points de distribution d'une demande de signature de certificat sont inclus dans les certificats émis avec ce modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

Définition de SubordinateCACertificate_ 3/V1 PathLen

Ce modèle est utilisé pour émettre des certificats d'une autorité de certification subordonnée avec une longueur de chemin d'accès de 3. Les certificats CA incluent une extension critique des contraintes de base avec le champ CA défini `TRUE` pour indiquer que le certificat peut être utilisé pour émettre des certificats CA. L'utilisation étendue de la clé n'est pas incluse, ce qui empêche l'utilisation du certificat d'une autorité de certification en tant que certificat du client ou du serveur TLS.

Pour de plus amples informations sur les chemins de certification, veuillez consulter Définition des contraintes de longueur sur le chemin d'accès de certification.

Certificat CA subordonné _ 3/V1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 3`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA]

*Les points de distribution d'une demande de signature de certificat sont inclus dans les certificats émis avec ce modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

SubordinateCACertificate_ 3_APICSRPassThrough/v1 définition PathLen

Ce modèle étend SubordinateCAertificate_ PathLen 3/V1 pour prendre en charge les valeurs de transmission API et CSR.

Certificat CA subordonné _ 3_apicsrpassthrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 3`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]

Paramètre X509v3	Valeur
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

Définition de SubordinateCACertificate_ 3_APIPassThrough/v1 PathLen

Ce modèle étend SubordinateCAertificate_ PathLen 3/V1 pour prendre en charge les valeurs de transmission d'API.

Certificat CA subordonné _ 3_APIPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transfert depuis l'API ou le CSR]
Sujet	[Transfert depuis l'API ou le CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 3`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA]

* Les points de distribution CRL ne sont inclus dans le modèle que si l'autorité de certification est configurée avec la génération de CRL activée.

SubordinateCACertificate_ 3_Définition de CSRPassThrough/v1 PathLen

Ce modèle est identique au modèle `SubordinateCACertificate_PathLen3` avec une différence : dans ce modèle, Autorité de certification privée AWS transmet des extensions supplémentaires à partir de la demande de signature de certificat (CSR) au certificat si les extensions ne sont pas spécifiées dans le modèle. Les extensions spécifiées dans le modèle remplacent toujours les extensions dans la demande de signature de certificat.

> **Note**
>
> Un CSR contenant des extensions supplémentaires personnalisées doit être créé en dehors deAutorité de certification privée AWS.

Certificat CA subordonné _ 3_CSRPassThrough/v1 PathLen

Paramètre X509v3	Valeur
Nom alternatif du sujet	[Transmission depuis CSR]
Sujet	[Transmission depuis CSR]
Contraintes de base	Critique, `CA:TRUE`, `pathlen: 3`
Identifiant de clé d'autorité	[Certificat SKI from CA]
Identifiant clé du sujet	[Dérivé de CSR]
Utilisation de la clé	Signature numérique critique`keyCertSign` , signe CRL
Points de distribution CRL*	[Transfert depuis la configuration CA ou CSR]

*Les points de distribution d'une demande de signature de certificat sont inclus dans les certificats émis avec ce modèle uniquement si l'autorité de certification est configurée avec la génération de la demande de signature de certificat activée.

Utilisation de l'Autorité de certification privée AWSAPI (exemples Java)

Vous pouvez utiliser l'API AWS Private Certificate Authority pour interagir par programmation avec le service en envoyant des demandes HTTP. Le service renvoie des réponses HTTP. Pour plus d'informations, consultez la section Référence des AWS Private Certificate Authority API.

Outre l'API HTTP, vous pouvez utiliser les kits SDK AWS et les outils de ligne de commande pour interagir avec Autorité de certification privée AWS. Cette dernière méthode est recommandée plutôt que d'utiliser l'API HTTP. Pour plus d'informations, consultez Outils pour Amazon Web Services. Les rubriques suivantes vous montrent comment utiliser AWS SDK for Java pour programmer l'API Autorité de certification privée AWS.

Le GetCertificateAuthorityCsrGetCertificate, et les DescribeCertificateAuthorityAuditReportopérations soutiennent les serveurs. Vous pouvez utiliser des programmes d'attente pour contrôler la progression de votre code en fonction de la présence ou de l'état de certaines ressources. Pour plus d'informations, consultez les rubriques suivantes, ainsi que la section Waiters AWS SDK for Java dans le blog des AWS développeurs.

Rubriques

- GetPolicy
- ImportCertificateAuthorityCertificate
- IssueCertificate
- ListCertificateAuthorities
- ListPermissions
- ListTags
- PutPolicy
- RestoreCertificateAuthority
- RevokeCertificate
- TagCertificateAuthorities
- UntagCertificateAuthority
- UpdateCertificateAuthority
- Créez des CA et des certificats avec des noms de sujet personnalisés
- Créez des certificats avec des extensions personnalisées

Création et activation d'une autorité de certification racine par programmation

Cet exemple de Java montre comment activer une autorité de certification racine à l'aide de ce qui suitAutorité de certification privée AWSActions d'API :

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

```
package com.amazonaws.samples;

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

```
import com.amazonaws.samples.GetCertificateAuthorityCertificate;
import com.amazonaws.auth.AWSStaticCredentialsProvider;


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

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

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

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

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

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

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

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

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

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

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

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

        // ** Execute core code samples for Root CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
```

```
        String rootCAArn = CreateCertificateAuthority(configCA, crlConfigure, CAtype,
client);
        String csr = GetCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = IssueCertificate(rootCAArn, csr, client);
        String rootCertificate = GetCertificate(rootCertificateArn, rootCAArn, client);
        ImportCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

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

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

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

        return client;
    }

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

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

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

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

        return rootCAArn;
    }

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

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

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

```
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

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

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

        return csr;
    }

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

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

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

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

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

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

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

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

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

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

        return rootCertificateArn;
    }

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

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

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

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

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

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

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

        return rootCertificate;
    }

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

        // Create the request object and set the signed certificate, chain and CA ARN.
```

```
        ImportCertificateAuthorityCertificateRequest importRequest =
            new ImportCertificateAuthorityCertificateRequest();

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

        importRequest.setCertificateChain(null);

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

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

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

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

Création et activation d'une autorité de certification subordonnée par programmation

Cet exemple Java montre comment activer une autorité de certification subordonnée à l'aide des actions d'Autorité de certification privée AWSAPI suivantes :

- 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

L'exemple Java suivant montre comment utiliser l'CreateCerticateAuthorityopération.

L'opération crée une autorité de certification subordonnée privée. Vous devez spécifier la configuration de l'autorité de configuration, la configuration de la révocation, le type de l'autorité de certification et un jeton d'idempotence facultatif.

La configuration de l'autorité de certification spécifie les éléments suivants :

- Nom de l'algorithme et taille de la clé à utiliser pour créer la clé privée de l'autorité de certification
- Le type de l'algorithme de signature que l'autorité de certification utilise pour signer
- Les informations sur l'objet X.500

La configuration de la liste de révocation de certificats spécifie les éléments suivants :

- La période d'expiration de la liste de révocation de certificats, en jours (la période de validité de la liste de révocation de certificats)
- Le compartiment Amazon S3 qui contiendra la CRL
- Un alias CNAME pour le compartiment S3 qui est inclus dans les certificats émis par l'autorité de certification

En cas de réussite, cette fonction renvoie l'Amazon Resource Name (ARN) de l'autorité de certification.

```
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);
    }
}
```

Votre sortie doit ressembler à ce qui suit :

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

Utilisation CreateCertificateAuthority pour prendre en charge Active Directory

L'exemple Java suivant montre comment utiliser l'CreateCerticateAuthorityopération pour créer une autorité de certification pouvant être installée dans le magasin Enterprise NTAuth de Microsoft Active Directory (AD).

L'opération crée une autorité de certification racine (CA) privée à l'aide d'identifiants d'objets (OID) personnalisés. Pour plus d'informations et un AWS CLI exemple d'opération équivalente, voir Création d'une autorité de certification pour la connexion à Active Directory.

En cas de réussite, cette fonction renvoie l'Amazon Resource Name (ARN) de l'autorité de certification.

```
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);
    }
}
```

Votre sortie doit ressembler à ce qui suit :

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

CreateCertificateAuthorityAuditReport

L'exemple Java suivant montre comment utiliser l'CreateCertificateAuthorityAuditReportopération.

L'opération crée un rapport d'audit qui répertorie chaque émission ou révocation de certificat. Le rapport est enregistré dans le compartiment Amazon S3 que vous spécifiez lors de la saisie. Vous pouvez générer un nouveau rapport toutes les 30 minutes.

```
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);

    }
}
```

Votre sortie doit ressembler à ce qui suit :

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

CreatePermission

L'exemple Java suivant montre comment utiliser l'CreatePermissionopération.

L'opération attribue des autorisations d'accès d'une autorité de certification privée à un principal de service AWS désigné. Les services peuvent être autorisés à créer et récupérer des certificats à partir d'une autorité de certification privée, ainsi qu'à répertorier les autorisations actives accordées par l'autorité de certification privée. Pour renouveler automatiquement les certificats via ACM, vous devez attribuer toutes les autorisations possibles (`IssueCertificate`,`GetCertificate`, et`ListPermissions`) de l'autorité de certification au principal de service ACM (`acm.amazonaws.com`). Vous pouvez trouver l'ARN d'une autorité de certification en appelant la ListCertificateAuthoritiesfonction.

Une fois qu'une autorisation est créée, vous pouvez l'inspecter avec la ListPermissionsfonction ou la supprimer avec la DeletePermissionfonction.

```
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

L'exemple Java suivant montre comment utiliser l'DeleteCertificateAuthorityopération.

Cette opération supprime l'autorité de certification privée (CA) que vous avez créée à l'aide de l'CreateCertificateAuthorityopération. L'opération `DeleteCertificateAuthority` exige que vous fournissiez un ARN pour que l'autorité de certification soit supprimée. Vous pouvez trouver l'ARN en appelant l'ListCertificateAuthoritiesopération. Vous pouvez supprimer l'autorité de certification privée immédiatement si son état est `CREATING` ou `PENDING_CERTIFICATE`. Toutefois, si vous avez déjà importé le certificat, vous ne pouvez pas le supprimer immédiatement. Vous devez d'abord désactiver l'autorité de certification en appelant l'UpdateCertificateAuthorityopération et en définissant le `Status` paramètre sur`DISABLED`. Vous pouvez ensuite utiliser le paramètre `PermanentDeletionTimeInDays` dans l'opération `DeleteCertificateAuthority` pour spécifier le nombre de jours, de 7 à 30. Pendant cette période, l'autorité de certification privée peut être restaurée au statut `disabled`. Par défaut, si vous ne définissez pas le paramètre `PermanentDeletionTimeInDays`, la période de restauration est de 30 jours. Une fois ce délai expiré, l'autorité de certification privée est supprimée de manière permanente et ne peut pas être restaurée. Pour plus d'informations, consultez Restauration d'une autorité de certification.

Pour un exemple Java qui montre comment utiliser l'RestoreCertificateAuthorityopération, consultezRestoreCertificateAuthority.

```
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

L'exemple Java suivant montre comment utiliser l'DeletePermissionopération.

L'opération supprime les autorisations qu'une autorité de certification privée a déléguées à un principal de AWS service à l'aide de l'CreatePermissionsopération. Vous pouvez trouver l'ARN d'une autorité de certification en appelant la ListCertificateAuthoritiesfonction. Vous pouvez vérifier les autorisations accordées par une autorité de certification en appelant la ListPermissionsfonction.

```
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

L'exemple Java suivant montre comment utiliser l'DeletePolicyopération.

L'opération supprime la politique basée sur les ressources attachée à une autorité de certification privée. Une politique basée sur les ressources est utilisée pour permettre le partage de CA entre comptes. Vous pouvez trouver l'ARN d'une autorité de certification privée en appelant l'ListCertificateAuthoritiesaction.

Les actions d'API associées incluent PutPolicyet GetPolicy.

```
package com.amazonaws.samples;

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

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

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

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

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

import java.util.ArrayList;

public class CreatePermission {

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

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

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

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

```
            .build();

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

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

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

        req.setActions(permissions);

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

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

DescribeCertificateAuthority

L'exemple Java suivant montre comment utiliser l'DescribeCertificateAuthorityopération.

L'opération répertorie les informations relatives à votre autorité de certification privée. Vous devez spécifier l'Amazon Resource Name (ARN) de l'autorité de certification privée. La sortie contient l'état de votre autorité de certification. Il peut s'agir de l'un des états suivants :

- `CREATING`— Autorité de certification privée AWS est en train de créer votre autorité de certification privée.
- `PENDING_CERTIFICATE`— Le certificat est en attente. Vous devez utiliser l'autorité de certification sur site subordonnée ou racine pour signer votre demande de signature de certificat, puis l'importer dans PCA.
- `ACTIVE`— Votre CA privée est active.
- `DISABLED`— Votre CA privée a été désactivée.
- `EXPIRED`— Votre certificat CA privé a expiré.
- `FAILED`— Votre autorité de certification privée ne peut pas être créée.
- `DELETED`— Votre CA privée est en cours de restauration, après quoi elle sera définitivement supprimée.

```
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

L'exemple Java suivant montre comment utiliser l'DescribeCertificateAuthorityAuditReportopération.

L'opération répertorie les informations relatives à un rapport d'audit spécifique que vous avez créé en appelant l'CreateCertificateAuthorityAuditReportopération. Les informations d'audit sont créées chaque fois que la clé privée de l'autorité de certification est utilisée. La clé privée est utilisée lorsque vous émettez un certificat, signez une liste de révocation de certificats ou révoquez un certificat.

```
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);
   }
}
```

Votre sortie doit ressembler à ce qui suit :

```
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

L'exemple Java suivant montre comment utiliser l'GetCertificateopération.

L'opération récupère un certificat à partir de votre autorité de certification privée. L'ARN du certificat est renvoyé lorsque vous appelez l'IssueCertificateopération. Vous devez spécifier à la fois l'ARN de votre autorité de certification privée et l'ARN du certificat émis lors de l'appel de l'opération `GetCertificate`. Vous pouvez récupérer le certificat si son état est `ISSUED`. Vous pouvez appeler l'CreateCertificateAuthorityAuditReportopération pour créer un rapport contenant des informations sur tous les certificats émis et révoqués par votre autorité de certification privée.

```
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);
    }
}
```

Votre sortie doit être une chaîne de certificats similaire à la chaîne suivante pour l'autorité de certification et le certificat que vous avez spécifiés.

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

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

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

GetCertificateAuthorityCertificate

L'exemple Java suivant montre comment utiliser l'GetCertificateAuthorityCertificateopération.

Cette opération récupère le certificat et la chaîne de certificats pour votre autorité de certification privée. Le certificat et la chaîne sont tous deux des chaînes codées en base64 au format PEM. La

chaîne n'inclut pas le certificat de l'autorité de certification. Chaque certificat de la chaîne signe le certificat précédent.

```
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);
   }
}
```

Votre sortie doit être un certificat et une chaîne similaires aux éléments suivants pour l'autorité de certification que vous avez spécifiée.

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

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

GetCertificateAuthorityCsr

L'exemple Java suivant montre comment utiliser l'GetCertificateAuthorityCsropération.

Cette opération récupère la demande de signature de certificat (CSR) pour votre autorité de certification privée. Le CSR est créé lorsque vous appelez l'CreateCertificateAuthorityopération. Récupérez la demande de signature de certificat de votre infrastructure X.509 sur site et signez-la à l'aide de votre autorité de certification racine ou d'une autorité de certification subordonnée. Réimportez ensuite le certificat signé dans ACM PCA en appelant l'ImportCertificateAuthorityCertificateopération. Le CSR est renvoyé sous forme de chaîne codée en base64 au format 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);
   }
}
```

Votre sortie doit être similaire à la sortie suivante pour l'autorité de certification que vous spécifiez. La demande de signature de certificat (CSR) est codée en base64 au format PEM. Enregistrez-la dans un fichier local, transférez-la dans votre infrastructure X.509 sur site et signez-la à l'aide de votre autorité de certification racine ou d'une autorité de certification subordonnée.

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

GetPolicy

L'exemple Java suivant montre comment utiliser l'GetPolicyopération.

L'opération récupère la politique basée sur les ressources attachée à une autorité de certification privée. Une politique basée sur les ressources est utilisée pour permettre le partage d'autorités de certification entre comptes. Vous pouvez trouver l'ARN d'une autorité de certification privée en appelant l'ListCertificateAuthoritiesaction.

Les actions d'API associées incluent PutPolicyet DeletePolicy.

```
package com.amazonaws.samples;

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

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

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

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

public class GetPolicy {

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

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

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

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

```
            .build();

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

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

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

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

ImportCertificateAuthorityCertificate

L'exemple Java suivant montre comment utiliser l'ImportCertificateAuthorityCertificateopération.

Cette opération importe votre certificat d'une autorité de certification privée signé dans Autorité de certification privée AWS. Avant de pouvoir appeler cette opération, vous devez créer l'autorité de certification privée en appelant l'CreateCertificateAuthorityopération. Vous devez ensuite générer une demande de signature de certificat (CSR) en appelant l'GetCertificateAuthorityCsropération. Récupérez la demande de signature de certificat de votre autorité de certification sur site, et utilisez un certificat racine ou un certificat subordonné pour la signer. Créez une chaîne de certificats, et copiez le certificat signé et la chaîne de certificats dans votre répertoire de travail.

```
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

L'exemple Java suivant montre comment utiliser l'IssueCertificateopération.

Cette opération utilise votre autorité de certification privée (CA) pour émettre un certificat d'entité finale. Cette opération renvoie l'Amazon Resource Name (ARN) du certificat. Vous pouvez récupérer le certificat en appelant le GetCertificateet en spécifiant l'ARN.

> **Note**
>
> L'IssueCertificateopération nécessite que vous spécifiiez un modèle de certificat. Cet exemple utilise le `EndEntityCertificate/V1` modèle. Pour plus d'informations sur tous les modèles disponibles, consultezComprendre les modèles de certificats.

```
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);
   }
}
```

Votre sortie doit ressembler à ce qui suit :

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

ListCertificateAuthorities

L'exemple Java suivant montre comment utiliserListCertificateAuthoritiesUsage operation.

Cette opération répertorie les autorités de certification privées (CA) que vous avez créées à l'aide deCreateCertificateAuthorityUsage operation.

```
package com.amazonaws.samples;

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

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

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

public class ListCertificateAuthorities {

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

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

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

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

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

      // Retrieve a list of your CAs.
      ListCertificateAuthoritiesResult result= null;
      try {
```

```
            result = client.listCertificateAuthorities(req);
        } catch (InvalidNextTokenException ex) {
            throw ex;
        }

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

Si des autorités de certification doivent être répertoriées, votre sortie doit ressembler à la sortie suivante :

```
[{
 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

L'exemple Java suivant montre comment utiliser l'ListPermissionsopération.

Cette opération répertorie les autorisations, le cas échéant, attribuées par votre autorité de certification privée. Les autorisations, y compris `IssueCertificate``GetCertificate`, et`ListPermissions`, peuvent être attribuées à un directeur de AWS service lors de l'CreatePermissionopération et révoquées lors de l'DeletePermissionsopération.

```
package com.amazonaws.samples;

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

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

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

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

public class ListPermissions {

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

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
```

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

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

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

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

      // List the tags.
      ListPermissionsResult result = null;
      try {
         result = client.listPermissions(req);
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch(RequestFailedException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      }

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

```
}
```

Si l'autorité de certification privée désignée a attribué des autorisations à un mandataire de service, votre sortie doit être similaire à ce qui suit :

```
[{
       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

L'exemple Java suivant montre comment utiliser l'ListTagsopération.

Cette opération répertorie les balises, le cas échéant, qui sont associées à votre autorité de certification privée. Les balises sont des étiquettes que vous pouvez utiliser pour identifier et organiser vos autorités de certification. Chaque balise est constituée d'une clé et d'une valeur facultative. Appelez l'TagCertificateAuthorityopération pour ajouter une ou plusieurs balises à votre autorité de certification. Appelez l'UntagCertificateAuthorityopération pour supprimer les balises.

```
package com.amazonaws.samples;

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

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

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

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

public class ListTags {

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

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

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

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

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

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

```
            throw ex;
        }

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

Si des balises doivent être répertoriées, votre sortie doit ressembler à la sortie suivante :

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

PutPolicy

L'exemple Java suivant montre comment utiliser l'PutPolicyopération.

L'opération associe une politique basée sur les ressources à une autorité de certification privée, permettant le partage entre comptes. Lorsqu'il est autorisé par une politique, un mandant résidant sur un autre AWS compte peut émettre et renouveler des certificats d'entité finale privés en utilisant une autorité de certification privée dont il n'est pas le propriétaire. Vous pouvez trouver l'ARN d'une autorité de certification privée en appelant l'ListCertificateAuthoritiesaction. Pour des exemples de politiques, consultez le Autorité de certification privée AWS guide sur les politiques basées sur les ressources.

Une fois qu'une politique est attachée à une autorité de certification, vous pouvez l'inspecter avec l'GetPolicyaction ou la supprimer avec l'DeletePolicyaction.

```
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

L'exemple Java suivant montre comment utiliser l'RestoreCertificateAuthorityopération. Une autorité de certification privée peut être restaurée à tout moment au cours de sa période de restauration. Actuellement, cette période peut durer de 7 à 30 jours à compter de la date de suppression et peut

être définie lorsque vous supprimez l'autorité de certification. Pour plus d'informations, consultez Restauration d'une autorité de certification. Voir aussi l'exemple 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

L'exemple Java suivant montre comment utiliser l'RevokeCertificateopération.

Cette opération révoque un certificat que vous avez émis en appelant l'IssueCertificateopération. Si vous avez activé une liste de révocation de certificats (CRL) lors de la création ou de la mise à jour de votre autorité de certification privée, les informations relatives aux certificats révoqués sont incluses dans la CRL. Autorité de certification privée AWSécrit la CRL dans un compartiment Amazon S3 que vous spécifiez. Pour plus d'informations, consultez la CrlConfigurationstructure.

```
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

L'exemple Java suivant montre comment utiliser l'TagCertificateAuthorityopération.

Cette opération ajoute une ou plusieurs balises à votre autorité de certification privée. Les balises sont des étiquettes que vous pouvez utiliser pour identifier et organiser vos ressources AWS. Chaque balise est constituée d'une clé et d'une valeur facultative. Lorsque vous appelez cette opération, vous spécifiez l'autorité de certification privée par son Amazon Resource Name (ARN). Vous spécifiez la balise à l'aide d'une paire clé-valeur. Pour identifier une caractéristique spécifique de cette autorité de certification, vous pouvez appliquer une balise à une seule autorité de certification privée. Ou,

pour filtrer une relation commune entre ces autorités de certification, vous pouvez appliquer la même balise à plusieurs autorités de certification privées. Pour supprimer une ou plusieurs balises, utilisez l'UntagCertificateAuthorityopération. Appelez l'ListTagsopération pour voir quels tags sont associés à votre autorité de certification.

```
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

L'exemple Java suivant montre comment utiliser l'UntagCertificateAuthorityopération.

Cette opération supprime une ou plusieurs balises de votre autorité de certification privée. Une balise est constituée d'une paire clé-valeur. Si vous ne spécifiez pas la valeur de la balise lorsque vous appelez cette opération, la balise sera supprimée quelle que soit la valeur. Si vous spécifiez une valeur, la balise est supprimée uniquement si elle est associée à la valeur spécifiée. Pour ajouter des balises à une autorité de certification privée, utilisez l'TagCertificateAuthorityopération. Appelez l'ListTagsopération pour voir quels tags sont associés à votre autorité de certification.

```
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

L'exemple Java suivant montre comment utiliser l'UpdateCertificateAuthorityopération.

L'opération met à jour l'état ou la configuration d'une autorité de certification privée. Votre autorité de certification privée doit être à l'état `ACTIVE` ou `DISABLED` pour que vous puissiez la mettre à jour. Vous pouvez désactiver une autorité de certification privée qui se trouve à l'état `ACTIVE` ou réactiver une autorité de certification à l'état `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;
        }
    }
}
```

Créez des CA et des certificats avec des noms de sujet personnalisés

Dans laCustomAttributeL'objet permet aux administrateurs de transmettre des identifiants d'objets (OID) personnalisés à des autorités de certification et à des certificats privés. Les OID personnalisés peuvent être utilisés pour créer des hiérarchies de noms de sujets spécialisées qui reflètent la structure et les besoins de votre organisation. Les certificats personnalisés doivent être créés à l'aide de l'un des`ApiPassthrough`modèles. Pour plus d'informations sur les modèles, consultez Variétés de modèles. Pour plus d'informations sur l'utilisation d'attributs personnalisés, consultezÉmission de certificats d'entité finale privésetProcédure de création d'une autorité de certification (CLI) .

Vous ne pouvez pas utiliser`StandardAttributes`en conjonction avec`CustomAttributes`. Cependant, vous pouvez transmettre des OID standard dans le cadre d'un`CustomAttributes`. Les OID des noms d'objet par défaut répertoriés dans le tableau suivant :

Nom du sujet	ID de l'objet
Pays	2.5.4.6
CommonName	2.5.4.3
DistinguishedNameQualifier	2.5.4.46
GenerationQualifier	2.5.4.4
GivenName	2.5.4.42
Initials	2.5.4.43
Locality	2.5.4.7
Organisation	2.5.4.10
OrganizationalUnit	2.5.4.11
Pseudonym	2.5.4.65
SerialNumber	2.5.4.5
État	2.5.4.8
Surname	2.5.4.4
Title	2.5.4.12

Rubriques

Créez une CA avec 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);
    }
}
```

Délivrer un certificat avec 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);
    }
}
```

Créez des certificats avec des extensions personnalisées

Dans laCustomExtensionCet objet permet aux administrateurs de définir des extensions X.509 personnalisées dans des certificats privés. Les certificats personnalisés doivent être créés à l'aide de l'un des ApiPassthrough modèles. Pour plus d'informations sur les modèles, consultez Variétés de modèles. Pour de plus amples informations sur l'utilisation d'extensions personnalisées, veuillez consulterÉmission de certificats d'entité finale privés.

Rubriques

Activez une autorité de certification subordonnée avec NameConstraints extension

```
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);
    }
}
```

Délivrer un certificat avec l'extension de déclaration 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);
   }
}
```

Utilisation de l'Autorité de certification privée AWSAPI pour implémenter le standard Matter (exemples Java)

Vous pouvez utiliser l'AWS Private Certificate AuthorityAPI pour créer des certificats conformes à la norme de connectivité Matter. Matter spécifie des configurations de certificats qui améliorent la sécurité et la cohérence des appareils de l'Internet des objets (IoT) sur plusieurs plateformes d'ingénierie. Pour plus d'informations sur Matter, consultez buildwithmatter.com.

Les exemples Java présentés dans cette section interagissent avec le service en envoyant des requêtes HTTP. Le service renvoie des réponses HTTP. Pour plus d'informations, consultez la section Référence des AWS Private Certificate Authority API.

Outre l'API HTTP, vous pouvez utiliser les kits SDK AWS et les outils de ligne de commande pour interagir avec Autorité de certification privée AWS. Cette dernière méthode est recommandée plutôt que d'utiliser l'API HTTP. Pour plus d'informations, consultez Outils pour Amazon Web Services. Les rubriques suivantes vous montrent comment utiliser AWS SDK for Java pour programmer l'API Autorité de certification privée AWS.

Le GetCertificateAuthorityCsrGetCertificate, et les DescribeCertificateAuthorityAuditReportopérations soutiennent les serveurs. Vous pouvez utiliser des programmes d'attente pour contrôler la progression de votre code en fonction de la présence ou de l'état de certaines ressources. Pour plus d'informations, consultez les rubriques suivantes, ainsi que la section Waiters AWS SDK for Java dans le blog des AWS développeurs.

Matter 1.2, publié en octobre 2023, prend en charge la révocation du DAC à l'aide de listes de révocation de certificats (CRL). Pour vous aider à vous conformer à la norme Matter actuelle, lorsque vous activez la révocation de la CRL pour les autorités de certification qui émettent des certificats Matter, dans l'`CrlConfiguration`objet, dans la `CrlDistributionPointExtensionConfiguration` structure, définis sur`OmitExtension`. `true`

Généralement, les autorités de certification intègrent le point de distribution CRL (CDP) dans les certificats qu'elles émettent afin que les parties utilisatrices chargées de la validation de la chaîne de certificats puissent récupérer la CRL et vérifier l'état du certificat. Dans Matter, l'URI du CDP n'est pas écrit dans les certificats. Au lieu de cela, les utilisateurs récupèrent les CDP depuis le Matter Distributed Compliance Ledger (DCL), le magasin de données fiable de Matter. Vous devez télécharger l'URI CDP dans le Matter DCL afin qu'il puisse être découvert lors de la validation des

DAC. Pour plus d'informations sur la détermination de l'URI CDP, consultezDéterminer l'URI du point de distribution CRL (CDP) . Pour plus d'informations sur Matter, consultez la documentation Matter DCL.

Rubriques

- Activer une autorité d'attestation de produit (PAA)
- Activer un intermédiaire d'attestation de produit (PAI)
- Création d'un certificat d'attestation d'appareil (DAC)
- Activez une autorité de certification racine pour les certificats opérationnels des nœuds (NOC).
- Activer une autorité de certification subordonnée pour les certificats opérationnels des nœuds (NOC)
- Création d'un certificat opérationnel de nœud (NOC)

Activer une autorité d'attestation de produit (PAA)

Cet exemple Java montre comment utiliser le Définition de RootcaCertificate_APIPassthrough/v1 modèle pour créer et installer un certificat Matter Root CA (PAA) pour l'attestation du produit. L'extension AuthorityKeyIdentifier (AKI) est facultative pour le PaaS. Pour définir un AKI, vous devez générer une valeur AKI codée en Base64 et la transmettre via un. CustomExtension

L'exemple appelle les actions Autorité de certification privée AWS d'API suivantes :

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Si vous rencontrez des problèmes, consultez Utilisation de la norme Matter la section Dépannage.

```
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);
    }
}
```

Activer un intermédiaire d'attestation de produit (PAI)

Cet exemple Java montre comment utiliser le BlankSubordinateDéfinition de CACertificate_0_APIPassThrough/v1 PathLen modèle pour créer et installer un certificat Matter Suordinate CA (PAI) pour l'attestation du produit. Vous devez générer une KeyUsage valeur codée en Base64 et la transmettre via un. CustomExtension

L'exemple appelle les actions Autorité de certification privée AWS d'API suivantes :

- CreateCertificateAuthority
- GetCertificateAuthorityCsr

- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate
- GetCertificateAuthorityCertificate

Si vous rencontrez des problèmes, consultez Utilisation de la norme Matter la section Dépannage.

```
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);
    }
}
```

Création d'un certificat d'attestation d'appareil (DAC)

Cet exemple Java montre comment utiliser le modèle BlankEndEntityCertificate_ CriticalBasicConstraints _APIPassthrough/v1 pour créer un certificat d'attestation Matter Device. Vous devez générer une KeyUsage valeur codée en Base64 et la transmettre via un. CustomExtension

L'exemple appelle l'action d'Autorité de certification privée AWSAPI suivante :

- IssueCertificate

Si vous rencontrez des problèmes, consultez Utilisation de la norme Matter la section Dépannage.

```
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);
   }
}
```

Activez une autorité de certification racine pour les certificats opérationnels des nœuds (NOC).

Cet exemple Java montre comment utiliser le Définition de RootcaCertificate_APIPassthrough/v1 modèle pour créer et installer un certificat Matter Root CA afin de délivrer des NOC. L'extension AuthorityKeyIdentifier (AKI) est facultative pour les certificats NOC Root CA. Pour définir un AKI, vous devez générer une valeur AKI codée en Base64 et la transmettre via un. CustomExtension

L'exemple appelle les actions Autorité de certification privée AWS d'API suivantes :

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Si vous rencontrez des problèmes, consultez Utilisation de la norme Matter la section Dépannage.

```
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);
    }
}
```

Activer une autorité de certification subordonnée pour les certificats opérationnels des nœuds (NOC)

Cet exemple Java montre comment utiliser le BlankSubordinateDéfinition de CACertificate_ 0_APIPassThrough/v1 PathLen modèle pour émettre et installer un certificat Matter Suordinate CA afin de délivrer des NOC. Vous devez générer une KeyUsage valeur codée en Base64 et la transmettre via un. CustomExtension

L'exemple appelle les actions Autorité de certification privée AWS d'API suivantes :

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

En cas de problème, consultez Utilisation de la norme Matter la section Dépannage.

```
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);
    }
}
```

Création d'un certificat opérationnel de nœud (NOC)

Cet exemple Java montre comment utiliser le modèle BlankEndEntityCertificate_ CriticalBasicConstraints _APIPassthrough/v1 pour créer un certificat opérationnel Matter Node. Vous devez générer une KeyUsage valeur codée en Base64 et la transmettre via un. CustomExtension

L'exemple appelle l'action d'Autorité de certification privée AWSAPI suivante :

- IssueCertificate

Si vous rencontrez des problèmes, consultez Utilisation de la norme Matter la section Dépannage.

```
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);
   }
}
```

Utilisation de l'Autorité de certification privée AWSAPI pour implémenter la norme relative au permis de conduire mobile (MdL) (exemples Java)

Vous pouvez utiliser l'AWS Private Certificate AuthorityAPI pour créer des certificats conformes à la norme ISO/IEC pour le permis de conduire mobile (MdL). Cette norme établit les spécifications d'interface pour la mise en œuvre d'un permis de conduire en association avec un appareil mobile, y compris les configurations des certificats.

Les exemples Java présentés dans cette section interagissent avec le service en envoyant des requêtes HTTP. Le service renvoie des réponses HTTP. Pour plus d'informations, consultez la page Référence de l'API AWS Private Certificate Authority.

Outre l'API HTTP, vous pouvez également utiliser les AWS SDK et les AWS CLI outils de gestionAutorité de certification privée AWS. Nous vous recommandons d'utiliser le SDK ou AWS CLI l'API HTTP. Pour plus d'informations, consultez Outils pour Amazon Web Services. Les rubriques suivantes vous montrent comment utiliser AWS SDK for Java pour programmer l'API Autorité de certification privée AWS.

Le `GetCertificateAuthorityCsrGetCertificate`, et les `DescribeCertificateAuthorityAuditReport`opérations soutiennent les serveurs. Vous pouvez utiliser des programmes d'attente pour contrôler la progression de votre code en fonction de la présence ou de l'état de certaines ressources. Pour plus d'informations, consultez les rubriques suivantes et Waiters in the AWS SDK for Java sur AWSIe blog des développeurs.

Rubriques

- Activer un certificat d'autorité de certification de l'autorité émettrice (IACA)
- Création d'un certificat de signataire de document

Activer un certificat d'autorité de certification de l'autorité émettrice (IACA)

Cet exemple Java montre comment utiliser le BlankRootDéfinition de CACertificate_ 0_APIPassThrough/v1 PathLen modèle pour créer et installer un certificat d'autorité de certification (IACA) conforme à la norme ISO/IEC MdL. Vous devez générer des valeurs codées en base64

pour `KeyUsageIssuerAlternativeName`, et`CRLDistributionPoint`, et les transmettre. `CustomExtensions`

> **Note**
>
> Le certificat de lien IACA établit un chemin de confiance entre l'ancien certificat racine IACA et le nouveau certificat racine IACA. L'autorité émettrice peut générer et distribuer un certificat de lien IACA pendant le processus de changement de clé IACA. Vous ne pouvez pas émettre de certificat de lien IACA en utilisant un certificat racine IACA avec `pathLen=0` set.

L'exemple appelle les actions Autorité de certification privée AWS d'API suivantes :

- 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);
    }
}
```

Création d'un certificat de signataire de document

Cet exemple Java montre comment utiliser le modèle BlankEndEntityCertificate_APIPassthrough/v1 pour créer un certificat de signataire de document conforme à la norme ISO/IEC MdL. Vous devez générer des valeurs codées en base64 pour `KeyUsageIssuerAlternativeName`, et `CRLDistributionPoint` et les transmettre. `CustomExtensions`

L'exemple appelle l'action d'Autorité de certification privée AWSAPI suivante :

- 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);
    }
}
```

Certificats CA privés signés en externe

Si la racine d'approbation de votre hiérarchie d'autorité de certification privée doit être une autorité de certification en dehors d'une Autorité de certification privée AWS, vous pouvez créer et auto-signer votre propre autorité de certification racine. Vous pouvez également obtenir un certificat d'une autorité de certification privée signé par une autorité de certification privée externe gérée par votre organisation. Quelle que soit sa source, vous pouvez utiliser cette autorité de certification obtenue de l'extérieur pour signer un certificat d'autorité de certification subordonnée privé qui Autorité de certification privée AWS gère.

> **Note**
>
> Les procédures de création ou d'obtention d'un fournisseur de services de confiance externe ne relèvent pas du champ d'application de ce guide.

L'utilisation d'une autorité de certification parent externe vous Autorité de certification privée AWS permet d'appliquer les contraintes de nom de l'autorité de certification telles que définies dans la section Contraintes de nom de la RFC 5280. Les contraintes de nom permettent aux administrateurs de l'autorité de certification de restreindre les noms de sujets dans les certificats.

Si vous envisagez de signer un certificat d'autorité de certification subordonnée privée auprès d'une autorité de certification externe, vous devez effectuer trois tâches avant de disposer d'une autorité de certification fonctionnelle : Autorité de certification privée AWS

1. Générez une demande de signature de certificat (CSR).
2. Soumettez le CSR à votre autorité de signature externe et retournez-le avec un certificat signé et une chaîne de certificats.
3. Installez un certificat signé dansAutorité de certification privée AWS.

Les procédures suivantes décrivent comment effectuer ces tâches à l'aide du AWS Management Console ou duAWS CLI.

Pour obtenir et installer un certificat CA signé en externe (console)

1. (Facultatif) Si vous n'êtes pas encore sur la page de détails de l'autorité de certification, ouvrez la Autorité de certification privée AWS console à l'adresse https://console.aws.amazon.com/acm-

pca/home. Sur la page Autorités de certification privées, choisissez une autorité de certification subordonnée dont le statut est En attente de certificat, Actif, Désactivé ou Expiré.

2. Choisissez Actions, Installer le certificat CA pour ouvrir la page Installer le certificat CA subordonné.
3. Sur la page Installer le certificat d'autorité de certification subordonnée, sous Sélectionner le type d'autorité de certification, choisissez Autorité de certification privée externe.
4. Sous CSR pour cette autorité de certification, la console affiche le texte ASCII codé en Base64 du CSR. Vous pouvez copier le texte à l'aide du bouton Copier ou vous pouvez choisir Exporter le CSR vers un fichier et l'enregistrer localement.

   > **Note**
   >
   > Le format exact du texte CSR doit être préservé lors du copier-coller.

5. Si vous ne pouvez pas effectuer immédiatement les étapes hors ligne pour obtenir un certificat signé auprès de votre autorité de signature externe, vous pouvez fermer la page et y revenir une fois que vous possédez un certificat signé et une chaîne de certificats.

   Sinon, si vous êtes prêt, effectuez l'une des opérations suivantes :

   - Collez le texte ASCII codé en Base64 de votre corps de certificat et de votre chaîne de certificats dans leurs zones de texte respectives.
   - Choisissez Upload pour charger le corps du certificat et la chaîne de certificats à partir des fichiers locaux dans leurs zones de texte respectives.
6. Choisissez Confirmer et installez.

Pour obtenir et installer un certificat CA (CLI) signé en externe

1. Utilisez la get-certificate-authority-csrcommande pour récupérer la demande de signature de certificat (CSR) pour votre autorité de certification privée. Si vous souhaitez envoyer la demande de création de certificat à votre affichage, utilisez l'option `--output text` pour éliminer les caractères CR/LF à la fin de chaque ligne. Pour envoyer la demande de création de certificat vers un fichier, utilisez l'option de redirection (**>**) suivie d'un nom de fichier.

   ```
   $ 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
```

Après avoir enregistré un CSR en tant que fichier local, vous pouvez l'inspecter à l'aide de la commande OpenSSL suivante :

```
openssl req -in path_to_CSR_file -text -noout
```

Cette commande génère une sortie similaire à la sortie suivante. Notez que l'extension de l'autorité de certification est TRUE, ce qui indique que la demande de signature de certificat concerne un certificat d'une autorité de certification.

```
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. Soumettez le CSR à votre autorité de signature externe et obtenez des fichiers contenant le certificat signé codé en Base64 PEM et la chaîne de certificats.
3. Utilisez la import-certificate-authority-certificatecommande pour importer le fichier de certificat CA privé et le fichier de chaîne dansAutorité de certification privée AWS.

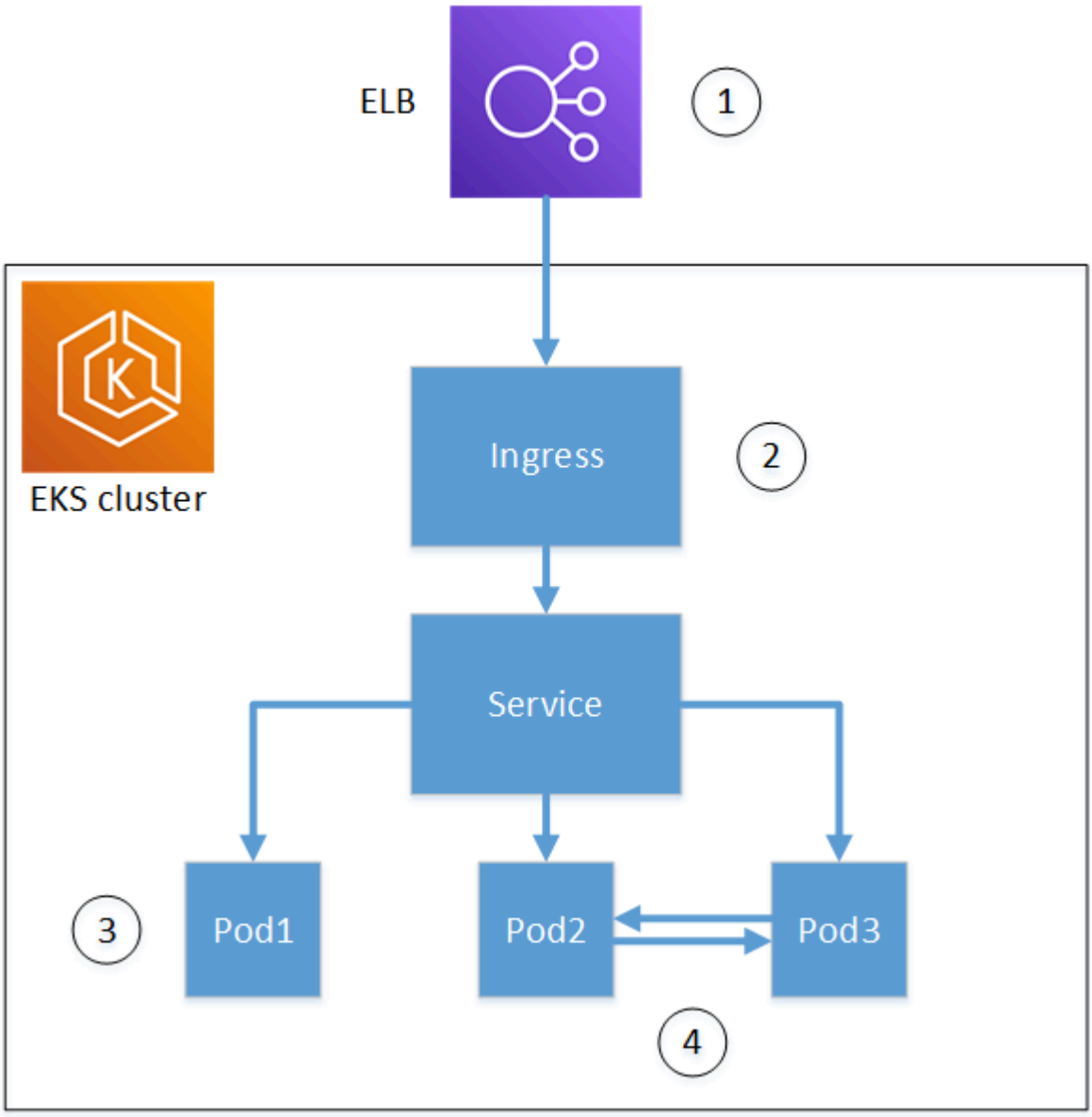
```
$ aws acm-pca import-certificate-authority-certificate \
--certificate-authority-arn arn:aws:acm-pca:region:account:\
certificate-authority/12345678-1234-1234-1234-123456789012 \
--certificate file://C:\example_ca_cert.pem \
--certificate-chain file://C:\example_ca_cert_chain.pem
```

Sécuriser Kubernetes avec Autorité de certification privée AWS

Les conteneurs et applications Kubernetes utilisent des certificats numériques pour fournir une authentification et un chiffrement sécurisés via le protocole TLS. Une solution largement adoptée pour la gestion du cycle de vie des certificats TLS dans Kubernetes est cert-manager, un module complémentaire de Kubernetes qui demande des certificats, les distribue aux conteneurs Kubernetes et automatise le renouvellement des certificats.

Autorité de certification privée AWSfournit un plug-in open source à cert-manager aws-privateca-issuer, pour les utilisateurs de cert-manager qui souhaitent configurer une autorité de certification sans stocker de clés privées dans le cluster. Les utilisateurs soumis à des exigences réglementaires en matière de contrôle d'accès et d'audit des opérations de l'autorité de certification peuvent utiliser cette solution pour améliorer l'auditabilité et garantir la conformité. Vous pouvez utiliser le plugin AWS Private CA Issuer avec Amazon Elastic Kubernetes Service (Amazon EKS), un Kubernetes AWS autogéré sur ou dans un Kubernetes sur site.

Le schéma ci-dessous montre certaines des options disponibles pour utiliser le protocole TLS dans un cluster Amazon EKS. Cet exemple de cluster, contenant diverses ressources, est situé derrière un équilibreur de charge. Les numéros identifient les points de terminaison possibles pour les communications sécurisées par TLS, notamment l'équilibreur de charge externe, le contrôleur d'entrée, un module individuel au sein d'un service et une paire de modules communiquant de manière sécurisée entre eux.



1. Terminaison au niveau de l'équilibreur de charge.

   Elastic Load Balancing (ELB) est un service AWS Certificate Manager intégré, ce qui signifie que vous pouvez approvisionner ACM auprès d'une autorité de certification privée, signer un certificat avec celle-ci et l'installer à l'aide de la console ELB. Cette solution fournit une communication cryptée entre un client distant et l'équilibreur de charge. Les données sont transmises non chiffrées au cluster EKS. Vous pouvez également fournir un certificat privé à un équilibreur autre que l'équilibreur de AWS charge pour mettre fin au protocole TLS.

2. Résiliation au niveau du contrôleur d'entrée Kubernetes.

Le contrôleur d'entrée réside dans le cluster EKS en tant qu'équilibreur de charge et routeur natifs. Si vous avez installé à la fois cert-manager et aws-privateca-issuerprovisionné le cluster avec une autorité de certification privée, Kubernetes peut installer un certificat TLS signé sur le contrôleur, lui permettant ainsi de servir de point de terminaison du cluster pour les communications externes. Les communications entre l'équilibreur de charge et le contrôleur d'entrée sont cryptées, et après l'entrée, les données sont transmises en clair aux ressources du cluster.

3. Terminaison dans un pod.

   Chaque pod est un groupe d'un ou de plusieurs conteneurs qui partagent des ressources de stockage et de réseau. Si vous avez installé à la fois cert-manager et aws-privateca-issuerprovisionné le cluster avec une autorité de certification privée, Kubernetes peut installer des certificats TLS signés sur les pods selon les besoins. Une connexion TLS se terminant par un pod n'est pas disponible par défaut pour les autres pods du cluster.

4. Communications sécurisées entre les pods.

   Vous pouvez également doter plusieurs pods de certificats leur permettant de communiquer entre eux. Les scénarios possibles sont les suivants :

   - Provisionnement à l'aide de certificats auto-signés générés par Kubernetes. Cela permet de sécuriser les communications entre les pods, mais les certificats auto-signés ne répondent pas aux exigences HIPAA ou FIPS.
   - Provisionnement à l'aide de certificats signés par une autorité de certification privée. Comme dans les numéros 2 et 3 ci-dessus, cela nécessite d'installer à la fois cert-manager et aws-privateca-issuerde provisionner le cluster avec une autorité de certification privée. Kubernetes peut ensuite installer des certificats TLS signés sur les pods selon les besoins.

Utilisation du gestionnaire de certificats entre comptes

Les administrateurs disposant d'un accès multicompte à une autorité de certification peuvent utiliser cert-manager pour provisionner un cluster Kubernetes. Pour plus d'informations, consultez Bonnes pratiques de sécurité pour l'accès entre comptes aux autorités de certification privées.

> **Note**
>
> Seuls certains modèles de Autorité de certification privée AWS certificats peuvent être utilisés dans des scénarios entre comptes. Consultez the section called "Modèles de certificats pris en charge " la liste des modèles disponibles.

Modèles de certificats pris en charge

Le tableau suivant répertorie les Autorité de certification privée AWS modèles qui peuvent être utilisés avec cert-manager pour provisionner un cluster Kubernetes.

Modèles pris en charge pour Kubernetes	Support pour l'utilisation entre comptes
BlankEndEntityCertificateDéfinition de _CSRPassThrough/V1	
CodeSigningCertificateDéfinition /V1	
EndEntityCertificateDéfinition /V1	✓
EndEntityClientAuthCertificateDéfinition /V1	✓
EndEntityServerAuthCertificateDéfinition /V1	✓
Définition SigningCertificate OCSP/V1	

Exemples de solutions

Les solutions d'intégration suivantes montrent comment configurer l'accès Autorité de certification privée AWS à un cluster Amazon EKS.

- Clusters Kubernetes compatibles TLS avec Amazon EKS Autorité de certification privée AWS
- Configuration du chiffrement end-to-end TLS sur Amazon EKS avec le nouveau AWS Load Balancer Controller

Autorité de certification privée AWS Connector for Active Directory

Qu'est-ce que AWS Private CA Connector for Active Directory

AWS Private CApeut émettre et gérer les certificats requis parAWS Managed Microsoft AD. À l'aide du Autorité de certification privée AWS connecteur pour Active Directory (connecteur pour AD), vous pouvez remplacer une autorité de certification d'entreprise locale ou tierce par une autorité de certification privée gérée dont vous êtes le propriétaire, en fournissant l'inscription de certificats aux utilisateurs, aux groupes et aux machines gérés par votre AD.

Vous pouvez utiliser le Connector for AD AWS Managed Microsoft AD pour éliminer l'infrastructure sur site en faisant migrer votre infrastructure AD et votre infrastructure à clé publique vers le cloud. Pour les clients qui souhaitent l'utiliser AWS Private CA avec leur AD sur site, cette fonctionnalité s'intègre également à AWS Managed Microsoft AD Connector.

Rubriques

Êtes-vous un utilisateur de Connector for AD pour la première fois ?

Si vous utilisez Connector for AD pour la première fois, nous vous recommandons de commencer par lire les sections suivantes :

- Qu'est-ce que c'est Autorité de certification privée AWS ?
- Présentation de AWS Directory Service

Accès au connecteur pour AD

Vous pouvez accéder à Connector for AD via la console et AWS CLI les API. Vous pouvez accéder au connecteur dans la console depuis la AWS Private CA console, depuis votre AWS Directory Service console ou en recherchant Connector for AD dans la barre AWS Management Console de recherche.

Tarification de Connector for AD

Connector for AD est proposé en tant que fonctionnalité sans frais supplémentaires. Autorité de certification privée AWS Vous ne payez que pour les autorités de certification privées et les certificats que vous émettez par leur intermédiaire.

Pour obtenir les dernières informations sur Autorité de certification privée AWS les prix, consultez la section AWS Private Certificate AuthorityTarification. Vous pouvez également utiliser le calculateur de AWS prix pour estimer les coûts.

Commencer à utiliser AWS Private CA Connector pour Active Directory

Avec AWS Private CA Connector for Active Directory, vous pouvez délivrer des certificats de votre autorité de certification privée à vos objets Active Directory à des fins d'authentification et de chiffrement. Lorsque vous créez un connecteur, vous AWS Private Certificate Authority créez un point de terminaison dans votre VPC pour que les objets de votre répertoire puissent demander des certificats.

Pour émettre des certificats, vous devez créer un connecteur et des modèles compatibles AD pour le connecteur. Lorsque vous créez un modèle, vous pouvez définir les autorisations d'inscription pour vos groupes AD.

Rubriques

- Connecteur pour les prérequis AD
- Créez un connecteur
- Configuration des politiques AD
- Création d'un modèle
- Gérer les autorisations des groupes AD

Connecteur pour les prérequis AD

Les éléments suivants sont requis pour Connector for AD.

Pour créer un connecteur pour AD, vous devez configurer une AWS Private Certificate Authority (CA) et un annuaire. Vous devez ensuite partager l'autorité de certification privée et le répertoire avec le

service Connector for AD. Vous devez également définir correctement les groupes de sécurité et les politiques IAM pour créer un connecteur.

Autorité de certification privée AWS

Configurez un Autorité de certification privée AWS pour l'émission de certificats pour les objets de votre répertoire. Pour plus d'informations, consultez Administration privée de CA.

Ils Autorité de certification privée AWS doivent être dans l'`Active`état requis pour créer un connecteur pour AD. Le nom de sujet de l'autorité de certification privée doit inclure un nom commun. La création d'un connecteur échouera si vous essayez de créer un connecteur à l'aide d'une autorité de certification privée sans nom commun.

Active Directory

En plus d'unAutorité de certification privée AWS, vous avez besoin d'un Active Directory dans un cloud privé virtuel (VPC). Connector for AD prend en charge les types de répertoires suivants proposés par AWS Directory Service :

- AWSMicrosoft Active Directory géré : AWS Directory Service vous pouvez exécuter Microsoft Active Directory (AD) en tant que service géré. AWS Directory Service for Microsoft Active Directoryégalement appeléAWS Managed Microsoft AD, est alimenté par Windows Server 2019. AvecAWS Managed Microsoft AD, vous pouvez exécuter des charges de travail sensibles aux annuaires dans le, AWS Cloud notamment Microsoft Sharepoint et des applications personnalisées basées sur .Net et SQL Server.
- Connecteur Active Directory : AD Connector est une passerelle d'annuaire qui peut rediriger les demandes d'annuaire vers votre Microsoft Active Directory local, sans mettre en cache aucune information dans le cloud. AD Connector prend en charge la connexion à un domaine hébergé sur Amazon EC2

> **Note**
>
> L'inscription de contrôleurs de domaine n'est pas prise en charge lors de l'utilisation du Connector for AD avecAWS Managed Microsoft AD.

Compte de service

Lorsque vous utilisez le Directory Service AD Connector, vous devez déléguer des autorisations supplémentaires au compte de service. Définissez une liste de contrôle d'accès (ACL) sur le compte de service pour permettre de :

- Ajouter et supprimer un nom principal de service (SPN) à lui-même
- Créez et mettez à jour les autorités de certification dans les conteneurs suivants :

```
#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
```

- Créez et mettez à jour un objet d'autorité de AuthCertificates certification NT (CA). Remarque : si l'objet NT AuthCertificates CA existe, vous devez lui déléguer des autorisations. Si l'objet n'existe pas, vous devez déléguer la possibilité de créer des objets enfants dans le conteneur Public Key Services.

```
#objects
CN=NTAuthCertificates,CN=Public Key Services,CN=Services,CN=Configuration
```

> **Note**
>
> Si vous en AWS Managed Microsoft AD utilisez, les autorisations supplémentaires seront déléguées automatiquement lorsque vous autoriserez le service Connector for AD à accéder à votre annuaire. Vous pouvez ignorer cette étape préalable.

Vous pouvez utiliser ce PowerShell script pour déléguer les autorisations supplémentaires. Cela créera l'objet d'autorité AuthCertifiates de certification NT. Remplacez « myconnectoraccount » par le nom du compte de service.

```
$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"
```

Stratégie IAM

Pour créer un connecteur pour AD, vous avez besoin d'une politique IAM qui vous permet de créer des ressources de connecteur, de partager votre autorité de certification privée avec le service Connector for AD et d'autoriser le service Connector for AD avec votre annuaire.

Voici un exemple de politique gérée par l'utilisateur :

```
{
    "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/*"
        }
    ]
}
```

Connector for AD nécessite des AWS RAM autorisations supplémentaires, à la fois pour une utilisation en console et en ligne de commande.

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

```
    ]
}
```

Partager Autorité de certification privée AWS avec Connector for AD

Vous devrez partager votre service AWS Private CA avec les connecteurs en utilisant le partage principal du AWS Resource Access Manager service.

Lorsque vous créez un connecteur dans la AWS console, le partage de ressources est automatiquement créé pour vous.

Lorsque vous créez un partage de ressources à l'aide deAWS CLI, vous utiliserez la AWS RAM create-resource-share commande.

La commande suivante crée un partage de ressources :

```
$  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
```

Le principal de service qui appelle CreateConnector dispose des autorisations d'émission de certificats sur le PCA. Pour empêcher les responsables de service qui utilisent Connector for AD d'avoir un accès général à vos Autorité de certification privée AWS ressources, limitez leurs autorisations à l'aide `CalledVia` de.

Autorisez Connector for AD avec votre annuaire

Vous autorisez le service Connector for AD avec votre annuaire afin que le connecteur puisse communiquer avec votre annuaire. Pour autoriser le service Connector for AD, vous devez créer un enregistrement d'annuaire. Pour plus d'informations sur la création d'un enregistrement dans un annuaire, voir Gestion des inscriptions dans les annuaires

Groupes de sécurité

La communication entre votre VPC et le connecteur Connector for AD est directeAWS PrivateLink, ce qui nécessite un ou plusieurs groupes de sécurité dotés de règles entrantes qui ouvrent les ports

443 TCP et UDP sur votre VPC. Ce groupe de sécurité vous sera demandé lorsque vous créerez un connecteur. Vous pouvez spécifier la source comme personnalisée et sélectionner le bloc CIDR de votre VPC. Vous pouvez choisir de restreindre davantage cette option (adresse IP, CIDR et ID de groupe de sécurité).

Créez un connecteur

Pour obtenir des instructions, reportez-vous à la procédure Création d'un connecteur

Configuration des politiques AD

Connector for AD n'est pas en mesure de visualiser ou de gérer la configuration de l'objet de politique de groupe (GPO) du client. Le GPO contrôle le routage des demandes AD vers le serveur du client Autorité de certification privée AWS ou vers d'autres serveurs d'authentification ou de distribution de certificats. Une configuration GPO non valide peut entraîner un routage incorrect de vos demandes. Il appartient aux clients de configurer et de tester la configuration du Connector for AD.

Les politiques de groupe sont associées à un connecteur, et vous pouvez choisir de créer plusieurs connecteurs pour un même AD. C'est à vous de gérer le contrôle d'accès à chaque connecteur si ses configurations de politique de groupe sont différentes.

La sécurité des appels du plan de données dépend de Kerberos et de la configuration de votre VPC. Toute personne ayant accès au VPC peut passer des appels sur le plan de données à condition d'être authentifiée auprès de l'AD correspondant. Cela existe en dehors des limites AWSAuth et la gestion des autorisations et de l'authentification dépend de vous, le client.

Dans Active Directory, suivez les étapes ci-dessous pour créer un GPO pointant vers l'URI généré lors de la création d'un connecteur. Cette étape est requise pour utiliser Connector for AD à partir de la console ou de la ligne de commande.

Configurez les GPO.

1. Ouvrez le gestionnaire de serveur sur le DC
2. Accédez à Outils et choisissez Gestion des politiques de groupe dans le coin supérieur droit de la console.
3. Accédez à Forêt > Domaines. Sélectionnez votre nom de domaine et cliquez avec le bouton droit de la souris sur votre domaine. Sélectionnez Créer un GPO dans ce domaine, liez-le ici... et entrez `PCA GP0` le nom.

4. Le GPO nouvellement créé sera désormais répertorié sous votre nom de domaine.
5. Choisissez PCA GPO, puis sélectionnez Modifier. Si une boîte de dialogue s'ouvre avec le message d'alerte « Ceci est un lien » et que les modifications seront propagées dans le monde entier, accusez réception du message pour continuer. L'éditeur de gestion des politiques de groupe devrait s'ouvrir.
6. Dans l'éditeur de gestion des stratégies de groupe, accédez à Configuration ordinateur > Stratégies > Paramètres Windows > Paramètres de sécurité > Politiques de clé publique (choisissez le dossier).
7. Accédez au type d'objet et choisissez Certificate Services Client - Certificate Enrollment Policy
8. Dans les options, remplacez le modèle de configuration par Activé.
9. Vérifiez que la politique d'inscription Active Directory est cochée et activée. Choisissez Ajouter.
10. La fenêtre du serveur de politique d'inscription aux certificats devrait s'ouvrir.
11. Entrez le point de terminaison du serveur de politique d'inscription des certificats qui a été généré lorsque vous avez créé votre connecteur dans le champ Entrez l'URI de la politique du serveur d'inscription.
12. Laissez le type d'authentification Windows intégré.
13. Choisissez Valider. Une fois la validation réussie, sélectionnez Ajouter. La boîte de dialogue se ferme.
14. Revenez à Certificate Services Client - Politique d'inscription des certificats et cochez la case à côté du connecteur nouvellement créé pour vous assurer que le connecteur est la politique d'inscription par défaut
15. Choisissez la politique d'inscription Active Directory, puis sélectionnez Supprimer.
16. Dans la boîte de dialogue de confirmation, choisissez Oui pour supprimer l'authentification basée sur LDAP.
17. Choisissez Appliquer et OK dans la fenêtre Client des services de certificats > Politique d'inscription des certificats et fermez-la.
18. Accédez au dossier Public Key Policies et choisissez Certificate Services Client - Auto-Enrollment.
19. Modifiez l'option Modèle de configuration sur Activé.
20. Vérifiez que les cases Renouveler les certificats expirés et Mettre à jour les certificats sont toutes deux cochées. Laissez les autres paramètres tels quels.
21. Choisissez Appliquer, puis OK, puis fermez la boîte de dialogue.

Configurez ensuite les politiques de clé publique pour la configuration utilisateur. Accédez à Configuration utilisateur > Politiques > Paramètres Windows > Paramètres de sécurité > Politiques relatives aux clés publiques. Suivez les procédures décrites aux étapes 6 à 21 pour configurer les politiques de clé publique pour la configuration utilisateur.

Une fois que vous avez terminé de configurer les GPO et les politiques de clé publique, les objets du domaine demanderont des certificats à Autorité de certification privée AWS Connector for AD et recevront des certificats émis parAutorité de certification privée AWS.

Création d'un modèle

Pour obtenir des instructions, reportez-vous à la procédure Création d'un modèle de connecteur

Gérer les autorisations des groupes AD

Pour obtenir des instructions, reportez-vous à la procédure Gestion des groupes AD et des autorisations pour les modèles

Autorité de certification privée AWSConnecteur pour les procédures Active Directory

Les procédures décrites dans cette section décrivent comment créer des connecteurs, configurer des modèles Autorité de certification privée AWS et intégrer Active Directory. Vous pouvez effectuer ces opérations à partir de la console Autorité de certification privée AWS Connector for AD, en utilisant la section Connector for AD de l'AWS CLIAPI Connector for AD ou en utilisant l'API Autorité de certification privée AWS Connector for AD.

> **Note**
>
> Bien que Autorité de certification privée AWS Connector for AD soit étroitement intégréAutorité de certification privée AWS, les deux services disposent d'API distinctes. Pour plus d'informations, consultez la référence AWS Private Certificate Authority d'API et la référence d'API du Autorité de certification privée AWS connecteur pour Active Directory.

Procédures

- Création d'un connecteur
- Création d'un modèle de connecteur

- Répertorier les connecteurs pour Active Directory
- Modèles de connecteurs de liste
- Afficher les détails du connecteur
- Afficher les détails du modèle de connecteur
- Gestion des inscriptions dans les annuaires
- Gestion des groupes AD et des autorisations pour les modèles
- Configuration du nom principal du service
- Connecteur de balisage pour les ressources AD

Création d'un connecteur

Utilisez les procédures suivantes pour créer un connecteur à l'aide de la console, de la ligne de commande ou de l'API AWS Private CA du Connector for Active Directory.

Création d'un connecteur (console)

Suivez les procédures ci-dessous pour créer et configurer un connecteur à l'aide de la AWS console.

Tâches

- Console ouverte
- Ouvrez le connecteur Create
- Choisissez ou créez un répertoire
- Choisissez une autorité de certification privée
- Configuration du balisage
- Vérifier et créer

Console ouverte

Connectez-vous à votre AWS compte et ouvrez la console AWS Private CA Connector for Active Directory à l'adresse**https://console.aws.amazon.com/pca-connector-ad/home**.

Ouvrez le connecteur Create

Sur la page d'accueil du premier service ou sur la page Connecteurs pour Active Directory, choisissez Create connector.

Choisissez ou créez un répertoire

Sur la page Créer un connecteur CA privé pour Active Directory, fournissez des informations dans la section Active Directory.

- Sous Sélectionnez votre type Active Directory, choisissez l'un des deux types disponibles :
  - AWS Directory Service for Microsoft Active Directory— Spécifie un Active Directory géré parAWS Directory Service.
  - Active Directory sur site avec AWS AD Connector : utilise AD Connector pour accéder à un Active Directory que vous hébergez sur site.
- Sous Sélectionnez votre répertoire, choisissez votre répertoire dans la liste.

  Vous pouvez également choisir Créer un répertoire, qui ouvre la AWS Directory Service console dans une nouvelle fenêtre. Lorsque vous avez fini de créer un nouveau répertoire, revenez à la console AWS Private CA Connector for Active Directory et actualisez la liste des répertoires. Votre nouveau répertoire devrait être disponible pour la sélection.

  > **Note**
  >
  > Lorsque vous créez un répertoire, notez que Connector for AD ne prend en charge que les types de répertoires suivants proposés dans la AWS Directory Service console :
  >
  > - AWS Managed Microsoft AD
  > - Connecteur AD

- Sous Sélectionner les groupes de sécurité pour le point de terminaison VPC, choisissez un groupe de sécurité dans la liste.

  Vous pouvez également choisir Créer un groupe de sécurité, ce qui ouvre la console Amazon EC2 sur la page Créer un groupe de sécurité dans une nouvelle fenêtre. Lorsque vous avez fini de créer un groupe de sécurité, revenez à la console AWS Private CA Connector for Active Directory et actualisez la liste des groupes de sécurité. Votre nouveau groupe de sécurité doit être disponible pour la sélection.

Choisissez une autorité de certification privée

Dans la section Autorité de certification privée, choisissez une autorité de certification privée dans la liste.

Vous pouvez également choisir Create Private CA, qui ouvre la Autorité de certification privée AWS console sur la page des autorités de certification privées dans une nouvelle fenêtre. Lorsque vous avez fini de créer une autorité de certification, retournez à la console AWS Private CA Connector for Active Directory et actualisez la liste des autorités de certification. Votre nouvelle autorité de certification devrait être disponible pour la sélection.

Configuration du balisage

Dans le volet Tags — facultatif, vous pouvez appliquer et supprimer des métadonnées sur votre ressource AD. Les balises sont des paires de chaînes clé-valeur dans lesquelles la clé doit être unique à la ressource et la valeur est facultative. Le volet affiche toutes les balises existantes pour la ressource dans un tableau. Les actions suivantes sont prises en charge.

- Choisissez Gérer les balises pour ouvrir la page Gérer les balises.
- Choisissez Ajouter un nouveau tag pour créer un tag. Renseignez le champ Clé et, éventuellement, le champ Valeur. Choisissez Enregistrer les modifications pour appliquer le tag.
- Cliquez sur le bouton Supprimer situé à côté d'une étiquette pour la marquer pour suppression, puis sélectionnez Enregistrer les modifications pour confirmer.

Vérifier et créer

Après avoir fourni les informations requises et passé en revue vos choix, choisissez Create connector. Cela ouvre la page de détails des connecteurs pour Active Directory où vous pouvez voir la progression de votre connecteur au fur et à mesure de sa création.

Une fois le processus de création d'un connecteur terminé, attribuez-lui un nom principal de service.

Création d'un connecteur pour Active Directory (AWS CLI)

Pour créer un connecteur pour Active Directory à l'aide de la CLI, utilisez la commande create-connector dans la section AWS Private CA Connecteur pour Active Directory du. AWS CLI

Création d'un connecteur pour Active Directory (API)

Pour créer un connecteur pour Active Directory avec l'API, utilisez l' CreateConnectoraction de l'API AWS Private CA Connector for Active Directory.

Création d'un modèle de connecteur

Création d'un modèle de connecteur (console)

Suivez les procédures suivantes pour créer et configurer un modèle de connecteur à l'aide duAWSconsole.

Tâches

- Console ouverte
- Choisissez le connecteur
- Rechercher une section de modèles
- Méthode de création du modèle
- Paramètres du modèle
- Configuration des paramètres des certificats
- Configuration du traitement des demandes et des paramètres d'inscription
- Configuration des extensions d'utilisation des clés
- Attribuer des politiques d'application
- Configurer des politiques d'application personnalisées
- Configuration des paramètres de cryptographie
- Configuration des groupes et des autorisations
- Configurer les modèles de remplacement
- Configuration du balisage
- Vérifier et créer

Console ouverte

Connectez-vous à votreAWSet ouvrez leAWS Private CAConnecteur pour console Active Directory à l'adresse**`https://console.aws.amazon.com/pca-connector-ad/home`**.

Choisissez le connecteur

Choisissez un connecteur parmiConnecteurs pour Active Directorylistez puis choisissezAfficher les détails.

Rechercher une section de modèles

Sur la page de détails du connecteur, recherchez leModèlessection, puis choisissezCréer un modèle.

Méthode de création du modèle

Sur leCréer un modèlepage, dans leMéthode de création du modèlesection, choisissez l'une des options de méthode.

- Commencez à partir d'un modèle prédéfini(par défaut) — Choisissez parmi une liste de modèles prédéfinis pour les applications AD :
  - Signature de code
  - Ordinateur
  - Authentification du contrôleur de domaine
  - Agent de restauration EFS
  - Agent d'inscription
  - Agent d'inscription (ordinateur)
  - IPSec
  - Authentification Kerberos
  - Serveur RAS et IAS
  - Connexion par carte à puce
  - Signature d'une liste de confiance
  - Signature de l'utilisateur
  - Authentification du poste
- Commencez à partir d'un modèle existant que vous avez créé— Choisissez parmi une liste de modèles personnalisés que vous avez créés précédemment.
- Commencez à partir d'un modèle vierge— Choisissez cette option pour commencer à créer un tout nouveau modèle.

Paramètres du modèle

Dans leParamètres du modèlesection, fournissez les informations suivantes :

- Nom du modèle— Le nom du modèle

- Version du schéma du modèle— La version du schéma du modèle. La version du schéma affecte la disponibilité des options du modèle comme suit :

  Schéma version 2

  - Prend en charge la compatibilité client de Windows XP/Windows Server 2003 et versions ultérieures.
  - Compatible uniquement avec les fournisseurs de services cryptographiques traditionnels.

  Schéma version 3

  - Prend en charge la compatibilité client de Windows Vista/Windows Server 2008 et versions ultérieures.
  - Permet au demandeur de renouveler avec la clé existante.
  - Supporte uniquement les principaux fournisseurs de stockage.

  Schéma version 4

  - Prend en charge la compatibilité client de Windows 8/Windows Server 2012 et versions ultérieures.
  - Permet au demandeur de renouveler avec la clé existante.
  - Supporte les anciens fournisseurs de services cryptographiques et les fournisseurs de stockage de clés.
- Compatibilité avec les clients— Le niveau de système d'exploitation minimum compatible avec le modèle. Choisissez l'une des options répertoriées :
  - Windows XP/Windows Server 2003
  - Windows Vista/Windows Server 2008
  - Windows 7/Windows Server 2008 R2
  - Windows 8 et versions ultérieures/Windows Server 2012
  - Windows 8 et versions ultérieures/Windows Server 2012 R2
  - Windows 8 et versions ultérieures/Windows Server 2016 et versions ultérieures

Configuration des paramètres des certificats

Dans leParamètres du certificatsection, définissez les paramètres suivants pour les certificats basés sur ce modèle.

- Type de certificat— Spécifiez s'il faut créerUtilisateurouOrdinateurcertificats.

- Inscription automatique— Choisissez d'activer ou non l'inscription automatique pour les certificats basés sur ce modèle.
- Période de validité— Spécifiez la période de validité du certificat sous la forme d'une valeur entière d'heures, de jours, de semaines, de mois ou d'années. La valeur minimale est de 2 heures.
- Période de renouvellement— Spécifiez une période de renouvellement de certificat sous la forme d'une valeur entière d'heures, de jours, de semaines, de mois ou d'années. La période de renouvellement ne doit pas dépasser 75 % de la période de validité.
- Nom du sujet— Choisissez une ou plusieurs options à inclure dans le nom du sujet en fonction des informations contenues dans Active Directory.

  > **Note**
  >
  > Au moins un nom de sujet ou une autre option de nom de sujet doit être spécifié.

  - Nom commun
  - DNS en tant que nom commun
  - Chemin du répertoire
  - E-mail
- Nom alternatif du sujet— Choisissez une ou plusieurs options à inclure dans le nom alternatif du sujet en fonction des informations contenues dans Active Directory.

  > **Note**
  >
  > Au moins un nom de sujet ou une autre option de nom de sujet doit être spécifié.

  - GUID du répertoire
  - nom DNS
  - DNS de domaine
  - E-mail
  - Nom principal du service (SPN)
  - Nom d'utilisateur principal (UPN)

Configuration du traitement des demandes et des paramètres d'inscription

Dans leGestion des demandes de certificats et options d'inscriptionsection, spécifiez l'objectif des certificats en fonction du modèle, en choisissant l'une des options suivantes.

- Signature
- Chiffrement
- Signature et chiffrement
- Signature et connexion par carte à puce

Choisissez ensuite laquelle des fonctionnalités suivantes vous souhaitez activer. Les options varient en fonction de l'objectif du certificat.

- Supprimer les certificats non valides (ne pas archiver)
- Inclure des algorithmes symétriques
- Clé privée exportable

Enfin, choisissez une option d'inscription au certificat. Les options varient en fonction de l'objectif du certificat.

- Aucune saisie utilisateur requise
- Inviter l'utilisateur lors de l'inscription
- Inviter l'utilisateur lors de l'inscription et lui demander de saisir des informations

Configuration des extensions d'utilisation des clés

Dans leParamètres d'extension d'utilisation des cléssection, choisissez l'option d'utilisation de la signature et de la clé de chiffrement.

Utilisation de la clé de signature

- Signature numérique
- La signature est une preuve d'origine (non répudiation)

Utilisation des clés de chiffrement

- Autoriser l'échange de clés sans chiffrement des clés (accord relatif aux clés)

- Autoriser l'échange de clés uniquement avec le chiffrement des clés (chiffrement des clés)
- Autoriser le chiffrement des données utilisateur (chiffrement des données)

Vous pouvez également choisirRendre les extensions d'utilisation essentiellespour les deux types de clés.

Attribuer des politiques d'application

Dans lePolitiques relatives aux applicationssection, choisissez toutes les politiques d'application applicables. Les politiques disponibles sont répertoriées sur plusieurs pages. Certaines politiques peuvent être présélectionnées en raison des paramètres précédents.

Configurer des politiques d'application personnalisées

Dans lePolitiques d'application personnaliséessection, vous pouvez ajouter des OID personnalisés au modèle et spécifier si les extensions de politique d'application sont critiques.

Configuration des paramètres de cryptographie

Dans leParamètres de cryptographiesection, choisissez les catégories suivantes de paramètres de cryptographie pour les certificats basés sur ce modèle.

1. Le contenu en haut de la section est déterminé parMéthode de création du modèle etParamètres du modèle que vous avez choisi précédemment.

   - Si vous avez accepté la valeur par défautVersion 2 du modèledansParamètres du modèle , puis les messages d'état suivants s'affichent ici :
     - Catégorie de fournisseur de cryptographie
     - Fournisseur de services cryptographiques traditionnel

     Dans ce cas, il n'y a aucun paramètre à configurer et vous pouvez passer à l'étape suivante.
   - Si vous avez spécifiéVersion du modèle 3dansParamètres du modèle , puis les messages d'état suivants s'affichent ici :
     - Catégorie de fournisseur de cryptographie
     - Fournisseur de stockage de clés

     Vous devez également choisir unAlgorithme cléà partir des options répertoriéesECDH_P256,ECDH_P384,ECDH_P521, etRSA.

- Si vous avez spécifiéVersion du modèle 4dansParamètres du modèle , vous devez alors choisir entreFournisseur de stockage de cléset unFournisseur de services cryptographiques traditionnel. Si tu choisisFourniture de rangement pour clés, puis unAlgorithme clédoit également être choisi parmi les options répertoriéesECDH_P256,ECDH_P384,ECDH_P521, etRSA.

2. Taille de clé minimale (bits)— Spécifiez la taille de clé minimale. Ce paramètre aura une incidence sur les fournisseurs de cryptographie disponibles.
3. Choisissez les fournisseurs de chiffrement qui peuvent être utilisés pour les demandes— Choisissez l'une des deux options disponibles :

   - Les demandes peuvent utiliser n'importe quel fournisseur disponible sur l'ordinateur du sujet
   - Les demandes doivent utiliser l'un des fournisseurs sélectionnés suivants

     Le choix de cette option ouvre unFournisseur de cryptographieliste. Vous pouvez sélectionner et hiérarchiser les fournisseurs à l'aide des boutons duCommandecolonne. Les fournisseurs suivants sont pris en charge :

     - Fournisseur cryptographique Microsoft Base v1.0
     - Fournisseur de chiffrement Microsoft Base DSS et Diffie-Hellman
     - Fournisseur de chiffrement par carte à puce Microsoft Base
     - Fournisseur de chiffrement Microsoft DH SChannel
     - Microsoft Enhanced Cryptographic Provider v1.0
     - Fournisseur cryptographique Microsoft Enhanced DSS et Diffie-Hellman
     - Fournisseur de chiffrement Microsoft Enhanced RSA et AES
     - Fournisseur de chiffrement Microsoft RSA SChannel

Configuration des groupes et des autorisations

Dans leGroupes et autorisationssection, vous pouvez consulter les modèles, les groupes existants et les autorisations d'inscription, ou vous pouvez choisirAjouter de nouveaux groupes et autorisationsbouton pour en ajouter de nouveaux. Le bouton ouvre un formulaire nécessitant les informations suivantes :

- Display name (Nom d'affichage)
- Identifiant de sécurité(CÔTÉ)
- S'inscrire, avec les options AUTORISER | REFUSER | NON DÉFINI

- Inscription automatique, avec les options AUTORISER | REFUSER | NON DÉFINI

Configurer les modèles de remplacement

Dans leRemplacer les modèlessection, vous pouvez informer Active Directory que le modèle actuel remplace un ou plusieurs modèles créés dans AD. Appliquez le modèle de remplacement en choisissantAjouter un modèle depuis Active Directory à remplaceret en spécifiant le nom commun du modèle de remplacement.

Configuration du balisage

Dans leTags — facultatifvolet, vous pouvez appliquer et supprimer des métadonnées sur votre ressource AD. Les balises sont des paires de chaînes clé-valeur dans lesquelles la clé doit être unique à la ressource et la valeur est facultative. Le volet affiche toutes les balises existantes pour la ressource dans un tableau. Les actions suivantes sont prises en charge.

- ChoisissezGérer les tagspour ouvrir leGérer les tagspage.
- Choisissez Ajouter un nouveau tag pour créer un tag. Renseignez leCléle champ et, éventuellement, leValeurchamp. ChoisissezEnregistrer les modificationspour appliquer le tag.
- Choisissez leSupprimercliquez à côté d'un tag pour le marquer pour suppression, puis choisissezEnregistrer les modificationspour confirmer.

Vérifier et créer

Après avoir fourni les informations requises et examiné vos choix, choisissezCréer un modèle. Cela ouvreDétails du modèle, où vous pouvez consulter les paramètres du nouveau modèle, modifier ou supprimer le modèle, gérer les groupes et les autorisations, gérer les modèles remplacés, gérer les balises et configurer la réinscription automatique pour les détenteurs de certificats.

Création d'un modèle de connecteur (CLI)

Utilisez lecréer un modèlecommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Création d'un modèle de connecteur (API)

Utilisez le CreateTemplateaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Répertorier les connecteurs pour Active Directory

Vous pouvez utiliser leAWS Private CAConnecteur pour console Active Directory ouAWS CLIpour répertorier les connecteurs que vous possédez.

Pour répertorier vos connecteurs à l'aide de la console

1. Connectez-vous à votreAWSet ouvrez leAWS Private CAConnecteur pour console Active Directory à l'adresse**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Consultez les informations contenues dans leConnecteurs pour Active Directoryliste. Vous pouvez parcourir plusieurs pages de connecteurs à l'aide des numéros de page en haut à droite. Chaque connecteur occupe une ligne affichant les colonnes d'informations suivantes par défaut.

- ID du connecteur— L'identifiant unique du connecteur.
- Nom du répertoire— La ressource Active Directory associée au connecteur.
- État du connecteur— État du connecteur. Les valeurs possibles sont les suivantes :Création|Actif|Suppression|Échoué.
- État du nom principal du service— État du nom principal du service (SPN) associé au connecteur. Les valeurs possibles sont les suivantes :Création|Actif|Suppression|Échoué.
- État de l'enregistrement dans le répertoire— Statut d'enregistrement du directeur associé. Les valeurs possibles sont les suivantes :Création|Actif|Suppression|Échoué.
- Créé à— Horodatage lors de la création du connecteur.

En choisissant l'icône représentant une roue dentée dans le coin supérieur droit de la console, vous pouvez personnaliser le nombre de connecteurs affichés sur une page à l'aide duTaille de pagepréférence.

Pour répertorier vos connecteurs à l'aide duAWS CLI

Utilisez lelist-connecteurscommande pour répertorier vos connecteurs.

Pour répertorier vos connecteurs à l'aide de l'API

Utilisez leListConnectorsaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Modèles de connecteurs de liste

Vous pouvez utiliser leAWS Private CAConnecteur pour console Active Directory ouAWS CLIpour répertorier les modèles de connecteurs que vous possédez. Les modèles de connecteurs sont basés surAWS Private CA BlankEndEntityCertificate_APIPassThrough/V1modèles.

Pour répertorier vos modèles à l'aide de la console

1. Connectez-vous à votreAWSet ouvrez leAWS Private CAConnecteur pour console Active Directory à l'adresse**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Choisissez un connecteur parmi lesConnecteurs pour Active Directorylistez puis choisissezAfficher les détails.
3. Sur la page de détails du connecteur, consultez les informations contenues dans leModèlessection. Vous pouvez parcourir plusieurs pages de modèles à l'aide des numéros de page en haut à droite. Chaque modèle occupe une ligne affichant les colonnes d'informations suivantes.

- Nom du modèle— Le nom lisible par l'homme du modèle.
- État du modèle— État du modèle. Les valeurs possibles sont les suivantes :Actif|Suppression.
- ID du modèle— L'identifiant unique du modèle.

Pour répertorier vos modèles à l'aide duAWS CLI

Utilisez lemodèles de listescommande pour répertorier les modèles pour le connecteur spécifié.

Pour répertorier vos modèles à l'aide de l'API

Utilisez le ListTemplatesaction dans leAWS Private CAConnecteur pour l'API Active Directory permettant de répertorier les modèles pour le connecteur spécifié.

Afficher les détails du connecteur

Utilisez les procédures suivantes pour afficher les détails de configuration d'un connecteur dans la console, la ligne de commande ou l'API pourAWS Private CAConnecteur pour Active Directory.

Afficher le connecteur (console)

Pour afficher les détails d'un connecteur (console)

1. Connectez-vous à votreAWSet ouvrez leAWS Private CAConnecteur pour console Active Directory à**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Choisissez un connecteur parmiConnecteurs pour Active Directorylistez puis choisissezAfficher les détails.
3. Sur la page des détails du connecteur, passez en revue les informations contenues dans le volet Détails du connecteur, qui inclut les éléments suivants :
   - ID du connecteur
   - État du connecteur
   - Informations supplémentaires sur le statut
   - Connecteur ARN
   - Point de terminaison du serveur de politique d'inscription aux
   - Nom du répertoire
   - ID du répertoire
   - Autorité de certification privée AWSsujet
   - Autorité de certification privée AWSstatut
   - Points de terminaison et groupes de sécurité VPC
4. Dans leModèlesvolet, vous pouvez créer ou gérer des modèles associés au connecteur.
5. À partir duNom principal du service (SPN)volet, vous pouvez afficher le nom principal du service associé au connecteur.
6. À partir duInscription à l'annuairevolet, vous pouvez afficher ou modifier l'enregistrement du répertoire associé au connecteur.
7. À partir duBalises —optionnelvolet, vous pouvez créer ou gérer les balises associées au connecteur.

Afficher le connecteur (CLI)

Utilisez leget-connectorcommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Afficher le connecteur (API)

Utilisez leGetConnectoraction dans leAWS Private CAConnecteur pour l'API Active Directory.

Afficher les détails du modèle de connecteur

Utilisez les procédures suivantes pour afficher les détails de configuration d'un modèle de connecteur à l'aide de la console, de la ligne de commande ou de l'API pourAWS Private CAConnecteur pour Active Directory

Afficher le modèle (console)

Pour afficher les détails d'un modèle de connecteur (console)

1. Connectez-vous à votreAWSet ouvrez leAWS Private CAConnecteur pour console Active Directory à l'adresse**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Choisissez un connecteur parmiConnecteurs pour Active Directorylistez puis choisissezAfficher les détails.
3. Sur la page de détails du connecteur, consultez les informations contenues dans leModèlessection, et sélectionnez le modèle que vous souhaitez inspecter. Ensuite, choisissezAfficher les détails.
4. Sur la page de détails,Détails du modèlele volet affiche les informations suivantes concernant le modèle :

    - Nom du modèle
    - ID du modèle
    - État du modèle
    - Version du schéma du modèle
    - Version du modèle
    - Modèle ARN
    - Type de certificat
    - Inscription automatique activée
    - Période de validité
    - Période de renouvellement
    - Exigences relatives au nom du sujet
    - Exigences relatives aux noms alternatifs du sujet

- Paramètres de demande de certificat et d'inscription
- Catégorie de fournisseur de cryptographie
- Algorithme clé
- Taille de clé minimale (bits)
- Algorithme de hachage
- Fournisseurs de cryptographie
- Paramètres d'extension d'utilisation des clés

Dans ce volet, vous pouvez également effectuer les actions suivantes à l'aide duModifier,Supprimer, etActionsboutons.

- Modifier
- Suppression
- Gérer les groupes et les autorisations— Pour plus d'informations, voirConfiguration des groupes et des autorisations.
- Gérer les modèles obsolètes— Pour plus d'informations, voirRévisez et créez.
- Gérer les tags— Pour plus d'informations, voirConnecteur de balisage pour les ressources AD.
- Réinscrire tous les détenteurs de certificats— Ce paramètre permet d'augmenter automatiquement la version principale d'un modèle. Tous les membres des groupes Active Directory autorisés à s'inscrire à l'aide d'un modèle recevront un nouveau certificat émis à l'aide de ce modèle. Pour en savoir plus, consultez l'API UpdateTemplate.

5. Le volet inférieur affiche une rangée d'onglets permettant de modifier la configuration du modèle.

- Groupes et autorisations— Affichez et gérez les autorisations permettant aux groupes Active Directory d'inscrire des certificats à l'aide de ce modèle. Pour plus d'informations, voirConfiguration des groupes et des autorisations
- Politiques relatives aux applications— Consultez et gérez les modèles de politiques d'application. Pour plus d'informations, voirAttribuer des politiques d'application.
- Modèles remplacés— Affichez et gérez les modèles remplacés. Pour plus d'informations, voirRévisez et créez.
- Tagoptionnel— Affichez et gérez le balisage sur ce modèle. Pour plus d'informations, veuillez consulter Connecteur de balisage pour les ressources AD.

Pour afficher les détails d'un modèle de connecteur (AWS CLI)

Afficher le modèle (CLI)

Utilisez leget-templatecommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Afficher le modèle (API)

Pour afficher les détails d'un modèle de connecteur (API)

Utilisez le GetTemplateaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Gestion des inscriptions dans les annuaires

Pour gérer les inscriptions aux annuaires (console)

Les inscriptions aux annuaires pour les connecteurs peuvent être gérées depuis le niveau supérieur duAWS Private CAConnecteur pour console Active Directory. Cette rubrique décrit les options de gestion disponibles.

1. Connectez-vous à votreAWSet ouvrez leAWS Private CAConnecteur pour console Active Directory à l'adresse**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Dans la zone de navigation de gauche, choisissezInscriptions dans le répertoire.
3. LeInscriptions dans le répertoirela page affiche un tableau des répertoires enregistrés avec les champs suivants :

   - ID du répertoire— L'identifiant unique du répertoire
   - Nom du répertoire— Le nom du site de domaine de l'annuaire
   - Type de répertoire
   - Enregistré— Le statut de l'enregistrement. Les valeurs prises en charge sont CREATION | ACTIVE | DELETING | FAILED.
   - État du répertoire— Le statut du répertoire

   L'utilisation peut utiliserRépertoire des registrespour créer un nouvel enregistrement.
4. Vous pouvez sélectionner l'un des enregistrements listés afin de le gérer. Cela permet deAfficher les détails de l'inscriptionetDésenregistrer le répertoireboutons. LeAfficher les détails de l'inscriptionce bouton ouvre la page de détails de l'enregistrement.

5. LeDétails de l'inscription au répertoirele volet affiche les informations suivantes :

   - Nom du site de domaine de l'annuaire
   - ID du répertoire— L'ID unique du répertoire. En choisissant le lien, vous accédez auAWS Directory Serviceconsole.
   - Type de répertoire
   - État— État du répertoire
   - ARN d'enregistrement dans le répertoire— Le nom de la ressource Amazon associée à l'enregistrement du répertoire
   - Informations supplémentaires sur le statut
6. Dans leConnecteurs et nom principal du service (SPN)volet, vous pouvez gérer les SPN pour le connecteur. Pour plus d'informations, voirAfficher les détails du connecteur.
7. Dans leBalises — facultatifvolet, vous pouvez appliquer et supprimer des métadonnées sur votre ressource AD. Les balises sont des paires de chaînes clé-valeur dans lesquelles la clé doit être unique à la ressource et la valeur est facultative. Le volet affiche toutes les balises existantes pour la ressource dans un tableau. Les actions suivantes sont prises en charge.

   - ChoisissezGérer les tagspour ouvrir leGérer les tagspage.
   - Choisissez Ajouter un nouveau tag pour créer un tag. Renseignez leCléchamp et, éventuellement, leValeurchamp. ChoisissezEnregistrer les modificationspour appliquer le tag.
   - Choisissez leSupprimercliquez à côté d'un tag pour le marquer pour suppression, puis choisissezEnregistrer les modificationspour confirmer.

Pour gérer les enregistrements d'annuaires (CLI)

Créez: Utilisez lecreate-directory-registrationcommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Récupérez:get-directory-registrationcommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Liste:list-directory-registrationscommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Supprimer:delete-directory-registrationcommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Pour gérer les inscriptions aux annuaires (API)

Créez:CreateDirectoryRegistrationaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Récupérez:GetDirectoryRegistrationaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Liste:ListDirectoryRegistrationsaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Supprimer:DeleteDirectoryRegistrationaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Gestion des groupes AD et des autorisations pour les modèles

Pour gérer les groupes de modèles et les autorisations (console)

Les groupes et les autorisations pour un modèle existant peuvent être gérés à partir de la page de détails du modèle. Pour plus d'informations, voir Afficher les détails du modèle de connecteur.

Définissez les autorisations selon lesquelles les groupes peuvent ou ne peuvent pas inscrire des certificats pour un modèle spécifique. Vous fournissez l'identifiant de sécurité (SID) du groupe. Vous définissez ensuite les autorisations d'inscription et d'inscription automatique pour le groupe. Pour l'inscription automatique, l'inscription et l'inscription automatique doivent être réglées sur « Autoriser ».

Rechercher l'identifiant de sécurité du groupe dans Active Directory

Vous pouvez utiliser le script ci-dessous pour rechercher l'identifiant de sécurité du groupe dans Active Directory.

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

Pour gérer les groupes de modèles et les autorisations (CLI)

Créez la commande : create-template-group-access-control-entry dans la section AWS Private CA Connector for Active Directory du. AWS CLI

Mise à jour : commande update-template-group-access-control-entry dans la section AWS Private CA Connector for Active Directory du. AWS CLI

Récupérez la commande : get-template-group-access-control-entry dans la section AWS Private CA Connector for Active Directory du. AWS CLI

Liste : commande list-template-group-access-control-entries dans la section AWS Private CA Connector for Active Directory du. AWS CLI

Supprimer la commande : delete-template-group-access-control-entries dans la section AWS Private CA Connector for Active Directory du. AWS CLI

Pour gérer les groupes de modèles et les autorisations (API)

Créer : CreateTemplateGroupAccessControlEntryaction dans l'API AWS Private CA Connector for Active Directory.

Mise à jour : UpdateTemplateGroupAccessControlEntryaction dans l'API AWS Private CA Connector for Active Directory.

Récupérer : GetTemplateGroupAccessControlEntryaction dans l'API AWS Private CA Connector for Active Directory.

Liste : ListTemplateGroupAccessControlEntriesaction dans l'API AWS Private CA Connector for Active Directory.

Supprimer : DeleteTemplateGroupAccessControlEntryaction dans l'API AWS Private CA Connector for Active Directory.

Configuration du nom principal du service

Pour gérer les noms principaux des services (console)

Le nom principal de service (SPN) d'un connecteur AD existant peut être géré à partir de la page de détails du connecteur. Pour plus d'informations, voir Gestion de l'enregistrement dans les annuairesAfficher les détails du connecteur

Pour gérer les noms principaux des services (CLI)

Créez: create-service-principal-namecommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Récupérez: get-service-principal-namecommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Liste: list-service-principal-namescommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Supprimer: delete-service-principal-namecommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Pour gérer les noms principaux des services (API)

Créez: CreateServicePrincipalNameaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Récupérez: GetServicePrincipalNameaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Liste: ListServicePrincipalNamesaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Supprimer: DeleteServicePrincipalNameaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Connecteur de balisage pour les ressources AD

Vous pouvez appliquer des balises à vos connecteurs, modèles et enregistrements de répertoires. Le balisage ajoute des métadonnées à une ressource qui peut faciliter l'organisation et la gestion.

Pour gérer le balisage des ressources (console)

Le balisage des ressources existantes est géré sur la page de détails de la ressource. Pour de plus amples informations, consultez les procédures suivantes.

- Afficher les détails du modèle de connecteur
- Gestion des inscriptions dans les annuaires

Pour gérer le balisage des ressources (CLI)

Tag:tag-ressourcecommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Balises de liste: list-tags-for-resourcecommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Débalancer:untag-resourcecommande dans leAWS Private CASection Connecteur pour Active Directory duAWS CLI.

Pour gérer le balisage des ressources (API)

Tag: TagResourceaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Balises de liste: ListTagsForResourceaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Débalancer: UntagResourceaction dans leAWS Private CAConnecteur pour l'API Active Directory.

Important - Il est acceptable d'utiliser des balises pour étiqueter des objets contenant des données confidentielles. Cependant, les balises elles-mêmes ne doivent contenir aucune information personnelle identifiable (PII), sensible ou confidentielle.

Autorité de certification privée AWS Connecteur pour le protocole SCEP (Simple Certificate Enrollment Protocol) (version préliminaire)

Connector for SCEP est en version préliminaire AWS Private CA et est sujet à modification.

Qu'est-ce que Connector for SCEP ?

Le connecteur pour le protocole SCEP (Simple Certificate Enrollment Protocol) relie AWS Private Certificate Authority vos appareils mobiles et équipements réseau compatibles avec le protocole SCEP. Avec Connector for SCEP, vous pouvez AWS Private CA émettre des certificats et inscrire vos appareils SCEP. Connector for SCEP peut être utilisé avec les systèmes de gestion des appareils mobiles (MDM) courants et est conçu pour fonctionner avec des clients ou des terminaux compatibles avec le SCEP.

Rubriques

Caractéristiques du connecteur pour SCEP

Support du protocole SCEP - Le SCEP est un protocole largement adopté pour obtenir des certificats d'identité numérique auprès d'une autorité de certification (CA) et les distribuer sur des appareils mobiles et du matériel réseau. Vous pouvez utiliser Connector for SCEP pour vous aider à inscrire vos points de terminaison à l'aide de SCEP.

Inscription d'appareils mobiles : vous pouvez utiliser Connector for SCEP avec les systèmes MDM les plus courants, notamment Microsoft Intune et Jamf Pro.

Émission de certificats à grande échelle : une fois que vous avez configuré vos appareils compatibles SCEP pour demander des certificats via le point de terminaison SCEP du connecteur, vos clients peuvent automatiquement demander des certificats auprès de. AWS Private CA

Comment démarrer avec Connector for SCEP

Pour commencer, lancez l'assistant guidé depuis la console de gestion Connector for SCEP qui vous aide à créer un connecteur et à désigner l'autorité de certification privée à utiliser avec le connecteur. Une fois ces étapes terminées, Connector for SCEP fournit un point de terminaison et d'autres paramètres de configuration que vous pouvez saisir dans vos systèmes MDM ou votre équipement réseau. Après avoir configuré vos systèmes MDM ou votre équipement réseau, vos clients demanderont automatiquement des certificats à AWS Private CA. Pour en savoir plus sur la façon de démarrer avec Connector for SCEP, consultezCommencer à utiliser AWS Private Certificate Authority Connector for SCEP.

Services connexes

Le connecteur pour SCEP est associé aux AWS services suivants.

- AWS Private Certificate Authority- vous AWS Private CA fournit un service d'autorité de certification privée hautement disponible, sans l'investissement initial et les coûts de maintenance permanents liés à l'exploitation de votre propre autorité de certification privée.
- AWS Private CA Connecteur pour Active Directory - Le connecteur pour AD lie votre Active Directory (AD) à AWS Private CA. Le connecteur négocie l'échange de certificats entre AWS Private CA les utilisateurs et les machines gérés par votre AD.

Accès au connecteur pour SCEP

Vous pouvez créer, accéder et gérer vos connecteurs Connector for SCEP à l'aide de l'une des interfaces suivantes :

- AWS Management Console- Fournit une interface Web que vous pouvez utiliser pour accéder à Connector for SCEP. Voir Connector pour la console de gestion SCEP.
- AWS Command Line Interface- Fournit des commandes pour un large éventail de AWS services, y compris Connector for SCEP. AWS CLI Il est pris en charge sur Windows, macOS et Linux. Pour plus d'informations, consultez AWS Command Line Interface.

- AWS SDK - Fournissez des API spécifiques au langage et prenez en charge de nombreux détails de connexion, tels que le calcul des signatures, la gestion des nouvelles tentatives de demande et la gestion des erreurs. Pour plus d'informations, consultez AWS Command Line Interface.
- Connecteur pour l'API SCEP : fournit des actions d'API de bas niveau que vous appelez à l'aide de requêtes HTTPS. L'utilisation de l'API Connector for SCEP est le moyen le plus direct d'accéder au service. Cependant, l'API Connector for SCEP nécessite que votre application gère des détails de bas niveau tels que la génération du hachage pour signer la demande et la gestion des erreurs. Pour plus d'informations, consultez la section Connector for SCEP API reference.

Tarification du connecteur pour SCEP

Le connecteur pour SCEP est proposé en tant que fonctionnalité sans Autorité de certification privée AWS frais supplémentaires. Vous ne payez que pour les AWS Private Certificate Authority opérations et les certificats utilisés pour créer et mettre à jour des connecteurs.

Pour obtenir les dernières informations sur Autorité de certification privée AWS les prix, consultez la section AWS Private Certificate Authority Tarification. Vous pouvez également utiliser le calculateur de AWS prix pour estimer les coûts.

Connecteur pour les concepts SCEP

> Connector for SCEP est en version préliminaire AWS Private CA et est sujet à modification.

Le connecteur pour SCEP est une fonctionnalité complémentaire pour AWS Private Certificate Authority.

Les concepts clés de Connector for SCEP sont les suivants :

Demande de signature de certificat (CSR)

: Les informations requises fournies à une autorité de certification pour qu'un certificat numérique soit émis. Ces informations contiennent une clé publique ainsi qu'une identité.

Mot de passe du défi

: Le protocole SCEP utilise des mots de passe de défi pour authentifier une demande avant de délivrer un certificat à une autorité de certification. Connector for SCEP gère les mots de passe de

défi SCEP en fonction du type de connecteur. Pour plus d'informations, consultez Fonctionnement de Connector for SCEP.

Révocation du certificat

La révocation d'un certificat est le processus de révocation d'un certificat émis avant sa date d'expiration. Vous pouvez révoquer le certificat CA privé associé à un connecteur RevokeCertificateen appelant l'API, le AWS SDK ou AWS Command Line Interface. AWS CloudFormation

Connecteur pour SCEP

Un connecteur pour les liaisons SCEP AWS Private CA vers vos appareils compatibles SCEP.

Gestion des appareils mobiles

La gestion des appareils mobiles (MDM) permet aux administrateurs informatiques de contrôler, de sécuriser et d'appliquer des politiques sur les smartphones, tablettes et autres terminaux ou appareils. De nombreux systèmes MDM proposent des intégrations intégrées pour l'inscription de certificats basés sur le SCEPM.

SCEP

Le SCEP est un protocole standardisé (RFC 8894) permettant de distribuer automatiquement les certificats. Le protocole fournit un point de terminaison permettant aux appareils de demander des certificats à une autorité de certification. Le SCEP utilise des mots de passe de type challenge pour autoriser l'émission de certificats aux appareils. Le SCEP est couramment utilisé pour les systèmes de gestion des appareils mobiles (MDM) et les équipements réseau. Les solutions MDM permettent aux administrateurs informatiques de contrôler, de sécuriser et d'appliquer des politiques sur les smartphones, les tablettes et d'autres entités telles que les postes de travail Apple. La plupart des solutions MDM prennent en charge le SCEP, telles que Microsoft Intune, Apple MDM et Jamf Pro. La plupart des équipements réseau, tels que les routeurs, les équilibreurs de charge, les hubs Wi-Fi, les appareils VPN et les pare-feux, utilisent le SCEP pour l'enregistrement automatique des certificats.

profil SCEP

Un profil SCEP contient des paramètres de configuration utilisés pour définir le profil de certificat. Cela inclut la période de validité du certificat, la taille de la clé, le nom de configuration SCEP, le mot de passe du défi, le nombre de tentatives infructueuses et l'intervalle entre les tentatives, ainsi que d'autres informations relatives à l'émission de certificats. Les systèmes MDM et les plateformes de gestion de certificats envoient généralement le profil SCEP au client qui demandera un certificat d'authentification.

Fonctionnement de Connector for SCEP

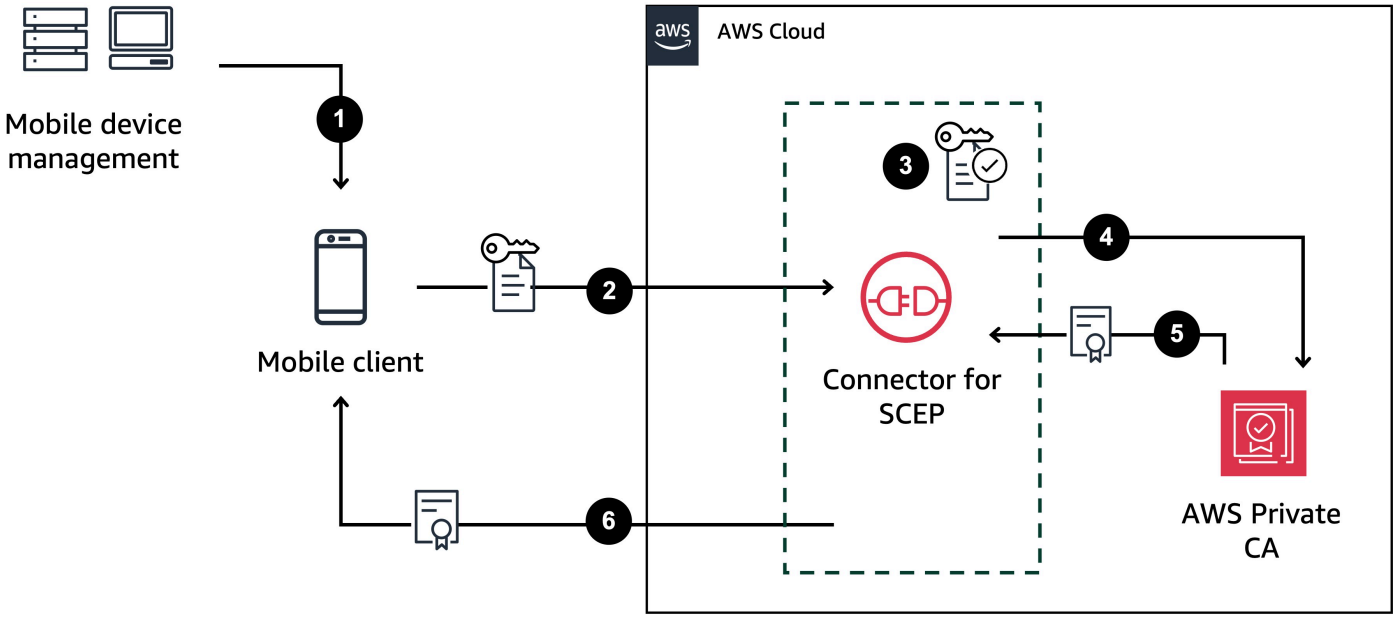
Connector for SCEP est en version préliminaire AWS Private CA et est sujet à modification.

Le protocole SCEP (Simple Certificate Enrollment Protocol) est un protocole standard utilisé pour l'inscription et le renouvellement des certificats. Connector for SCEP est un serveur SCEP basé sur la norme RFC 8894 qui émet automatiquement des certificats AWS Private Certificate Authority à vos clients SCEP. Lorsque vous créez un connecteur, Connector for SCEP fournit un point de terminaison HTTPS auquel les clients SCEP peuvent demander des certificats. Les clients s'authentifient à l'aide d'un mot de passe de défi inclus dans leur demande de signature de certificat (CSR) adressée au service. Vous pouvez utiliser Connector for SCEP avec les solutions de gestion des appareils mobiles (MDM) les plus populaires, notamment Microsoft Intune et Jamf Pro, pour inscrire des appareils mobiles. Il fonctionne avec n'importe quel client ou point de terminaison compatible avec le SCEP.

Connector for SCEP propose deux types de connecteurs : à usage général et Connector for SCEP pour Microsoft Intune. Les sections suivantes décrivent leur fonctionnement.

Usage général

Un connecteur à usage général est conçu pour fonctionner avec les terminaux compatibles avec le protocole SCEP—à l'exception de Microsoft Intune— pour lequel nous disposons d'un connecteur spécial. Les connecteurs polyvalents vous permettent de gérer les mots de passe du défi SCEP. Le schéma suivant utilise un système de gestion des appareils mobiles (MDM) à titre d'exemple, mais les mêmes fonctionnalités s'appliquent si vous utilisez un système ou un appareil analogue compatible avec le protocole SCEP.

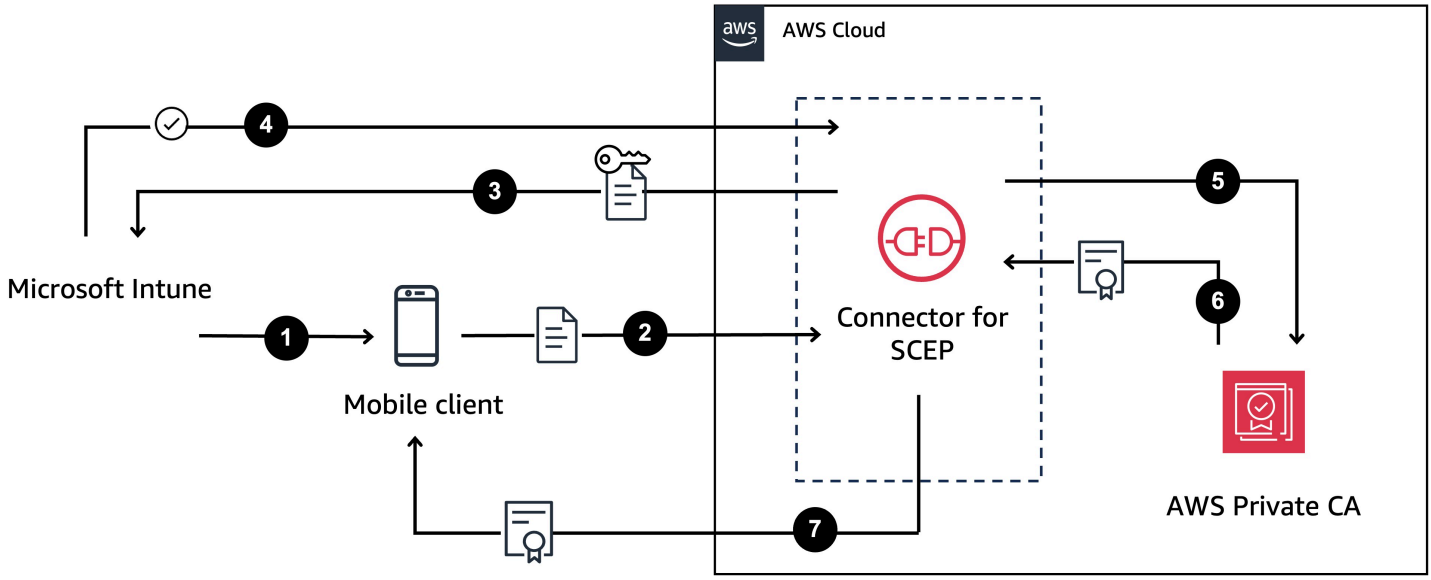


1. Le système MDM (ou appareil ou système analogue) envoie un profil SCEP au client mobile. Un profil SCEP contient les paramètres de configuration utilisés pour définir le profil de certificat, notamment la période de validité du certificat, la taille de la clé, le nom de la configuration SCEP, le mot de passe du défi, le nombre de tentatives infructueuses et l'intervalle entre les tentatives, ainsi que d'autres informations relatives à l'émission de certificats.
2. Le client mobile demande un certificat et envoie également une demande de signature de certificat (CSR) qui inclut un mot de passe de défi.
3. Le connecteur pour SCEP valide le mot de passe du défi. S'il est valide, le service demande un certificat AWS Private CA au nom du client mobile.
4. AWS Private CA émet le certificat et l'envoie à Connector for SCEP.
5. Le connecteur pour SCEP envoie le certificat émis au client mobile.

AWS Private Certificate Authority Connecteur pour SCEP pour Microsoft Intune

AWS Private CA Le connecteur pour SCEP pour Microsoft Intune est conçu pour être utilisé avec Microsoft Intune. Avec le type de connecteur Connector for SCEP pour Microsoft Intune, vous utiliserez Microsoft Intune pour gérer vos mots de passe de défi SCEP. Pour plus d'informations sur l'utilisation de Connector for SCEP avec Microsoft Intune, consultez. Utilisation du connecteur pour SCEP pour Microsoft Intune

Lorsque vous utilisez Connector for SCEP avec Microsoft Intune, certaines fonctionnalités sont activées en accédant à Microsoft Intune via l'API Microsoft. Votre utilisation du connecteur pour le SCEP et des AWS services associés ne vous dispense pas d'avoir besoin d'une licence valide pour utiliser le service Microsoft Intune. Vous devriez également consulter les politiques de protection des applications Microsoft Intune®.



1. Microsoft Intune envoie un profil SCEP au client mobile. Le profil contient un mot de passe de défi crypté que le client mobile place dans le CSR.
2. Le client mobile demande un certificat et envoie le CSR à Connector for SCEP.
3. Connector for SCEP envoie le CSR à Microsoft Intune pour autorisation.
4. Microsoft Intune déchiffre le mot de passe du défi dans le CSR. S'il est valide, Microsoft Intune envoie l'approbation à Connector pour que SCEP émette le certificat au client mobile.
5. Connector for SCEP demande un certificat AWS Private CA au nom du client mobile.
6. AWS Private CA émet le certificat et l'envoie à Connector for SCEP.
7. Le connecteur pour SCEP envoie le certificat émis au client mobile.

Considérations et limites liées à l'utilisation de Connector for SCEP

Considérations

Modes de fonctionnement CA

Vous ne pouvez utiliser Connector for SCEP qu'avec des autorités de certification privées utilisant un mode de fonctionnement général. Connector for SCEP émet par défaut des certificats d'une durée de validité d'un an. Une autorité de certification privée utilisant un mode de certificat de courte durée ne prend pas en charge l'émission de certificats dont la durée de validité est supérieure à sept jours. Pour plus d'informations sur les modes de fonctionnement, consultezModes d'autorité de certification.

Mots de passe contestés

- Diffusez vos mots de passe difficiles avec le plus grand soin et ne les partagez qu'avec des personnes et des clients dignes de confiance. Un mot de passe de défi unique peut être utilisé pour émettre n'importe quel certificat, quel que soit le sujet et le réseau SAN, ce qui représente un risque pour la sécurité.
- Si vous utilisez un connecteur à usage général, nous vous recommandons de changer fréquemment vos mots de passe de défi manuellement.

Conformité à la RFC 8894

Le connecteur pour SCEP s'écarte du protocole RFC 8894 en fournissant des points de terminaison HTTPS au lieu de points de terminaison HTTP.

CSR

- Si une demande de signature de certificat (CSR) envoyée à Connector for SCEP n'inclut pas l'extension Extended Key Usage (EKU), nous définirons la valeur EKU sur. `clientAuthentication` Pour plus d'informations, voir 4.2.1.12. Utilisation étendue des clés dans la RFC 5280.
- Nous prenons en charge `ValidityPeriod` et `ValidityPeriodUnits` personnalisons les attributs dans les CSR. Si votre CSR n'inclut pas de`ValidityPeriod`, nous délivrons un certificat dont la durée de validité est d'un an. N'oubliez pas que vous ne pourrez peut-être pas définir ces attributs dans votre système MDM. Mais si vous pouvez les configurer, nous les soutenons. Pour plus d'informations sur ces attributs, consultez SzEnrollment_Name_Value_Pair.

Partage de terminaux

Distribuez les points de terminaison d'un connecteur uniquement aux parties de confiance. Traitez les points de terminaison comme secrets, car toute personne capable de trouver votre nom de domaine complet et votre chemin uniques peut récupérer votre certificat CA.

Limites

Les limitations suivantes s'appliquent au Connector for SCEP.

Mots de passe de défi dynamiques

Vous ne pouvez créer des mots de passe de défi statiques qu'avec des connecteurs à usage général. Pour utiliser des mots de passe dynamiques avec un connecteur à usage général, vous devez créer votre propre mécanisme de rotation qui utilise les mots de passe statiques du connecteur. Connector for SCEP for Microsoft Intune Les types de connecteurs prennent en charge les mots de passe dynamiques, que vous gérez à l'aide de Microsoft Intune.

HTTP

Connector for SCEP prend uniquement en charge le protocole HTTPS et crée des redirections pour les appels HTTP. Si votre système repose sur le protocole HTTP, assurez-vous qu'il est compatible avec les redirections HTTP fournies par Connector for SCEP.

CA privées partagées

Vous ne pouvez utiliser Connector for AD qu'avec des autorités de certification privées dont vous êtes le propriétaire.

Configuration du connecteur pour SCEP

> Connector for SCEP est en version préliminaire AWS Private CA et est sujet à modification.

Les procédures décrites dans cette section vous aident à démarrer avec Connector for SCEP. Cela suppose que vous avez déjà créé un AWS compte. Après avoir effectué les étapes de cette page, vous pouvez créer un connecteur pour le SCEP.

Rubriques

Étape 1 : créer une AWS Identity and Access Management politique

Pour créer un connecteur pour SCEP, vous devez créer une politique IAM qui accorde à Connector for SCEP la capacité de créer et de gérer les ressources nécessaires au connecteur, et d'émettre des certificats en votre nom. Pour plus d'informations sur l'IAM, voir Qu'est-ce que l'IAM ? dans le guide de l'utilisateur IAM.

L'exemple suivant est une politique gérée par le client que vous pouvez utiliser pour 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": "*"
        }
    ]
}
```

Étape 2 : créer une autorité de certification privée

Pour utiliser Connector for SCEP, vous devez associer une autorité de certification privée AWS Private Certificate Authority au connecteur. Nous vous recommandons d'utiliser une autorité de certification privée uniquement pour le connecteur, en raison des failles de sécurité inhérentes au protocole SCEP.

L'autorité de certification privée doit répondre aux exigences suivantes :

- Il doit être actif et utiliser le mode de fonctionnement général.
- Vous devez être propriétaire de l'autorité de certification privée. Vous ne pouvez pas utiliser une autorité de certification privée qui a été partagée avec vous par le biais du partage entre comptes.

Tenez compte des considérations suivantes lors de la configuration de votre autorité de certification privée à utiliser avec Connector for SCEP :

- Contraintes de nom DNS — Envisagez d'utiliser des contraintes de nom DNS pour contrôler les domaines autorisés ou interdits dans les certificats émis pour vos appareils SCEP. Pour plus d'informations, consultez Comment appliquer les contraintes de nom DNS dans AWS Private Certificate Authority.

- Révocation — Activez les OCSP ou les CRL sur votre autorité de certification privée pour permettre la révocation. Pour plus d'informations, consultez Configuration d'une méthode de révocation des certificats.
- PII — Nous vous conseillons de ne pas ajouter d'informations personnelles identifiables (PII) ou d'autres informations confidentielles ou sensibles dans vos certificats CA. En cas de faille de sécurité, cela permet de limiter l'exposition d'informations sensibles.
- Stocker les certificats racines dans les magasins de confiance : stockez vos certificats CA racine dans les magasins de confiance de votre appareil, afin de pouvoir vérifier les certificats et les valeurs de retour de GetCertificateAuthorityCertificate. Pour plus d'informations sur les magasins de confiance auxquels ils se rapportent AWS Private CA, consultezRoot CA .

Pour plus d'informations sur la création d'une autorité de certification privée, consultezCréation d'une autorité de certification privée.

Étape 3 : créer un partage de ressources à l'aide de AWS Resource Access Manager

Si vous utilisez Connector for SCEP par programmation à l'aide du AWS SDK ou de l' AWS Command Line Interface API Connector for SCEP, vous devez partager votre autorité de certification privée avec Connector for SCEP en utilisant le partage principal de service. AWS Resource Access Manager Cela donne à Connector for SCEP un accès partagé à votre autorité de certification privée. Lorsque vous créez un connecteur dans la AWS console, nous créons automatiquement le partage de ressources pour vous. Pour plus d'informations sur le partage de ressources, voir Création d'un partage de ressources dans le Guide de AWS RAM l'utilisateur.

Pour créer un partage de ressources à l'aide de AWS CLI, vous pouvez utiliser la AWS RAM create-resource-share commande. La commande suivante crée un partage de ressources. Spécifiez l'ARN de l'autorité de certification privée que vous souhaitez partager comme valeur 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
```

Le principal de service qui appelle `CreateConnector` dispose des autorisations d'émission de certificats sur l'autorité de certification privée. Pour empêcher les responsables de service qui utilisent Connector for SCEP d'avoir un accès général à vos Autorité de certification privée AWS ressources, limitez leurs autorisations à l'aide de. `CalledVia`

Commencer à utiliser AWS Private Certificate Authority Connector for SCEP

> Connector for SCEP est en version préliminaire AWS Private CA et est sujet à modification.

Avec AWS Private Certificate Authority Connector for SCEP, vous pouvez émettre des certificats depuis votre autorité de certification privée vers des appareils compatibles avec le SCEP et des systèmes de gestion des appareils mobiles (MDM). Lorsque vous créez un connecteur, vous AWS Private Certificate Authority créez une URL SCEP publique pour demander des certificats et vous fournissez également des informations que vous pouvez utiliser pour les intégrer à vos systèmes MDM.

Pour délivrer des certificats, vous devez créer une autorité de certification AWS Private Certificate Authority privée, créer un connecteur, puis configurer vos systèmes et appareils MDM compatibles SCEPM pour demander des certificats au connecteur.

Rubriques

Avant de commencer

Le didacticiel suivant vous guide tout au long du processus de création d'un connecteur pour le SCEP.

Pour suivre ce didacticiel, vous aurez besoin d'une autorité de certification privée et d'un appareil compatible avec le protocole SCEPM. Vous devez également d'abord remplir les conditions requises répertoriées dans la Configuration du connecteur pour SCEP section.

La procédure suivante explique comment créer un connecteur à l'aide de la AWS console.

Tâches

- Étape 1 : Création d'un connecteur
- Étape 2 : Copier les détails du connecteur dans votre système MDM

Étape 1 : Création d'un connecteur

Vous allez créer soit un connecteur à usage général, soit un connecteur pour SCEP pour Microsoft Intune. Les connecteurs à usage général sont conçus pour être utilisés avec les points de terminaison compatibles SCEP, et vous gérez les mots de passe de défi SCEP. Connector for SCEP pour Microsoft Intune est destiné à être utilisé avec Microsoft Intune, et vous gérez les mots de passe de défi à l'aide de Microsoft Intune.

General-purpose

Pour créer un connecteur à usage général

Connectez-vous à votre AWS compte et ouvrez la console Connector for SCEP à **https://console.aws.amazon.com/pca-connector-scep/home** l'adresse.

1. Sélectionnez Créer un connecteur.
2. Sur la page Créer un connecteur, attribuez éventuellement un nom convivial au connecteur dans le champ Name tag. Le nom sera affiché dans votre liste de connecteurs. Si vous le souhaitez, vous pouvez ajouter d'autres balises au connecteur en sélectionnant Ajouter d'autres balises. Une étiquette est une étiquette que vous attribuez à une AWS ressource. Chaque balise est constituée d'une clé et d'une valeur facultative. Vous pouvez utiliser des balises pour rechercher et filtrer vos ressources ou suivre vos AWS coûts.
3. Sous Type de connecteur, sélectionnez Usage général.
4. Sous Autorité de certification privée, choisissez l'autorité de certification privée à utiliser avec ce connecteur. Vous pouvez également en créer une nouvelle en sélectionnant Créer une autorité de certification privée. En raison des vulnérabilités inhérentes au protocole SCEP, nous recommandons d'utiliser une autorité de certification privée dédiée à ce connecteur. Si vous avez créé une nouvelle autorité de certification, lorsque vous avez fini de la créer AWS Private CA, retournez à la console Connector for SCEP et actualisez la liste des autorités de certification privées. Votre nouvelle autorité de certification privée devrait être disponible pour la sélection.

5. Sous Mot de passe du défi, sélectionnez Générer automatiquement le mot de passe du défi. Nous vous fournirons un mot de passe de défi statique lors de la création de ce connecteur.
6. Sélectionnez Créer un connecteur.

Microsoft Intune

Pour créer un connecteur pour SCEP pour Microsoft Intune

Connectez-vous à votre AWS compte et ouvrez la console Connector for SCEP à **https://console.aws.amazon.com/pca-connector-scep/home** l'adresse.

1. Sélectionnez Créer un connecteur.
2. Sur la page Créer un connecteur, attribuez éventuellement un nom convivial au connecteur dans le champ Name tag. Le nom sera affiché dans votre liste de connecteurs. Si vous le souhaitez, vous pouvez ajouter d'autres balises au connecteur en sélectionnant Ajouter d'autres balises. Une étiquette est une étiquette que vous attribuez à une AWS ressource. Chaque balise est constituée d'une clé et d'une valeur facultative. Vous pouvez utiliser des balises pour rechercher et filtrer vos ressources ou suivre vos AWS coûts.
3. Sous Type de connecteur, sélectionnez Microsoft Intune.
   a. Pour l'ID de l'application (client), entrez l'ID de l'application (client) indiqué lors de l'enregistrement de votre application Microsoft Entra ID. Pour plus d'informations sur l'utilisation de Microsoft Intune avec Connector for SCEP, consultez. Utilisation de Connector for SCEP avec des systèmes MDM
   b. Pour l'ID de répertoire (tenant) ou le domaine principal, entrez l'ID de répertoire (tenant) ou le domaine principal indiqué lors de l'enregistrement de votre application Microsoft Entra ID.
4. Sous Autorité de certification privée, choisissez l'autorité de certification privée à utiliser avec ce connecteur. Vous pouvez également en créer une nouvelle en sélectionnant Créer une autorité de certification privée. En raison des vulnérabilités inhérentes au protocole SCEP, nous recommandons d'utiliser une autorité de certification privée dédiée à ce connecteur. Si vous avez créé une nouvelle autorité de certification, lorsque vous avez fini de la créer AWS Private CA, retournez à la console Connector for SCEP et actualisez la liste des autorités de certification privées. Votre nouvelle autorité de certification privée devrait être disponible pour la sélection.
5. Sélectionnez Créer un connecteur.

Étape 2 : Copier les détails du connecteur dans votre système MDM

Après avoir créé votre connecteur, vous devez copier les informations suivantes du connecteur dans votre système MDM. Pour afficher les détails d'un connecteur à l'aide de la console, sélectionnez le connecteur dans la liste de la page Connecteurs pour la console SCEP.

- URL SCEP publique : il s'agit du point de terminaison du connecteur auprès duquel vos clients SCEP demanderont des certificats. Veillez à ne fournir ce point de terminaison qu'à des entités de confiance.
- (Usage général) Mot de passe de défi - Sous Mots de passe de défi, sélectionnez le mot de passe que vous avez automatiquement généré lors de la procédure précédente, puis sélectionnez Afficher le mot de passe pour afficher le mot de passe. Pour créer un mot de passe supplémentaire, sélectionnez Créer un mot de passe. Veillez à distribuer les mots de passe avec soin et à ne les distribuer qu'aux personnes et aux clients dignes de confiance. Un mot de passe de défi unique peut être utilisé pour émettre n'importe quel certificat, quel que soit le sujet et le réseau SAN. Il doit donc être manipulé avec précaution.
- Valeurs Open ID (Microsoft Intune) - Si vous effectuez une intégration à Microsoft Intune, vous devez copier l'émetteur de l'Open ID, le sujet Open ID et le public Open ID dans le code d'identification OpenID Connect (OIDC) associé à votre inscription à l'application Microsoft Entra. Pour plus d'informations, voir Utilisation de Connector for SCEP avec des systèmes MDM.

Utilisation de Connector for SCEP avec des systèmes MDM

> Connector for SCEP est en version préliminaire AWS Private Certificate Authority et est sujet à modification.

Les sections suivantes décrivent comment utiliser Connector for SCEP avec des systèmes de gestion des appareils mobiles (MDM) spécifiques.

Rubriques

- Utilisation du connecteur pour SCEP avec Jamf Pro
- Utilisation du connecteur pour SCEP pour Microsoft Intune

Utilisation du connecteur pour SCEP avec Jamf Pro

Vous pouvez l'utiliser AWS Private CA en tant qu'autorité de certification (CA) externe avec la solution de gestion des appareils mobiles (MDM) Jamf Pro. Ce guide fournit des instructions sur la façon de configurer Jamf Pro en tant que proxy SCEP après avoir créé un AWS Private Certificate Authority connecteur pour SCEP.

Exigences relatives à Jamf Pro

Votre implémentation de Jamf Pro doit répondre aux exigences suivantes.

- Vous devez utiliser Jamf Pro 10.0.0 ou version ultérieure.
- Vous devez activer le paramètre Activer l'authentification basée sur les certificats dans Jamf Pro. Vous trouverez des informations détaillées sur ce paramètre sur la page des paramètres de sécurité de Jamf Pro dans la documentation de Jamf Pro.

Prérequis

Pour utiliser Connector for SCEP avec Jamf Pro, vous devez d'abord créer une autorité de certification privée et un connecteur à usage général pour SCEP. Pour obtenir des instructions, veuillez consulter Configuration du connecteur pour SCEP.

Configuration AWS Private CA en tant que CA externe dans Jamf Pro

Après avoir créé un connecteur pour le SCEP, vous devez le définir AWS Private CA en tant que CA externe dans Jamf Pro. Vous pouvez AWS Private CA le définir en tant qu'autorité de certification externe globale. Vous pouvez également utiliser un profil de configuration Jamf Pro AWS Private CA pour émettre différents certificats en fonction de différents cas d'utilisation, tels que l'émission de certificats pour un sous-ensemble d'appareils de votre organisation. Les conseils relatifs à la mise en œuvre des profils de configuration Jamf Pro dépassent le cadre de ce document.

Pour configurer AWS Private CA en tant qu'autorité de certification externe dans Jamf Pro

1. Dans la console Jamf Pro, accédez à la page des paramètres des certificats PKI en accédant à Paramètres > Global > Certificats PKI.
2. Sélectionnez l'onglet Modèle de certificat de gestion.
3. Sélectionnez External CA.

4. Tâche de sélection Modifier.
5. (Facultatif) Sélectionnez Activer Jamf Pro en tant que proxy SCEP pour les profils de configuration. Vous pouvez utiliser les profils de configuration Jamf Pro pour émettre différents certificats adaptés à des cas d'utilisation spécifiques. Pour obtenir des conseils sur l'utilisation des profils de configuration dans Jamf Pro, consultez la section Activation de Jamf Pro en tant que proxy SCEP pour les profils de configuration dans la documentation de Jamf Pro.
6. Sélectionnez Utiliser une autorité de certification externe compatible SCEP pour l'inscription d'ordinateurs et d'appareils mobiles.
7. (Facultatif) Sélectionnez Utiliser Jamf Pro comme proxy SCEP pour l'inscription d'ordinateurs et d'appareils mobiles. Si vous rencontrez des échecs lors de l'installation du profil, consultezRésoudre les problèmes d'installation des profils.
8. Copiez et collez l'URL SCEP publique du connecteur pour SCEP depuis les détails du connecteur vers le champ URL de Jamf Pro. Pour afficher les détails d'un connecteur, choisissez-le dans la liste Connecteurs pour le SCEP. Vous pouvez également obtenir l'URL en appelant GetConnectoret en copiant la `Endpoint` valeur de la réponse.
9. (Facultatif) Entrez le nom de l'instance dans le champ Nom. Par exemple, vous pouvez lui donner un nom AWS Private CA.
10. Sélectionnez Static pour le type de défi.
11. Copiez le mot de passe de défi de votre connecteur et collez-le dans le champ Challenge. Pour consulter les mots de passe de défi de votre connecteur, accédez à la page de détails de votre connecteur dans la AWS console et sélectionnez le bouton Afficher le mot de passe. Vous pouvez également obtenir le mot de passe de défi d'un connecteur en appelant GetChallengePassword et en copiant la `Password` valeur de la réponse.
12. Collez le mot de passe du défi dans le champ Vérifier le défi.
13. Choisissez une taille de clé. Nous recommandons une taille de clé de 2048 ou plus.
14. (Facultatif) Sélectionnez Utiliser comme signature numérique. Sélectionnez cette option à des fins d'authentification pour accorder aux appareils un accès sécurisé à des ressources telles que le Wi-Fi et le VPN.
15. (Facultatif) Sélectionnez Utiliser pour le chiffrement par clé.
16. (Facultatif) Entrez une chaîne hexadécimale dans le champ Empreinte digitale. Pour obtenir des instructions sur la création d'une empreinte digitale de votre autorité de certification privée, consultez(Facultatif) Ajoutez une empreinte CA.
17. Sélectionnez Save.

Création et téléchargement d'un certificat de signature de profil

Pour utiliser Connector for SCEP avec Jamf Pro, vous devez fournir les certificats de signature et d'autorité de certification de l'autorité de certification privée associée à votre connecteur. Vous pouvez le faire en téléchargeant un keystore de certificats de signature de profil sur Jamf Pro contenant les deux certificats. Vous devez générer une demande de signature de certificat (CSR) à l'aide de vos processus internes et la faire signer par AWS Private Certificate Authority. Les instructions suivantes expliquent comment créer un keystore de certificats et le télécharger dans Jamf Pro. L'exemple suivant utilise OpenSSL, mais vous pouvez générer une demande de signature de certificat à l'aide de votre méthode préférée.

1. À l'aide d'OpenSSL, générez une clé privée en exécutant la commande suivante :

```
openssl genrsa -out local.key 2048
```

2. Générez une demande de signature de certificat (CSR) :

```
openssl req -new -key local.key -sha512 -out local.csr -
subj "/CN=MySigningCertificate/O=MyOrganization" -addext
 keyUsage=critical,digitalSignature,nonRepudiation
```

3. À l'aide du AWS CLI, émettez le certificat de signature à l'aide du CSR que vous avez généré à la deuxième étape. Exécutez la commande suivante et notez l'ARN du certificat dans la réponse :

```
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. Obtenez le certificat de signature en exécutant la commande suivante à l'aide de l'ARN du certificat indiqué à l'étape 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. Obtenez le certificat CA en exécutant la commande suivante :

```
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. À l'aide d'OpenSSL, générez le keystore du certificat de signature au format p12. Vous utiliserez les `crt` fichiers que vous avez générés au cours des étapes 4 et 5. Exécutez la commande suivante :

```
openssl pkcs12 -export -in local.crt -inkey local.key -certfile ca.crt -name "CA
 Chain" -out local.p12
```

7. Lorsque vous y êtes invité, entrez un mot de passe d'exportation. Ce mot de passe est le mot de passe de votre keystore et vous devrez l'utiliser ultérieurement.
8. Dans Jamf Pro, accédez au modèle de certificat de gestion et accédez au volet CA externe.
9. Au bas du volet CA externe, sélectionnez Modifier la signature et les certificats CA.
10. Suivez les instructions affichées à l'écran pour télécharger la signature et les certificats de l'autorité de certification pour l'autorité de certification externe.

(Facultatif) Ajoutez une empreinte CA

L'ajout d'une empreinte d'autorité de certification permet aux appareils administrés de vérifier l'autorité de certification et de ne demander des certificats qu'à l'autorité de certification.

1. Obtenez le certificat CA privé depuis l'une des AWS Private CA consoles ou à l'aide du GetCertificateAuthorityCertificate. Enregistrez-le sous forme de `ca.pem` fichier.
2. Dans OpenSSL, exécutez la commande suivante :

```
openssl x509 -in ca.pem -sha256 -fingerprint
```

3. Copiez et collez la sortie dans le champ Fingerprint indiqué dans la procédure précédente.

(Facultatif) Installer le certificat lors de l'inscription initiée par l'utilisateur

Pour installer le certificat CA privé de votre connecteur sur un client ou un appareil lors de l'inscription initiée par l'utilisateur, configurez les paramètres d'inscription initiée par l'utilisateur de Jamf Pro. Cela permet à Jamf Pro d'installer vos AWS Private CA certificats sur le client ou l'appareil lorsqu'ils en font la demande. Il est de votre responsabilité de tester votre configuration pour vous assurer qu'elle est compatible avec votre implémentation de Connector for SCEP. Pour plus d'informations sur les paramètres d'inscription initiés par l'utilisateur de Jamf Pro, consultez la section Paramètres d'inscription initiés par l'utilisateur dans la documentation de Jamf Pro.

Résoudre les problèmes d'installation des profils

Si l'installation de votre profil échoue après avoir activé l'option Utiliser Jamf Pro comme proxy SCEP pour l'inscription d'ordinateurs et d'appareils mobiles, essayez ce qui suit.

Message d'erreur	Atténuation
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:15001>`	Si ce message d'erreur s'affiche alors que vous tentez de vous inscrire, recommencez l'inscription. Plusieurs essais peuvent être nécessaires avant que l'inscription ne réussisse.
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:14006>`	Votre mot de passe de défi est peut-être mal configuré. Vérifiez que le mot de passe de défi dans Jamf Pro correspond au mot de passe de défi de votre connecteur.

Utilisation du connecteur pour SCEP pour Microsoft Intune

Vous pouvez l'utiliser AWS Private CA en tant qu'autorité de certification (CA) externe avec le système de gestion des appareils mobiles (MDM) Microsoft Intune. Ce guide fournit des instructions sur la façon de configurer Microsoft Intune après avoir créé un connecteur pour SCEP pour Microsoft Intune.

Prérequis

Avant de créer un connecteur pour SCEP pour Microsoft Intune, vous devez remplir les conditions préalables suivantes.

- Créez un identifiant Entra.
- Créez un tenant Microsoft Intune.
- Créez un enregistrement d'application dans votre identifiant Microsoft Entra. Voir Mettre à jour les autorisations demandées par une application dans l'identifiant Microsoft Entra dans la documentation Microsoft Entra pour savoir comment gérer les autorisations au niveau de

l'application pour l'enregistrement de votre application. L'enregistrement de l'application doit disposer des autorisations suivantes :

- Sous Intune, définissez scep_challenge_provider.
- Pour Microsoft Graph, définissez Application.Read.All et User.Read.

- Vous devez accorder l'application dans votre accord d'administrateur d'inscription à l'application. Pour plus d'informations, consultez la section Accorder le consentement d'un administrateur à l'échelle du locataire pour une application dans la documentation Microsoft Entra.

> **Tip**
>
> Lorsque vous créez l'enregistrement de l'application, prenez note de l'ID de l'application (client) et de l'ID du répertoire (locataire) ou du domaine principal. Lorsque vous créez votre connecteur pour SCEP pour Microsoft Intune, vous devez entrer ces valeurs. Pour plus d'informations sur la façon d'obtenir ces valeurs, voir Création d'une application Microsoft Entra et d'un principal de service pouvant accéder aux ressources dans la documentation Microsoft Entra.

AWS Private CA Autoriser l'utilisation de l'application Microsoft Entra ID

Après avoir créé un connecteur pour SCEP pour Microsoft Intune, vous devez créer un identifiant fédéré dans le cadre de l'enregistrement des applications Microsoft afin que le connecteur pour SCEP puisse communiquer avec Microsoft Intune.

Pour configurer AWS Private CA en tant qu'autorité de certification externe dans Microsoft Intune

1. Dans la console Microsoft Entra ID, accédez aux enregistrements des applications.
2. Choisissez l'application que vous avez créée pour l'utiliser avec Connector for SCEP. L'ID d'application (client) de l'application sur laquelle vous cliquez doit correspondre à celui que vous avez spécifié lors de la création du connecteur.
3. Sélectionnez Certificats et secrets dans le menu déroulant Géré.
4. Sélectionnez l'onglet Federated credentials.
5. Sélectionnez Ajouter un identifiant.
6. Dans le menu déroulant du scénario d'identification fédérée, sélectionnez Autre émetteur.
7. Copiez et collez la valeur de l'émetteur OpenID contenue dans les détails de votre Connector for SCEP for Microsoft Intune dans le champ Émetteur. Pour afficher les détails d'un connecteur,

choisissez-le dans la liste Connecteurs pour le SCEP de la AWS console. Vous pouvez également obtenir l'URL en appelant GetConnectorpuis en copiant la `Issuer` valeur de la réponse.

8. Copiez et collez la valeur OpenID Audience depuis les détails de votre Connector for SCEP pour Microsoft Intune dans le champ Audience. Pour afficher les détails d'un connecteur, choisissez-le dans la liste Connecteurs pour le SCEP de la AWS console. Vous pouvez également obtenir l'URL en appelant GetConnectorpuis en copiant la `Subject` valeur de la réponse.
9. (Facultatif) Entrez le nom de l'instance dans le champ Nom. Par exemple, vous pouvez lui donner un nom AWS Private CA.
10. (Facultatif) Entrez une description dans le champ Description.
11. Sélectionnez Modifier (facultatif) dans le champ Audience. Copiez et collez la valeur du sujet OpenID depuis votre connecteur dans le champ Objet. Vous pouvez consulter la valeur de l'émetteur OpenID sur la page de détails du connecteur de la console. AWS Vous pouvez également obtenir l'URL en appelant GetConnectorpuis en copiant la `Audience` valeur de la réponse.
12. Sélectionnez Ajouter.

Configuration d'un profil de configuration Microsoft Intune

Une fois que vous AWS Private CA avez autorisé à appeler Microsoft Intune, vous devez utiliser Microsoft Intune pour créer un profil de configuration Microsoft Intune qui indique aux appareils de contacter Connector for SCEP pour l'émission de certificats.

1. Créez un profil de configuration de certificat fiable. Vous devez télécharger le certificat CA racine de la chaîne que vous utilisez avec Connector for SCEP dans Microsoft Intune pour établir la confiance. Pour plus d'informations sur la création d'un profil de configuration de certificat fiable, consultez la section Profils de certificats racine fiables pour Microsoft Intune dans la documentation Microsoft Intune.
2. Créez un profil de configuration de certificat SCEP qui oriente vos appareils vers le connecteur lorsqu'ils ont besoin d'un nouveau certificat. Le type de profil du profil de configuration doit être un certificat SCEP. Pour le certificat racine du profil de configuration, assurez-vous d'utiliser le certificat sécurisé que vous avez créé à l'étape précédente.

   Pour les URL du serveur SCEP, copiez et collez l'URL SCEP publique figurant dans les détails de votre connecteur dans le champ URL du serveur SCEP. Pour afficher les détails d'un connecteur, choisissez-le dans la liste Connecteurs pour le SCEP. Vous pouvez également

obtenir l'URL en appelant GetConnector, puis en copiant la `Endpoint` valeur de la réponse. Pour obtenir des conseils sur la création de profils de configuration dans Microsoft Intune, voir Création et attribution de profils de certificat SCEP dans Microsoft Intune dans la documentation Microsoft Intune.

> **Note**
>
> Pour les appareils autres que Mac OS et iOS, si vous ne définissez pas de période de validité dans le profil de configuration, Connector for SCEP émet un certificat d'une validité d'un an. Si vous ne définissez pas de valeur EKU (Extended Key Usage) dans le profil de configuration, Connector for SCEP émet un certificat avec l'EKU défini avec. `Client Authentication (Object Identifier: 1.3.6.1.5.5.7.3.2)` Pour les appareils macOS ou iOS, Microsoft Intune ne respecte `ExtendedKeyUsage` pas `Validity` les paramètres de vos profils de configuration. Pour ces appareils, Connector for SCEP délivre un certificat d'une durée de validité d'un an à ces appareils par le biais de l'authentification du client.

Vérifiez la connexion au connecteur pour SCEP

Après avoir créé un profil de configuration Microsoft Intune qui pointe vers le point de terminaison Connector for SCEP, vérifiez qu'un appareil inscrit peut demander un certificat. Pour confirmer, assurez-vous qu'il n'y a aucun échec d'attribution de politique. Pour confirmer, sur le portail Intune, accédez à Appareils > Gérer les appareils > Configuration et vérifiez que rien n'est répertorié dans la section Défaillances d'attribution des politiques de configuration. Si tel est le cas, confirmez votre configuration à l'aide des informations issues des procédures précédentes. Si votre configuration est correcte et que des défaillances persistent, consultez Collecter les données disponibles depuis un appareil mobile.

Pour plus d'informations sur l'inscription d'appareils, voir Qu'est-ce que l'inscription d'appareils ? dans la documentation Microsoft Intune.

Résolution des problèmes

En cas de problèmes lors de l'utilisation d' Autorité de certification privée AWS, consultez les rubriques suivantes.

Rubriques

Création et signature d'un certificat CA privé

Une fois que vous avez créé votre autorité de certification privée, vous devez récupérer la demande de signature de certificat et l'envoyer à une autorité de certification racine ou intermédiaire dans votre infrastructure X.509. Votre autorité de certification utilise la demande de signature de certificat pour créer votre certificat d'autorité de certification privée, puis signe le certificat avant de vous le renvoyer.

Il n'est malheureusement pas possible de fournir des conseils spécifiques sur les problèmes liés à la création et la signature de votre certificat d'autorité de certification privée. Les détails de votre infrastructure X.509 et de la hiérarchie de l'autorité de certification qu'elle contient dépassent le cadre de cette Autorité de certification privée AWS documentation.

Latence dans les réponses OCSP

La réactivité de l'OCSP peut être plus lente si l'appelant est géographiquement éloigné d'un cache périphérique régional ou de la région de l'autorité de certification émettrice. Pour plus d'informations sur la disponibilité du cache périphérique régional, consultez Global Edge Network. Nous recommandons de délivrer les certificats dans une région proche de celle où ils seront utilisés.

Configuration d'Amazon S3 pour autoriser la création d'un compartiment CRL

Il est possible que votre autorité de certification privée ne parvienne pas à créer un compartiment CRL si l'accès public (paramètres du compartiment) d'Amazon S3 est appliqué à votre compte. Vérifiez vos paramètres Amazon S3 dans ce cas. Pour plus d'informations, consultez Utilisation de la fonctionnalité de blocage de l'accès public Amazon S3.

Révocation d'un certificat CA autosigné

Vous ne pouvez pas révoquer un certificat d'une autorité de certification auto-signé. Au lieu de cela, vous devez supprimer l'autorité de certification.

Gestion des exceptions

Une Autorité de certification privée AWS commande peut échouer pour plusieurs raisons. Pour obtenir plus d'informations sur chaque exception ainsi que des recommandations pour les résoudre, veuillez consulter le tableau ci-dessous.

Autorité de certification privée AWS Exceptions

Exception renvoyée par Autorité de certification privée AWS	Description	Correction
`AccessDeniedException`	Les autorisations requises pour utiliser la commande donnée n'ont pas été déléguées par une autorité de certification privée au compte appelant.	Pour plus d'informations sur la délégation des autorisations dans Autorité de certification privée AWS, voirAttribuer des autorisations de renouvellement de certificat à ACM.
`InvalidArgsException`	Une demande de création ou de renouvellement de certificat a été effectuée avec des paramètres non valides.	Vérifiez la documentation individuelle de la commande pour vous assurer que vos paramètres d'entrée sont valides. Si vous créez un

Exception renvoyée par Autorité de certification privée AWS	Description	Correction
		nouveau certificat, assurez-vous que l'algorithme de signature demandé peut être utilisé avec le type de clé de l'autorité de certification.
`InvalidStateException`	L'autorité de certification privée associée ne peut pas renouveler le certificat, car son état n'est pas `ACTIVE`.	Essayez de restaurer l'autorité de certification privée. Si la période de restauration de l'autorité de certification privée n'est pas en cours, l'autorité de certification ne peut pas être restaurée et le certificat ne peut pas être renouvelé.
`LimitExceededException`	Chaque autorité de certification dispose d'un quota de certificats qu'elle peut délivrer. L'autorité de certification privée associée au certificat désigné a atteint son quota. Pour plus d'informations, consultez la section Service Quotas dans le Références générales AWS Guide.	Contactez le AWS Support Centre pour demander une augmentation de quota.
`MalformedCSRException`	La demande de signature de certificat qui a été envoyée à Autorité de certification privée AWS ne peut pas être vérifiée ou validée.	Vérifiez que votre demande de signature de certificat a été correctement générée et configurée.

Exception renvoyée par Autorité de certification privée AWS	Description	Correction
`OtherException`	Une erreur interne a provoqué l'échec de la demande.	Essayez à nouveau d'exécuter la commande. Si le problème persiste, contactez le AWS Support Centre.
`RequestFailedException`	Un problème réseau dans votre AWS environnement a entraîné l'échec de la demande.	Réitérez la requête . Si l'échec persiste, vérifiez la configuration de votre Amazon VPC (VPC).
`ResourceNotFoundException`	L'autorité de certification privée qui a émis le certificat a été supprimée et n'existe plus.	Demandez un nouveau certificat à une autre autorité de certification active.

Exception renvoyée par Autorité de certification privée AWS	Description	Correction
`ThrottlingException`	Une action d'API demandée a échoué car elle dépassait un quota.	Vérifiez que vous n'émettez pas plus d'appels que ce qui est autorisé Autorité de certification privée AWS. Une erreur `ThrottlingException` peut également se produire parce que vous avez rencontré une condition temporaire et non parce que vous avez dépassé un quota. Si vous rencontrez l'erreur et que vous n'avez pas effectué d'appels au-delà du quota, réessayez votre demande. Si vous vous heurtez à un quota, vous pouvez demander une augmentation. Pour plus d'informations, consultez la section Service Quotas dans le Références générales AWS Guide.

Exception renvoyée par Autorité de certification privée AWS	Description	Correction
`ValidationException`	Les paramètres d'entrée de la demande ont été mal formatés, ou la période de validité du certificat racine se termine avant la période de validité du certificat demandé.	Vérifiez les exigences de syntaxe des paramètres d'entrée de la commande ainsi que la période de validité du certificat racine de votre autorité de certification. Pour de plus amples informations sur la modification de la période de validité, veuillez consulter Mettre à jour votre CA privée.

Utilisation de la norme Matter

La norme de connectivité Matter spécifie les configurations de certificats qui améliorent la sécurité et la cohérence des appareils de l'Internet des objets (IoT). Des exemples Java permettant de créer des certificats d'autorité de certification racine, d'autorité de certification intermédiaire et d'entité finale conformes à Matter sont disponibles à l'adresse. Utilisation de l'Autorité de certification privée AWSAPI pour implémenter le standard Matter (exemples Java)

Pour faciliter le dépannage, les développeurs de Matter fournissent un outil de vérification des certificats appelé chip-cert. Les erreurs signalées par l'outil sont répertoriées dans le tableau suivant avec les correctifs.

Code d'erreur	Signification	Correction
0x00000035	`BasicConstraints KeyUsage`, et les `ExtensionKeyUsage` extensions doivent être marquées comme critiques.	Assurez-vous d'avoir sélectionné le bon modèle pour votre cas tion.

Code d'erreur	Signification	Correction
0x000000 0	L'extension de l'identifiant de clé d'autorité doit être présente.	Autorité de certification privée AWS ne définit pas l'extension de iant de clé d'autorité sur les certificats racine. Vous devez génér AuthorityKeyIdentifier valeur codée en Base64 à l'aide du CSR, transmettre à un. CustomExtension Pour plus d'informations, co Activez une autorité de certification racine pour les certificats op nels des nœuds (NOC). et Activer une autorité d'attestation de p (PAA).
0x000000 E	Le certificat est expiré.	Assurez-vous que le certificat que vous utilisez n'est pas expiré

Code d'erreur	Signification	Correction

0x000000
4
	Échec de validation de la chaîne de certificats.	Cette erreur peut se produire si vous tentez de créer un certifica finale conforme à Matter sans utiliser les exemples Java fournis utilisent l' Autorité de certification privée AWS API pour transme certificat correctement configuré. KeyUsage
		
		Par défaut, Autorité de certification privée AWS génère des vale KeyUsage extension de neuf bits, le neuvième bit générant un c supplémentaire. Matter ignore l'octet supplémentaire lors des co ns de format, ce qui entraîne des échecs de validation en chaîn Cependant, un CustomExtensiondans le `APIPassthrough` m peut être utilisé pour définir le nombre exact d'octets de la `KeyU` valeur. Pour obtenir un exemple, consultez Création d'un certifi opérationnel de nœud (NOC).
		
		Si vous modifiez l'exemple de code ou utilisez un autre utilitaire qu'OpenSSL, vous devez effectuer une vérification manuelle afi les erreurs de validation de chaîne.
		
		Pour vérifier que les conversions sont sans perte
		
		1. Utilisez openssl pour vérifier qu'un certificat, un certificat de (entité finale), contient une chaîne valide. Dans cet exemple `rcac.pem` il s'agit du certificat de l'autorité de certification r `icac.pem` du certificat de l'autorité de certification intermé `noc.pem` du certificat de nœud.
		
		`openssl verify -verbose -CAfile <(cat rcac.pem icac noc.pem`
		
		2. Utilisez chip-cert pour convertir le certificat de nœud au for au format TLV (tag, length, value) et vice versa.
		
		`./chip-cert convert-cert noc.pem noc.chip -c`

Code d'erreur	Signification	Correction
		`./chip-cert convert-cert noc.chip noc_converted.pem` Les fichiers `noc.pem` `noc_converted.pem` doivent être exactement les mêmes que ceux confirmés par un outil de comparaison de chaînes.

Connecteur pour les erreurs et les défaillances d'AD

Utilisez les informations fournies ici pour vous aider à diagnostiquer et à corriger les erreurs et les échecs de création lorsque vous travaillez avec Connector for AD.

Rubriques

Erreurs

Connector for AD envoie des messages d'erreur pour plusieurs raisons. Pour obtenir des informations sur chaque erreur et des recommandations pour les résoudre, consultez le tableau suivant. Vous pouvez recevoir ces erreurs en vous abonnant aux événements Amazon EventBridge Scheduler (source de l'événement :`aws.pca-connector-ad`) ou en utilisant l'inscription manuelle sous Windows.

Code d'erreur	Signification	Correction
0x8FFFA000	L'authentification Kerberos a échoué.	Si vous utilisez l'inscription automatique, corrigez le principal de votre service de AWS ressources. Si vous utilisez l'interface utilisateur d'Active

Code d'erreur	Signification	Correction
		Directory pour obtenir un certificat, `gpupdate /force` lancez-le.
0x8FFFA001	Le message SOAP doit contenir un en-tête d'action.	Ajoutez un en-tête d'action.
0x8FFFA002	Le connecteur n'a pas accès à l'autorité de certification privée à laquelle il est connecté.	Partagez votre autorité de certification privée avec le connecteur en créant un AWS Resource Access Manager (RAM) à partager entre votre autorité de certification privée et le service Connector for AD.
0x8FFFA003	L'autorité de certification privée pour ce connecteur n'est pas active.	Déplacez l'autorité de certification privée à l'état actif. Si le certificat de votre autorité de certification privée est en attente, installez-le.
0x8FFFA004	L'autorité de certification privée pour ce connecteur n'existe pas.	Passez votre autorité de certification à l'état Actif si elle est à l'état Supprimé. Si votre autorité de certification privée est définitivement supprimée, créez un nouveau connecteur avec une autre autorité de certification.
0x8FFFA005	Le modèle a spécifié l'`directoryGuid` attribut pour le sujet du certificat ou le nom alternatif du sujet, mais l'attribut n'a pas été trouvé dans l'objet AD pour le demandeur.	Active Directory n'a pas généré de `directoryGuid` fichier pour votre annuaire. Résoudre les problèmes dans Active Directory.

Code d'erreur	Signification	Correction
0x8FFFA006	Le modèle a spécifié l'`dnsHostName` attribut pour le sujet du certificat ou le nom alternatif du sujet, mais l'attribut n'a pas été trouvé dans l'objet AD pour le demandeur.	Ajoutez l'`dnsHostName` attribut à votre objet AD.
0x8FFFA007	Le modèle spécifiait l'attribut d'e-mail à inclure dans l'objet du certificat ou dans le nom alternatif du sujet, mais l'attribut n'a pas été trouvé dans l'objet AD du demandeur.	Ajoutez l'attribut e-mail à votre objet AD
0x8FFFA008	Le message SOAP doit avoir un en-tête d'action correspondant à l'un `http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy/IPolicy/GetPolicies` ou à l'autre`http://schemas.microsoft.com/windows/pki/2009/01/enrollment/RST/wstep` .	Mettez à jour l'en-tête de l'action pour utiliser l'une des valeurs spécifiées.
0x8FFFA009	Le BinarySecurityToken doit être encodé dans`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary` .	Mettez à jour le type de jeton de sécurité binaire.

Code d'erreur	Signification	Correction
0x8FFFA00A	Le BinarySecurityToken n'est pas valide.	Vérifiez que le CSR est correctement généré.
0x8FFFA00B	Le type de valeur BinarySecurityToken doit être `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#PKCS7ouhttp://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10` .	Mettez à jour le type de valeur du jeton de sécurité binaire avec une valeur valide.
0x8FFFA00C	Le CMS non valide BinarySecurityToken contenu.	Le Base64 est valide mais la syntaxe des messages cryptographiques (CMS) n'est pas valide. Passez en revue la syntaxe du CMS.
0x8FFFA00D	Ils BinarySecurityToken contenaient un CSR non valide.	Vérifiez que le CSR a été correctement généré.
0x8FFFA00E	L'autorité de certification privée n'a pas pu émettre de certificat à l'aide du modèle spécifique.	Consultez l'exception de validation de AWS Private CA. Vous pouvez consulter l'exception de validation sur Amazon EventBridge ou AWS CloudTrail.
0x8FFFA00F	Le message SOAP doit avoir un type de demande de`http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .	Définissez le type de demande sur`http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .

Code d'erreur	Signification	Correction
0x8FFFA010	Le message SOAP doit comporter un en-tête to correspondant au `CertificateEnrollmentPolicyServerEndpoint` champ du connecteur ou au champ URI de la réponse XCEP.	Définissez l'en-tête du jeton de sécurité de la demande `CertificateEnrollmentPolicyServerEndpoint` sur le champ ou sur le champ URI de la réponse XCEP.
0x8FFFA011	Le message SOAP ne doit comporter qu'un seul en-tête d'action.	Vérifiez l'en-tête du message SOAP du jeton de sécurité de la demande et définissez correctement l'en-tête.
0x8FFFA012	Le message SOAP ne doit comporter qu'un seul `messageId` en-tête.	Vérifiez l'en-tête du message SOAP du jeton de sécurité de la demande et définissez correctement l'en-tête.
0x8FFFA013	Le message SOAP ne doit comporter qu'un seul en-tête to.	Vérifiez l'en-tête du message SOAP du jeton de sécurité de la demande et définissez correctement l'en-tête.
0x8FFFA014	Le demandeur n'a pas accès au modèle demandé.	Autorisez le groupe du demandeur à s'inscrire à l'aide du modèle demandé en créant une entrée de contrôle d'accès.
0x8FFFA015	L'extension `CertificateTemplateInformation` ou l'`CertificateTemplateName` extension doivent être présentes dans le BinarySecurityToken.	Ajoutez l'extension de sécurité à votre CSR.

Code d'erreur	Signification	Correction
0x8FFFA016	Le modèle demandé n'a pas été trouvé pour le connecteur donné.	Les modèles sont des ressources secondaires pour chaque connecteur. Créez le modèle pour le connecteur à l'aide de `createTemplate` .
0x8FFFA017	La demande a été refusée suite à une limitation des demandes.	Ralentissez le taux de demandes.
0x8FFFA018	Le message SOAP doit contenir un `to` en-tête.	Vérifiez l'en-tête du message SOAP.
0x8FFFA019	Impossible de traiter le message SOAP en raison d'un en-tête non reconnu.	Vérifiez l'en-tête du message SOAP.
0x8FFFA01A	Le modèle spécifiait l'attribut UPN à inclure dans l'objet du certificat ou dans le nom alternatif du sujet, mais l'attribut n'a pas été trouvé dans l'objet AD du demandeur.	Ajoutez un UPN à l'objet Active Directory.

Défaillances de création de connecteurs

La création d'un connecteur peut échouer pour diverses raisons. Lorsque la création du connecteur échoue, vous recevrez la raison de l'échec dans la réponse de l'API. Si vous utilisez la console, la raison de l'échec est affichée sur la page de détails du connecteur, sous le champ Détails de statut supplémentaires dans le conteneur des détails du connecteur. Le tableau suivant décrit les raisons de l'échec et les étapes recommandées pour le résoudre.

État de défaillance	Description	Correction
`CA_CERTIFICATE_REGISTRATION_FAILED`	Connector for AD n'est pas en mesure d'importer des certificats CA dans votre répertoire.	Consultez la page Conditions préalables et vérifiez que votre compte de service dispose des autorisations appropriées. Après avoir délégué les autorisations appropriées à votre compte de service, supprimez le connecteur défaillant et créez-en un nouveau. Pour plus d'informations sur la délégation d'autorisations, voir Déléguer des privilèges à votre compte de service dans le Guide d'AWS Directory Service administration.
`DIRECTORY_ACCESS_DENIED`	Connector for AD ne parvient pas à accéder à votre répertoire.	Vous devez autoriser Connector for AD à accéder à votre annuaire. Consultez la Stratégie IAM section pour vous assurer que la politique IAM associée à votre AWS compte vous permet d'accéder aux annuaires et de les décrire. Après avoir accordé les autorisations appropriées à votre AWS rôle, supprimez le connecteur défaillant et créez-en un nouveau.
`INTERNAL_FAILURE`	Connector for AD a connu une défaillance interne.	Réessayez ultérieurement. Supprimez le connecteu

État de défaillance	Description	Correction
		r défaillant et créez-en un nouveau.
PRIVATECA_ACCESS_DENIED	Connector for AD ne parvient pas à accéder à votre autorité de certification privée.	Consultez la page Conditions préalables et vérifiez que vous êtes autorisé à créer un connecteur. Pour plus d'informations, consultez Stratégie IAM. Si vous créez un connecteur par le biais AWS CLI d'une API, consultez la page Conditions préalables et vérifiez que vous avez partagé l'autorité de certification privée avec Connector for AD à l'aide AWS Resource Access Manager de Connector for AD. Après avoir vérifié et corrigé les autorisations IAM et le partage AWS RAM des ressources, supprimez le connecteur défaillant et créez-en un nouveau.
PRIVATECA_RESOURCE_NOT_FOUND	Connector for AD ne trouve pas l'autorité de certification privée spécifiée.	Assurez-vous de spécifier le nom de ressource Amazon (ARN) CA privé correct, puis supprimez le connecteur défaillant et créez-en un nouveau en utilisant l'ARN CA privé que vous avez prévu.

État de défaillance	Description	Correction
`SECURITY_GROUP_NOT_IN_VPC`	Le groupe de sécurité ne se trouve pas dans le VPC qui héberge votre répertoire.	Utilisez un groupe de sécurité situé dans le VPC qui héberge votre répertoire. Pour plus d'informations, consultez Groupes de sécurité. Supprimez le connecteur défaillant et créez-en un nouveau avec un groupe de sécurité situé dans le VPC.
`VPC_ACCESS_DENIED`	Connector for AD ne peut pas accéder au VPC Amazon qui héberge votre annuaire.	Vérifiez vos autorisations IAM. Supprimez le connecteur défaillant et créez-en un nouveau. Pour un exemple de politique IAM incluant des autorisations d'accès, voir Stratégie IAM
`VPC_ENDPOINT_LIMIT_EXCEEDED`	Connector for AD ne peut pas créer de point de terminaison dans votre Amazon VPC. Vous avez atteint le nombre limite de points de terminaison VPC que vous pouvez créer pour votre compte.	Supprimez les points de terminaison Amazon VPC ou demandez une augmentation de limite. Une fois que vous avez effectué l'une des deux étapes, supprimez le connecteur défaillant et créez-en un nouveau. Pour plus d'informations sur les quotas, consultez la section Quotas d'Amazon Virtual Private Cloud Service.

État de défaillance	Description	Correction
VPC_RESOURCE_NOT_FOUND	Connector for AD ne trouve pas le VPC spécifié.	Assurez-vous que vous avez spécifié le VPC correct et que celui-ci existe. Supprimez ensuite le connecteur défaillant et créez-en un nouveau en utilisant le VPC ID correct.

Échecs de création de SPN

La création du nom principal du service (SPN) peut échouer pour diverses raisons. Lorsque la création du SPN échoue, vous recevrez la raison de l'échec dans la réponse de l'API. Si vous utilisez la console, la raison de l'échec est affichée sur la page de détails du connecteur, sous le champ Détails de statut supplémentaires dans le conteneur du nom principal du service (SPN). Le tableau suivant décrit les raisons de l'échec et les étapes recommandées pour le résoudre.

État de défaillance	Description	Correction
DIRECTORY_ACCESS_DENIED	Connector for AD ne peut pas accéder à votre annuaire.	Accordez à Connector for AD l'accès à votre annuaire. Pour un exemple de politique IAM incluant des autorisations permettant d'accéder à un annuaire, consultezStratégie IAM.
DIRECTORY_NOT_REACHABLE	Connector for AD ne peut pas accéder à votre annuaire.	Vérifiez le réseau entre AWS et votre répertoire, puis réessayez de créer un SPN.
DIRECTORY_RESOURCE_NOT_FOUND	Connector for AD ne trouve pas le répertoire spécifié.	Assurez-vous de spécifier l'ID de répertoire correct, puis supprimez le connecteu

État de défaillance	Description	Correction
		r défaillant et créez-en un nouveau en utilisant l'ID de répertoire prévu.
`INTERNAL_FAILURE`	Connector for AD a connu une défaillance interne.	Réessayez ultérieurement.
`SPN_EXISTS_ON_DIFFERENT_AD_OBJECT`	Le nom principal du service (SPN) existe sur un autre objet Active Directory.	Supprimez le SPN de l'objet Active Directory, puis réessayez de le créer.
`SPN_LIMIT_EXCEEDED`	Connector for AD ne peut pas créer le SPN car vous avez atteint la limite de SPN par répertoire. Le nombre maximum de SPN par répertoire est de 10.	Supprimez un ou plusieurs SPN de votre compte, puis réessayez de créer le SPN.

Erreurs d'échec de création du connecteur pour AD

La création d'un connecteur pour AD peut échouer pour diverses raisons. Lorsque la création du connecteur échoue, vous recevrez la raison de l'échec dans la réponse de l'API. Si vous utilisez la console, la raison de l'échec est affichée sur la page de détails du connecteur, sous le champ Détails de statut supplémentaires du conteneur des détails du connecteur. Le tableau suivant décrit les raisons de l'échec et les étapes recommandées pour le résoudre.

Termes et concepts

Les termes et les concepts suivants peuvent vous aider lorsque vous utilisez AWS Private Certificate Authority (Autorité de certification privée AWS).

Rubriques

Approbation

Pour permettre à un navigateur Web d'approuver l'identité d'un site Web, le navigateur doit être en mesure de vérifier le certificat de ce site. Toutefois, les navigateurs n'approuvent qu'un petit nombre de certificats appelés certificats d'autorité de certification racine. Un tiers de confiance, appelé autorité de certification (CA), valide l'identité du site Web et émet un certificat numérique signé pour l'opérateur du site Web. Le navigateur peut ensuite vérifier la signature numérique afin de valider l'identité du site Web. Si la validation aboutit, le navigateur affiche une icône de verrouillage dans la barre d'adresse.

Certificats de serveur TLS

Les transactions HTTPS requièrent des certificats de serveur pour authentifier un serveur. Un certificat de serveur est une structure de données X.509 v3 qui lie la clé publique figurant dans le certificat à l'objet du certificat. Un certificat TLS est signé par une autorité de certification (CA). Il

contient le nom du serveur, la période de validité, la clé publique l'algorithme de signature, ainsi que d'autres données.

Signature du certificat

Une signature numérique est un hachage chiffré sur un certificat. Une signature est utilisée pour confirmer l'intégrité des données du certificat. Votre autorité de certification privée crée une signature à l'aide d'une fonction de hachage de chiffrement telle que SHA256 sur le contenu du certificat de taille variable. Cette fonction de hachage produit une chaîne de données de taille fixe qu'il n'est, de fait, pas possible de falsifier. Cette chaîne est appelée un hachage. L'autorité de certification chiffre ensuite la valeur de hachage avec sa clé privée et concatène le hachage chiffré avec le certificat.

Autorité de certification

Une autorité de certification émet et, si nécessaire, révoque des certificats numériques. Le type le plus courant de certificat repose sur la norme ISO X.509. Un certificat X.509 confirme l'identité de l'objet du certificat et lie cette identité à une clé publique. L'objet peut être un utilisateur, une application, un ordinateur ou un autre appareil. L'autorité de certification signe un certificat en hachant le contenu, puis en chiffrant le hachage avec la clé privée associée à la clé publique du certificat. Une application cliente, par exemple un navigateur web qui doit confirmer l'identité d'un objet, utilise la clé publique pour déchiffrer la signature du certificat. Ensuite, elle hache le contenu du certificat et compare la valeur hachée à la signature déchiffrée pour déterminer si elles correspondent. Pour plus d'informations sur la signature des certificats, consultez Signature du certificat.

Vous pouvez utiliser Autorité de certification privée AWS pour créer une autorité de certification privée et utiliser l'autorité de certification privée pour émettre des certificats. Votre autorité de certification privée émet uniquement des certificats SSL/TLS privés à utiliser au sein de votre organisation. Pour plus d'informations, consultez Certificat privé. Votre autorité de certification privée nécessite également un certificat avant que vous puissiez l'utiliser. Pour plus d'informations, consultez Certificat CA.

Root CA

Bloc de construction cryptographique et source de confiance à parti desquels des certificats peuvent être émis. Elle est composée d'une clé privée pour la signature (émission) de certificats et d'un certificat racine qui identifie l'autorité de certification racine et lie la clé privée au nom de l'autorité

de certification. Le certificat racine est distribué aux magasins de confiance de chaque entité d'un environnement. Les administrateurs construisent des magasins d'approbation pour inclure uniquement les autorités de certification qu'ils approuvent. Les administrateurs mettent à jour ou intègrent les magasins de confiance dans les systèmes d'exploitation, les instances et les images des machines hôtes des entités de leur environnement. Lorsque des ressources tentent de se connecter les unes aux autres, elles vérifient les certificats que présente chaque entité. Un client vérifie la validité des certificats et vérifie si une chaîne existe du certificat vers un certificat racine installé dans le magasin d'approbation. Si ces conditions sont remplies, une « poignée de main » est établie entre les ressources. Cette liaison prouve à l'aide d'un chiffrement l'identité de chaque entité à l'autre et crée un canal de communication chiffré (TLS/SSL) entre elles.

Certificat CA

Le certificat d'une autorité de certification confirme l'identité de l'autorité de certification et la lie à la clé publique figurant dans le certificat.

Vous pouvez utiliser Autorité de certification privée AWS pour créer une autorité de certification racine privée ou une autorité de certification subordonnée privée, chacune basée sur un certificat d'une autorité de certification. Les certificats d'une autorité de certification subordonnée sont signés par un autre certificat d'une autorité de certification supérieur dans une chaîne de confiance. Cependant, le certificat est auto-signé pour les autorités de certification racines. Vous pouvez également établir une autorité racine externe (hébergée sur site, par exemple). Vous pouvez ensuite utiliser votre autorité racine pour signer un certificat d'une autorité de certification racine subordonnée hébergé par Autorité de certification privée AWS.

L'exemple suivant montre les champs classiques que comporte un certificat d'une autorité de certification X.509 Autorité de certification privée AWS. Notez que pour le certificat d'une autorité de certification, la valeur `CA:` du champ `Basic Constraints` est définie sur `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
```

Certificat racine de l'autorité de certification

Une autorité de certification (CA) existe généralement au sein d'une structure hiérarchique qui contient plusieurs autres autorités de certification avec des relations parent—enfant clairement définies entre elles. Les autorités de certification subordonnées sont certifiées par leurs autorités de certification parent, ce qui crée une chaîne de certificats. L'autorité de certification située en haut de la hiérarchie est appelée l'autorité de certification racine, et son certificat est appelé le certificat racine. En général, ce certificat est auto-signé.

Certificat d'entité finale

Un certificat d'entité finale identifie une ressource, telle qu'un serveur, une instance, un conteneur ou un appareil. Contrairement aux certificats CA, les certificats d'entité finale ne peuvent pas être utilisés pour émettre des certificats. Les autres termes courants pour désigner le certificat d'entité finale sont le certificat « client » ou le certificat « feuille ».

Certificats auto-signés

Certificat signé par l'émetteur au lieu d'une autorité de certification supérieure. Contrairement aux certificats émis à partir d'une racine sécurisée gérée par une autorité de certification, les certificats auto-signés agissent comme leur propre racine et présentent donc des limites importantes : ils peuvent être utilisés pour fournir un chiffrement filaire mais pas pour vérifier l'identité, et ils ne peuvent pas être révoqués. Du point de vue de la sécurité, ils ne sont pas acceptables. Cependant, les organisations les utilisent quand même, car ils sont faciles à générer, qu'ils ne nécessitent aucune expertise ni infrastructure, et que de nombreuses applications les acceptent. Aucun contrôle n'est en place pour l'émission de certificats auto-signés. Les organisations qui les utilisent encourent de plus grands risques de panne étant donné qu'elles n'ont aucun moyen d'effectuer le suivi des dates d'expiration des certificats.

Certificat privé

Autorité de certification privée AWSles certificats sont des certificats SSL/TLS privés que vous pouvez utiliser au sein de votre organisation, mais qui ne sont pas fiables sur l'Internet public. Utilisez-les pour identifier les ressources telles que les clients, les serveurs, les applications, les services, les appareils et les utilisateurs. Lorsque vous établissez un canal de communication chiffré sécurisé, chaque ressource utilise un certificat similaire à l'exemple suivant, ainsi que des techniques cryptographiques pour prouver son identité à une autre ressource. Les points de terminaison des API internes, les serveurs web, les utilisateurs VPN, les appareils IoT et de nombreuses autres applications utilisent des certificats privés pour établir des canaux de communication chiffrés nécessaires pour fonctionner en toute sécurité. Par défaut, les certificats privés ne sont pas approuvés publiquement. Un administrateur interne doit explicitement configurer des applications pour approuver les certificats privés et distribuer les certificats.

```
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
```

Chemin du certificat

Un client qui s'appuie sur un certificat valide l'existence d'un chemin entre le certificat d'entité finale, éventuellement via une chaîne de certificats intermédiaires, et une racine fiable. Le client vérifie que chaque certificat le long du chemin est valide (non révoqué). Il vérifie également que le certificat d'entité finale n'a pas expiré, qu'il est intègre (qu'il n'a pas été falsifié ou modifié) et que les contraintes du certificat sont appliquées.

Contrainte de longueur de chemin

Les contraintes de base pathLenConstraintd'un certificat CA définissent le nombre de certificats CA subordonnés qui peuvent exister dans la chaîne située en dessous. Par exemple, un certificat CA avec une contrainte de longueur de chemin de zéro ne peut pas avoir d'autorité de certification

subordonnée. Une autorité de certification avec une valeur un pour la contrainte de longueur de chemin peut avoir jusqu'à un niveau d'autorités de certification subordonnées sous elle. La RFC 5280 définit cela comme « le nombre maximum de certificats non-self-issued intermédiaires qui peuvent suivre ce certificat dans un chemin de certification valide ». La valeur de longueur du chemin exclut le certificat d'entité finale, bien que le langage informel concernant la « longueur » ou la « profondeur » d'une chaîne de validation puisse l'inclure... ce qui prête à confusion.

Historique du document

Le tableau suivant décrit les modifications importantes apportées à cette documentation depuis janvier 2018. En plus des principales modifications répertoriées ici, nous mettons fréquemment à jour la documentation pour améliorer les descriptions et les exemples, et pour répondre aux commentaires que vous nous envoyez. Pour recevoir une notification concernant les modifications importantes, utilisez le lien figurant dans le coin supérieur droit pour vous abonner aux flux RSS.

Modification	Description	Date
AWS Private CA supporte désormais Connector for SCEP (version préliminaire)	Utilisez Connector for SCEP pour établir un lien avec vos clients AWS Private CA et appareils compatibles avec le protocole SCEP.	11 juin 2024
Nouveaux conseils de dépannage des connecteurs	Ajout de deux nouvelles sections sur le dépannage des échecs de création de connecteurs et de SPN.	4 avril 2024
Ajout de l'extension CDP pour Matter	Ajoute la prise en charge de l'extension CDP (Certificate Revocation List Distribution Point) pour Matter.	25 janvier 2024
AWS Private CA Support de l'API pour MdL	Ajout du support API pour la création de certificats conformes à la norme ISO/IEC pour le permis de conduire mobile (MdL).	16 janvier 2024
AWS Private CA Connecteur pour Active Directory	Guide de l'utilisateur, API et support CLI pour Connector for AD. Pour plus d'informations, consultez la documentation du Connector for AD.	24 août 2023

Modification des noms des politiques de sécurité pour qu'ils correspondent au nouveau nom de service	Adoption de nouveaux noms pour les politiques IAM AWS gérées qui spécifient des autorisations standard sur Autorité de certification privée AWS. Pour plus d'informations, consultez la section Politiques AWS gérées.	13 février 2023
Ajout d'un outil de suivi des modifications pour les politiques AWS gérées	Documentation ajoutée pour suivre les modifications apportées aux politiques IAM AWS gérées qui spécifient des autorisations standard sur Autorité de certification privée AWS. Pour plus d'informations, consultez la section Mises à jour des politiques AWS gérées pour Autorité de certification privée AWS.	11 novembre 2022
Support de l'API et de la CLI pour les autorités de certification qui émettent des certificats de courte durée	Avec l'introduction des modes d'utilisation de l'autorité de certification, une autorité de certification peut être configurée pour émettre des certificats à usage général ou exclusivement de courte durée. Pour plus d'informations, consultez la section Modes d'autorité de certification.	24 octobre 2022

Changement de marque du service et mise à jour de la console	Le service est renommé en AWS Private Certificate Authority (Autorité de certification privée AWS). La Autorité de certification privée AWS console bénéficie d'améliorations en termes d'utilisabilité, notamment des panneaux d'aide intégrés qui renvoient à une documentation complète.	27 septembre 2022
Support de certificats conforme à la norme MATTER	Trois nouveaux modèles de certificats ajoutent la prise en charge des certificats d'autorité de certification et d'entité finale conformes à MATTER. Pour plus d'informations, consultez la section Comprendre les modèles de certificats.	20 juillet 2022
Prise en charge d'une nouvelle région	Point de terminaison ajouté pour l'Asie-Pacifique (Jakarta). Pour une liste complète des points de AWS Private CA terminaison, voir Points de terminaison et quotas de l'autorité de certification privée ACM.	4 mai 2022
Support pour les attributs et extensions personnalisés	Utilisez l'CustomAttributeobjet pour configurer des CA et des certificats personnalisés, et l'CustomExtension objet pour configurer des certificats personnalisés.	16 mars 2022

Support pour les OCSP gérés	Consultez la section Configuration d'une méthode de révocation de certificat pour les options de révocation, y compris l'OCSP.	18 août 2021
Support de la fonctionnalité S3 Block Public Access pour les CRL	Voir Activation de la fonctionnalité S3 Block Public Access.	27 mai 2021
Exemples d'implémentation Java nouveaux et mis à jour	Consultez la section Utilisation de l'API ACM Private CA (exemples Java).	9 septembre 2020
Prise en charge d'une nouvelle région	Points de terminaison ajoutés pour l'Afrique (Le Cap) et l'Europe (Milan). Pour une liste complète des points de Autorité de certification privée AWS terminaison, voir Points de terminaison et quotas des autorités de certification AWS Certificate Manager privées.	27 août 2020
Accès CA privé entre comptes pris en charge	AWS Certificate Manager les utilisateurs peuvent être autorisés à émettre des certificats à l'aide d'autorités de certification privées dont ils ne sont pas propriétaires. Pour plus d'informations, consultez la section Accès entre comptes aux autorités de certification privées.	17 août 2020

Support des points de terminaison VPC () PrivateLink	Ajout de la prise en charge de l'utilisation des points de terminaison VPC (AWS PrivateLink) pour améliorer la sécurité du réseau. Pour plus d'informations, consultez ACM Private CA VPC AWS PrivateLink Endpoints ().	26 mars 2020
Ajout d'une section dédiée à la sécurité	La documentation de sécurité pour AWS a été consolidée dans une section dédiée à la sécurité. Pour plus d'informations sur la sécurité, consultez la section Sécurité dans AWS Certificate Manager une autorité de certification privée.	26 mars 2020
Modèle ARN ajouté aux rapports d'audit.	Pour de plus amples informations, veuillez consulter Création d'un rapport d'audit pour votre autorité de certification privée.	6 mars 2020
CloudFormation soutien	Support ajouté pour AWS CloudFormation. Pour de plus amples informations, veuillez consulter le document Référence du type de ressource ACMPCA dans le Guide de l'utilisateur AWS CloudFormation .	22 janvier 2020

CloudWatch Intégration des événements	Intégration à CloudWatch Events pour les événement s asynchrones, notamment la création d'une autorité de certification, l'émission de certificats et la création de CRL. Pour plus d'informations, consultez la section Utilisation CloudWatch des événements.	23 décembre 2019
Points de terminaison FIPS	Des points de terminaison FIPS ont été ajoutés pour AWS GovCloud (USA Est) et AWS GovCloud (USA Ouest). Pour une liste complète des points de Autorité de certifica tion privée AWS terminais on, voir Points de terminais on et quotas des autorités de certification AWS Certificate Manager privées.	13 décembre 2019
Autorisations basées sur des balises	Autorisations basées sur des balises prises en charge à l'aide des nouvelles API `TagResource` , `UntagReso urce` et `ListTagsF orResource` . Pour de plus amples informations sur les contrôles basés sur des balises, veuillez consultez Contrôle de l'accès à et pour les utilisateurs et les rôles IAM à l'aide de balises de ressources IAM.	5 novembre 2019

Application des contraintes de nom	Ajout de la prise en charge de l'application des contraintes de nom d'objet sur les certificats d'une autorité de certification importés. Pour de plus amples informations, veuillez consulter Application de contraintes de nom sur une autorité de certification privée.	28 octobre 2019
Nouveaux modèles de certificats	De nouveaux modèles de certificats ont été ajoutés, y compris des modèles pour la signature de code avec AWS Signer. Pour de plus amples informations, veuillez consulter Utilisation de modèles.	1 octobre 2019
Planification de votre CA	Nouvelle section ajoutée sur la planification de votre PKI à l'aide de Autorité de certification privée AWS. Pour de plus amples informations, veuillez consulter Planification du déploiement de votre autorité de certification privée ACM.	30 septembre 2019
Prise en charge de régions supplémentaires	Ajout du support régional pour la région AWS Asie-Pacifique (Hong Kong). Pour obtenir la liste complète des régions prises en charge, consultez la section Points de terminaison et quotas des autorités de certification AWS Certificate Manager privées.	24 juillet 2019

Ajout d'une prise en charge complète de la hiérarchie privée des CA	Support pour la création et l'hébergement d'autorités de certification racines, il n'est plus nécessaire d'avoir recours à un parent externe.	20 juin 2019
Prise en charge de régions supplémentaires	Ajout du support régional pour les régions AWS GovCloud (ouest des États-Unis et est des États-Unis). Pour obtenir la liste complète des régions prises en charge, consultez AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	8 mai 2019
Prise en charge de régions supplémentaires	Ajout du support régional pour les régions AWS Asie-Pacifique (Mumbai et Séoul), USA Ouest (Californie du Nord) et UE (Paris et Stockholm). Pour obtenir la liste complète des régions prises en charge, consultez la section Points de terminaison et quotas des autorités de certification AWS Certificate Manager privées.	4 avril 2019
Processus de renouvellement des certificats de test	Les clients peuvent désormais tester manuellement la configuration de leur flux de travail de renouvellement géré ACM. Pour plus d'informations, consultez Test de la configuration de renouvellement géré ACM.	14 mars 2019

Prise en charge de régions supplémentaires	Ajout du support régional pour la région AWS UE (Londres). Pour obtenir la liste complète des régions prises en charge, consultez AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	1er août 2018
Restaurer les autorités de certification supprimées	La restauration des autorités de certification privées permet aux clients de restaurer les autorités de certification pendant une durée maximale de 30 jours à compter de leur suppression. Pour de plus amples informations, veuillez consulter Restauration de votre autorité de certification privée.	le 20 juin 2018

Mises à jour antérieures

Le tableau suivant décrit l'historique des publications de documentation AWS Private Certificate Authority antérieures à juin 2018.

Modification	Description	Date
Nouveau guide	Cette version présente AWS Private Certificate Authority.	04 avril 2018

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.