aws

User Guide

AWS Private Certificate Authority

Version latest

AWS Private Certificate Authority: User Guide

Table of Contents

Was ist AWS Private CA?

AWS Private CA ermöglicht die Erstellung von Hierarchien privater Zertifizierungsstellen (CA), einschließlich Stamm- und untergeordneter Zertifizierungsstellen CAs, ohne dass die Investitions- und Wartungskosten für den Betrieb einer lokalen Zertifizierungsstelle anfallen. Ihre private Firma CAs kann X.509-Zertifikate für Endeinheiten ausstellen, die unter anderem in folgenden Szenarien nützlich sind:

- Erstellen verschlüsselter TLS-Kommunikationskanäle
- Authentifizieren von Benutzern, Computern, API-Endpunkten und IoT-Geräten
- Kryptografische Code-Signatur
- Implementieren des Online Certificate Status Protocol (OCSP) zum Abrufen des Zertifikatswiderrufungsstatus

AWS Private CA Auf Operationen kann über die AWS-Managementkonsole, über die AWS Private CA API oder über die zugegriffen werden. AWS CLI

Topics

Regionale Verfügbarkeit für AWS Private Certificate Authority

Wie bei den meisten AWS Ressourcen handelt es sich auch bei privaten Zertifizierungsstellen (CAs) um regionale Ressourcen. Um private CAs in mehr als einer Region verwenden zu können, müssen Sie Ihre CAs in diesen Regionen erstellen. Private Daten können nicht CAs zwischen Regionen kopiert werden. Besuchen Sie AWS Regionen und -Endpunkte in der Allgemeine AWS-Referenz oder dieTabelle der AWS -Regionen, um mehr über die regionale Verfügbarkeit für AWS Private CA zu erfahren.

> **Note**
>
> ACM ist derzeit in einigen Regionen verfügbar, in denen dies nicht der AWS Private CA Fall ist.

Dienste integriert mit AWS Private Certificate Authority

Wenn Sie AWS Certificate Manager ein privates Zertifikat anfordern, können Sie dieses Zertifikat mit jedem Dienst verknüpfen, der in ACM integriert ist. Dies gilt sowohl für Zertifikate, die mit einem AWS Private CA Stamm verkettet sind, als auch für Zertifikate, die mit einem externen Stamm verkettet sind. Weitere Informationen finden Sie im AWS Certificate Manager Benutzerhandbuch unter Integrierte Dienste.

Sie können Private auch CAs in Amazon Elastic Kubernetes Service integrieren, um die Ausstellung von Zertifikaten innerhalb eines Kubernetes-Clusters zu ermöglichen. Weitere Informationen finden Sie unter Kubernetes sichern mit AWS Private Certificate Authority.

> **Note**
>
> Amazon Elastic Kubernetes Service ist kein integrierter ACM-Service.

Wenn Sie die AWS Private CA API verwenden, ein Zertifikat AWS CLI ausstellen oder ein privates Zertifikat aus ACM exportieren, können Sie das Zertifikat an einem beliebigen Ort installieren.

Unterstützte kryptografische Algorithmen in AWS Private Certificate Authority

AWS Private CA unterstützt die folgenden kryptografischen Algorithmen für die Generierung privater Schlüssel und das Signieren von Zertifikaten.

Unterstützter Algorithmus

Algorithmen mit privaten Schlüsseln	Signaturalgorithmen
ML_DSA_44	ML_DSA_44
ML_DSA_65	ML_DSA_65

Algorithmen mit privaten Schlüsseln	Signaturalgorithmen
ML_DSA_87	ML_DSA_87
RSA_2048	SHA256MIT RSA
RSA_3072	SHA384MIT RSA
RSA_4096	SHA512MIT RSA
EC_Prime256V1	SHA256MIT ECDSA
EC_SECP384R1	SHA384MIT ECDSA
EC_SECP521R1	SHA512MIT ECDSA
SM2 (nur Regionen China)	SM3WITHSM2

Diese Liste gilt nur für Zertifikate, die direkt AWS Private CA über die Konsole, API oder Befehlszeile ausgestellt wurden. Wenn AWS Certificate Manager Zertifikate mithilfe einer Zertifizierungsstelle von ausgestellt werden AWS Private CA, werden einige, aber nicht alle dieser Algorithmen unterstützt. Weitere Informationen finden Sie unter Anfordern eines privaten Zertifikats im AWS Certificate Manager Benutzerhandbuch.

> **Note**
>
> Für RSA oder ECDSA muss die angegebene Signaturalgorithmusfamilie mit der Schlüsselalgorithmusfamilie des privaten Schlüssels der CA übereinstimmen.
> Für ML-DSA ist die Hash-Funktion als Teil des Algorithmus selbst definiert. Bei ML-DSA gibt es keine Möglichkeit, eine andere Hash-Funktion auszuwählen. Um die Abwärtskompatibilität mit dem aufrechtzuerhalten APIs, wird derselbe Wert für den Schlüsselalgorithmus und den Signaturalgorithmus verwendet.

RFC 5280-Konformität in AWS Private Certificate Authority

AWS Private CA erzwingt bestimmte in RFC 5280 definierte Einschränkungen nicht. Umgekehrt gilt dies auch: Bestimmte zusätzliche Einschränkungen, die für eine private Zertifizierungsstelle geeignet sind, werden erzwungen.

Erzwungen

- Not After date (Nicht-nach-Datum). Gemäß RFC 5280 verhindert AWS Private CA die Ausstellung von Zertifikaten, deren `Not After`-Datum später als das `Not After`-Datum des Zertifikats der ausstellenden CA ist.
- Grundlegende Einschränkungen. AWS Private CA erzwingt grundlegende Einschränkungen und die Pfadlänge in importierten CA-Zertifikaten.

  Grundlegende Einschränkungen geben an, ob es sich bei der durch das Zertifikat identifizierten Ressource um eine Zertifizierungsstelle handelt und ob diese Zertifikate ausstellen kann. In AWS Private CA importierte CA-Zertifikate müssen die Erweiterung für grundlegende Einschränkungen enthalten, und die Erweiterung muss markiert sein `critical`. Zusätzlich zum `critical` Flag `CA=true` muss es gesetzt werden. AWS Private CA erzwingt grundlegende Einschränkungen, indem eine Validierungsausnahme aus den folgenden Gründen fehlschlägt:

  - Die Erweiterung ist nicht im CA-Zertifikat enthalten.
  - Die Erweiterung ist nicht markiert `critical`.

  Die Pfadlänge (pathLenConstraint) bestimmt, wie viele untergeordnete Objekte hinter dem importierten CA-Zertifikat existieren CAs können. AWS Private CA erzwingt die Pfadlänge, indem es aus den folgenden Gründen mit einer Validierungsausnahme fehlschlägt:

  - Das Importieren eines CA-Zertifikats würde die Pfadlängenbeschränkung im CA-Zertifikat oder in einem anderen CA-Zertifikat in der Kette verletzen.
  - Das Ausstellen eines Zertifikats würde eine Pfadlängenbeschränkung verletzen.
- Namenseinschränkungen geben einen Namensraum an, in dem sich alle Antragstellernamen in nachfolgenden Zertifikaten in einem Zertifizierungspfad befinden müssen. Einschränkungen gelten für den eindeutigen Namen des Antragstellers und für alternative Namen des Antragstellers.

Nicht erzwungen

- Zertifikatsrichtlinien. Zertifikatsrichtlinien regeln die Bedingungen, unter denen eine Zertifizierungsstelle Zertifikate ausstellt.
- Blockieren Sie AnyPolicy. Wird in Zertifikaten verwendet, die ausgestellt wurden für. CAs
- Alternativer Name des Ausstellers. Ermöglicht die Zuordnung zusätzlicher Identitäten zum Aussteller des CA-Zertifikats.
- Politische Einschränkungen. Diese Einschränkungen begrenzen die Kapazität einer Zertifizierungsstelle zum Ausstellen untergeordneter CA-Zertifikate.

- Zuordnungen von Richtlinien. Wird in CA-Zertifikaten verwendet. Listet ein oder mehrere Paare von auf OIDs; jedes Paar enthält ein issuerDomainPolicy und subjectDomainPolicy a.
- Attribute des Betreffverzeichnisses. Wird verwendet, um Identifikationsattribute des Betreffs zu vermitteln.
- Zugriff auf Informationen zum Fachgebiet. So greifen Sie auf Informationen und Dienste für den Betreff des Zertifikats zu, in dem die Erweiterung enthalten ist.
- Subject Key Identifier (SKI) und Authority Key Identifier (AKI). Das RFC benötigt ein CA-Zertifikat, um die SKI-Erweiterung zu enthalten. Von der Zertifizierungsstelle ausgestellte Zertifikate müssen eine AKI-Erweiterung enthalten, die der SKI des CA-Zertifikats entspricht. AWS erzwingt diese Anforderungen nicht. Wenn Ihr CA-Zertifikat keinen SKI enthält, ist das ausgestellte Endentitäts- oder das untergeordnete CA-Zertifikat stattdessen der SHA-1-Hash des öffentlichen Schlüssels des Ausstellers.
- SubjectPublicKeyInfound Alternativer Betreffname (SAN). Beim Ausstellen eines Zertifikats werden die Erweiterungen SubjectPublicKeyInfo und die SAN-Erweiterungen aus der bereitgestellten CSR AWS Private CA kopiert, ohne eine Überprüfung durchzuführen.

Preisgestaltung für AWS Private Certificate Authority

Ihrem Konto wird ab dem Zeitpunkt, zu dem Sie sie erstellen, ein monatlicher Preis für jede private CA in Rechnung gestellt. Zudem wird Ihnen für jedes ausgestellte Zertifikat eine Gebühr berechnet. Diese Gebühr beinhaltet Zertifikate, die Sie aus ACM exportieren, und Zertifikate, die Sie über die AWS Private CA API oder AWS Private CA CLI erstellen. Für eine private CA wird nichts mehr berechnet, nachdem sie gelöscht wurde. Wenn Sie aber eine gelöschte CA wiederherstellen, bezahlen Sie für die Zeit zwischen Löschung und Wiederherstellung. Private Zertifikate, auf deren privaten Schlüssel Sie nicht zugreifen können, sind kostenlos. Dazu gehören Zertifikate, die mit integrierten Diensten wie Elastic Load Balancing und API Gateway verwendet werden. CloudFront

Die neuesten AWS Private CA Preisinformationen finden Sie unter AWS Private Certificate Authority Preise. Sie können auch den AWS Preisrechner verwenden, um die Kosten zu schätzen.

Begriffe und Konzepte für AWS Private CA

Die folgenden Begriffe und Konzepte können Ihnen bei der Arbeit mit helfen AWS Private Certificate Authority.

Themen

- Vertrauensstellung
- TLS-Serverzertifikate
- Signatur des Zertifikats
- Zertifizierungsstelle
- Stammzertifizierungsstelle
- CA-Zertifikat
- CA-Stammzertifikat
- Zertifikat der endgültigen Entität
- Selbstsignierte Zertifikate
- Privates Zertifikat
- Pfad des Zertifikats
- Beschränkung der Pfadlänge

Vertrauensstellung

Damit ein Webbrowser der Identität einer Website vertraut, muss der Browser das Zertifikat der Website überprüfen können. Browser vertrauen jedoch nur einer kleinen Anzahl von Zertifikaten, die als CA-Stammzertifikate bekannt sind. Ein vertrauenswürdiger Dritter, als Zertifikatsstelle (Certificate Authority, CA) bezeichnet, validiert die Identität der Website und stellt ein signiertes digitales Zertifikat für den Betreiber der Website aus. Der Browser kann dann die digitale Signatur überprüfen, um die Identität der Website zu validieren. Wenn die Validierung erfolgreich ist, zeigt der Browser ein Schlosssymbol in der Adressleiste an.

TLS-Serverzertifikate

HTTPS-Transaktionen erfordern Serverzertifikate zur Authentifizierung eines Servers. Ein Serverzertifikat ist eine X.509-v3-Datenstruktur, mit der der öffentliche Schlüssel im Zertifikat an das Subject des Zertifikats gebunden wird. Ein TLS-Zertifikat wird von einer Zertifizierungsstelle (CA) signiert. Es enthält den Namen des Servers, den Gültigkeitszeitraum, den öffentlichen Schlüssel, den Signaturalgorithmus und vieles mehr.

Signatur des Zertifikats

Eine digitale Signatur ist ein verschlüsselter Hash über ein Zertifikat. Eine Signatur wird verwendet, um die Integrität der Zertifikatdaten zu bestätigen. Ihre private Zertifizierungsstelle erstellt eine

Signatur mithilfe einer kryptografischen Hash-Funktion, z. B. SHA256 über dem Inhalt des Zertifikats mit variabler Größe. Diese Hash-Funktion erzeugt eine effektiv fälschungssichere Datenzeichenfolge mit fester Größe. Diese Zeichenfolge wird als Hash bezeichnet. Die Zertifizierungsstelle verschlüsselt dann den Hash-Wert mit dem privaten Schlüssel und verkettet den verschlüsselten Hash mit dem Zertifikat.

Zertifizierungsstelle

Eine Zertifizierungsstelle (Certificate Authority, CA) stellt digitale Zertifikate aus und widerruft sie bei Bedarf. Der gängigste Zertifikatstyp basiert auf dem ISO X.509-Standard. Ein X.509-Zertifikat bestätigen die Identität des Zertifikatantragstellers und bindet die Identität an einen öffentlichen Schlüssel. Dabei kann es sich um einen Benutzer, eine Anwendung, einen Computer oder ein anderes Gerät handeln. Die Zertifizierungsstelle signiert ein Zertifikat, indem der Inhalt gehasht wird und der Hash dann mit dem privaten Schlüssel verschlüsselt wird, der mit dem öffentlichen Schlüssel im Zertifikat verbunden ist. Eine Client-Anwendung wie z. B. ein Webbrowser, der die Identität eines Antragstellers bestätigen muss, verwendet den öffentlichen Schlüssel zum Entschlüsseln der Zertifikatsignatur. Anschließend wird der Inhalt des Zertifikats gehasht und der gehashte Wert mit der entschlüsselten Signatur verglichen, um festzustellen, ob sie übereinstimmen. Weitere Informationen zur Zertifikatsignatur finden Sie unter Signatur des Zertifikats.

Sie können AWS Private CA damit eine private Zertifizierungsstelle erstellen und die private Zertifizierungsstelle verwenden, um Zertifikate auszustellen. Ihre private Zertifizierungsstelle stellt nur private SSL-/TLS-Zertifikate für die Verwendung innerhalb Ihrer Organisation aus. Weitere Informationen finden Sie unter Privates Zertifikat. Ihre private Zertifizierungsstelle erfordert auch ein Zertifikat, bevor Sie sie verwenden können. Weitere Informationen finden Sie unter CA-Zertifikat.

Stammzertifizierungsstelle

Eine Stammzertifizierungsstelle ist ein kryptografischer Baustein und der Ausgangspunkt für das Ausstellen vertrauenswürdiger Zertifikate. Sie besteht aus einem privaten Schlüssel zur Unterzeichnung (Ausstellung) von Zertifikaten und einem Stammzertifikat, das die Stammzertifizierungsstelle identifiziert und den privaten Schlüssel mit ihrem Namen verknüpft. Das Stammzertifikat wird an die Vertrauensspeicher jeder Entität in einer Umgebung verteilt. Administratoren erstellen Vertrauensspeicher, die nur Vertrauensspeicher enthalten CAs , denen sie vertrauen. Administratoren aktualisieren oder integrieren die Trust Stores in den Betriebssystemen, Instanzen und Hostcomputer-Images von Entitäten in ihrer Umgebung. Versuchen Ressourcen, gegenseitig eine Verbindung herzustellen, werden die Zertifikate der jeweiligen Entitäten überprüft. Ein Client prüft die Zertifikate auf ihre Gültigkeit und darauf, ob eine Kette vom Zertifikat zu

einem Stammzertifikat im Trust Store vorhanden ist. Wenn diese Bedingungen erfüllt sind, erfolgt ein „Handshake“ zwischen den Ressourcen. Dieser „Handshake“ weist die Identität der einzelnen Entitäten gegenüber den anderen kryptografisch nach und richtet einen verschlüsselten Kommunikationskanal (TLS/SSL) zwischen diesen ein.

CA-Zertifikat

Ein Zertifizierungsstellenzertifikat bestätigt die Identität einer Zertifizierungsstelle und bindet sie an den öffentlichen Schlüssel, der im Zertifikat enthalten ist.

Sie können AWS Private CA es verwenden, um eine private Stammzertifizierungsstelle oder eine private untergeordnete Zertifizierungsstelle zu erstellen, die jeweils durch ein CA-Zertifikat abgesichert sind. Untergeordnete CA-Zertifikate werden von einem anderen CA-Zertifikat in einer Vertrauenskette signiert. Bei einer Stammzertifizierungsstelle ist das Zertifikat jedoch selbstsigniert. Sie können auch eine externe Stammzertifizierungsstelle einrichten (z. B. lokal gehostet). Anschließend können Sie mit Ihrer Stammzertifizierungsstelle ein untergeordnetes Stamm-CA-Zertifikat signieren, das von AWS Private CA gehostet wird.

Das folgende Beispiel zeigt die typischen Felder, die in einem AWS Private CA X.509-CA-Zertifikat enthalten sind. Beachten Sie, dass für ein CA-Zertifikat der `CA:`-Wert im Feld `Basic Constraints` auf `TRUE` festgelegt ist.

```
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
```

CA-Stammzertifikat

Eine Zertifizierungsstelle (CA) befindet sich in der Regel innerhalb einer hierarchischen Struktur, die mehrere andere CAs mit klar definierten Eltern-Kind-Beziehungen zwischen ihnen enthält. Untergeordnete oder untergeordnete Personen CAs werden von ihren Eltern zertifiziert CAs, wodurch eine Zertifikatskette entsteht. Die CA oben in der Hierarchie wird als die Stamm-CA bezeichnet und ihr Zertifikat wird Stammzertifikat genannt. Dieses Zertifikat ist in der Regel selbstsigniert.

Zertifikat der endgültigen Entität

Ein Endzertifizierungszertifikat identifiziert eine Ressource, z. B. einen Server, eine Instanz, einen Container oder ein Gerät. Im Gegensatz zu CA-Zertifikaten können Entity-Zertifikate nicht zur Ausstellung von Zertifikaten verwendet werden. Andere gebräuchliche Begriffe für Endzertifikate sind „Client“ oder „Leaf“ -Zertifikat.

Selbstsignierte Zertifikate

Ein Zertifikat, das anstatt von einer höheren Zertifizierungsstelle vom Aussteller signiert ist. Im Gegensatz zu Zertifikaten, die von einem sicheren Stamm ausgestellt wurden, der von einer Zertifizierungsstelle verwaltet wird, fungieren selbstsignierte Zertifikate als eigene Stammzertifikate und weisen daher erhebliche Einschränkungen auf: Sie können zur Verschlüsselung auf der Leitung verwendet werden, aber nicht zur Überprüfung der Identität, und sie können nicht gesperrt werden. Sie sind aus sicherheitstechnischer Sicht inakzeptabel. Organisationen nutzen sie dennoch, weil sie einfach zu generieren sind, kein Fachwissen und keine Infrastruktur benötigen und von vielen Anwendungen akzeptiert werden. Es gibt keine Kontrollen für die Ausstellung von selbstsignierten Zertifikaten. Organisationen, die selbstsignierte Zertifikate verwenden, gehen ein höheres Risiko für Ausfälle ein, welche durch das Ablaufen von Zertifikaten verursacht werden, denn sie verfügen über keine Möglichkeit, die Ablaufdaten nachzuverfolgen.

Privates Zertifikat

AWS Private CA Zertifikate sind private SSL/TLS-Zertifikate, die Sie innerhalb Ihrer Organisation verwenden können, die jedoch im öffentlichen Internet nicht vertrauenswürdig sind. Verwenden Sie sie zum Identifizieren von Ressourcen wie z. B. Clients, Servern, Anwendungen, Services, Geräten und Benutzern. Wenn ein sicherer verschlüsselter Kommunikationskanal hergestellt wird, verwendet jede Ressource ein Zertifikat wie das folgende sowie Verschlüsselungstechniken zum Nachweis ihrer Identität an eine andere Ressource. Interne API-Endpunkte, Webserver, VPN-Benutzer, IoT-Geräte und zahlreiche weitere Anwendungen verwenden private Zertifikate zur Herstellung von verschlüsselten Kommunikationskanälen, die für ihre sichere Operation notwendig sind. Private Zertifikate sind standardmäßig nicht öffentlich vertrauenswürdig. Ein interner Administrator muss Anwendungen explizit konfigurieren, damit sie privaten Zertifikaten vertrauen und diese verteilen.

```
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
```

Pfad des Zertifikats

Ein Client, der auf ein Zertifikat angewiesen ist, überprüft, ob ein Pfad vom Endentitätszertifikat, möglicherweise über eine Kette von Zwischenzertifikaten, zu einem vertrauenswürdigen Stamm existiert. Der Client überprüft, ob alle Zertifikate des Pfads gültig sind (nicht widerrufen). Außerdem wird überprüft, ob das Endzertifikat nicht abgelaufen ist, seine Integrität besitzt (nicht manipuliert oder verändert wurde) und ob die Beschränkungen im Zertifikat durchgesetzt werden.

Beschränkung der Pfadlänge

Die grundlegenden Einschränkungen pathLenConstraintfür ein CA-Zertifikat legen die Anzahl der untergeordneten CA-Zertifikate fest, die in der darunter liegenden Kette vorhanden sein können. Beispielsweise kann ein CA-Zertifikat mit einer Pfadlängenbeschränkung von Null kein untergeordnetes CAs Zertifikat haben. Einer Zertifizierungsstelle mit einer Pfadlängenbeschränkung von eins kann bis zu einer untergeordneten Ebene untergeordnet sein CAs . RFC 5280 definiert dies als „die maximale Anzahl von non-self-issued Zwischenzertifikaten, die diesem Zertifikat in einem gültigen Zertifizierungspfad folgen können“. Der Wert für die Pfadlänge schließt das Endzertifikat aus, obwohl es in informellen Formulierungen über die „Länge“ oder „Tiefe“ einer Validierungskette enthalten sein kann... was zu Verwirrung führen kann.

Was ist der beste Zertifikatsservice für meine Bedürfnisse?

Es gibt zwei AWS Dienste für die Ausstellung und Bereitstellung von X.509-Zertifikaten. Wählen Sie den Service, der Ihren Bedürfnissen am besten entspricht. Zu den Überlegungen gehört, ob Sie öffentlich oder privat zugängliche Zertifikate, benutzerdefinierte Zertifikate, Zertifikate, die Sie in anderen AWS Diensten einsetzen möchten, oder ob Sie automatisierte Zertifikatsverwaltung und -erneuerung benötigen.

1. AWS Private CA – Dieser Service richtet sich an Unternehmenskunden, die eine Public-Key-Infrastruktur (PKI) innerhalb der AWS -Cloud und ist für den privaten Gebrauch innerhalb einer Organisation bestimmt. Mit AWS Private CA können Sie Ihre eigene Zertifizierungsstellenhierarchie erstellen und damit Zertifikate ausstellen, um interne Benutzer, Computer, Anwendungen, Dienste, Server und andere Geräte zu authentifizieren und Computercode zu signieren. Zertifikate, die von einer privaten Zertifizierungsstelle ausgestellt werden, sind nur innerhalb Ihrer Organisation vertrauenswürdig, nicht im Internet.

   Nachdem Sie eine private Zertifizierungsstelle erstellt haben, können Sie Zertifikate direkt ausstellen (d. h. ohne Validierung durch eine Drittanbieter-CA) und sie an die internen Anforderungen Ihrer Organisation anpassen. Beispielsweise können Sie:

   - Zertifikate mit einem beliebigen Zertifikatthemennamen erstellen.
   - Zertifikate mit einem beliebigen Ablaufdatum erstellen.
   - Jeden unterstützten privaten Schlüsselalgorithmus und Schlüssellänge verwenden.
   - Jeden unterstützten Signaturalgorithmus verwenden.
   - Das Ausstellen von Zertifikaten mithilfe von Vorlagen steuern.

   Sie sind an der richtigen Stelle für diesen Service. Melden Sie sich zunächst bei der https://console.aws.amazon.com/acm-pca/-Konsole an.

2. AWS Certificate Manager (ACM) — Dieser Dienst verwaltet Zertifikate für Unternehmenskunden, die eine öffentlich vertrauenswürdige, sichere Webpräsenz mithilfe von TLS benötigen. Sie können ACM-Zertifikate in AWS Elastic Load Balancing, Amazon CloudFront, Amazon API Gateway und anderen integrierten Services bereitstellen. Die gebräuchlichste Anwendung dieser Art ist eine sichere öffentliche Website mit erheblichen Anforderungen an den Datenverkehr.

   Mit diesem Service können Sie öffentliche, von ACM bereitgestellte Zertifikate (ACM-Zertifikate) oder Zertifikate, die Sie in ACM importieren, verwenden.. Wenn Sie AWS Private CA eine

Zertifizierungsstelle erstellen, kann ACM die Zertifikatsausstellung von dieser privaten Zertifizierungsstelle aus verwalten und die Verlängerung von Zertifikaten automatisieren.

Weitere Informationen finden Sie im AWS Certificate Manager -Benutzerhandbuch.

AWS Private CA bewährte Verfahren

Bewährte Verfahren sind Empfehlungen, die Ihnen helfen können, sie AWS Private CA effektiv zu nutzen. Die folgenden bewährten Methoden basieren auf realen Erfahrungen von AWS Private CA Kunden AWS Certificate Manager und Kunden.

Dokumentation der Struktur und der Richtlinien von CA

AWS empfiehlt, alle Ihre Richtlinien und Praktiken für den Betrieb Ihrer CA zu dokumentieren. Dazu kann Folgendes gehören:

- Schlussfolgerung für Ihre Entscheidungen bezüglich der CA-Struktur
- Ein Diagramm, das Ihre CAs und ihre Beziehungen zeigt
- Richtlinien zu CA-Gültigkeitszeiträumen
- Planung der CA-Abfolge
- Richtlinien zur Pfadlänge
- Berechtigungenkatalog
- Beschreibung der administrativen Kontrollstrukturen
- Sicherheit

Sie können diese Informationen in zwei Dokumenten erfassen, die als Certification Policy (CP) und Certification Practices Statement (CPS) bezeichnet werden. Die Rahmenbedingungen für die Erfassung wichtiger Informationen zu Ihren Zertifizierungsstellen-Operationen finden Sie in RFC 3647.

Minimiere die Nutzung der Stammzertifizierungsstelle, wenn möglich

Eine Stammzertifizierungsstelle sollte im Allgemeinen nur zur Ausstellung von Zertifikaten für Zwischenzertifikate verwendet werden CAs. Auf diese Weise kann die Stammzertifizierungsstelle vor Gefahren geschützt aufbewahrt werden, während die CAs Zwischenzertifizierungsstelle die tägliche Aufgabe der Ausstellung von Endzertifikaten übernimmt.

Wenn es in Ihrer Organisation derzeit jedoch üblich ist, Endentitätszertifikate direkt von einer Stammzertifizierungsstelle auszustellen, AWS Private CA kann dies diesen Arbeitsablauf unterstützen und gleichzeitig die Sicherheit und die Betriebskontrollen verbessern. Für die Ausstellung von Endentitätszertifikaten in diesem Szenario ist eine IAM-Berechtigungsrichtlinie erforderlich, laut der Ihre Stammzertifizierungsstelle eine Endentitätszertifikatvorlage verwenden darf. Weitere Informationen zu IAM-Richtlinien finden Sie unter Identity and Access Management (IAM) für AWS Private Certificate Authority.

> **Note**
>
> Diese Konfiguration bringt Einschränkungen mit sich, die zu betrieblichen Herausforderungen führen können. Wenn Ihre Stammzertifizierungsstelle beispielsweise kompromittiert ist oder verloren geht, müssen Sie eine neue Stammzertifizierungsstelle erstellen und diese an alle Clients in Ihrer Umgebung verteilen. Bis dieser Wiederherstellungsvorgang abgeschlossen ist, können Sie keine neuen Zertifikate ausstellen. Das Ausstellen von Zertifikaten direkt von einer Stammzertifizierungsstelle verhindert auch das Einschränken des Zugriff und das Begrenzen der Anzahl der vom Stammverzeichnis ausgestellten Zertifikate. Diese beiden Verfahren gelten als bewährte Methoden für die Verwaltung einer Stammzertifizierungsstelle.

Geben Sie der Root-CA ihre eigene AWS-Konto

Es wird empfohlen, eine Stammzertifizierungsstelle und eine untergeordnete Zertifizierungsstelle in zwei verschiedenen AWS Konten zu erstellen. Dadurch verfügen Sie über zusätzlichen Schutz und eine bessere Zugriffskontrolle für Ihre Stammzertifizierungsstelle. Hierfür exportieren Sie die CSR aus der untergeordneten Zertifizierungsstelle in ein Konto und signieren sie mit einer Stammzertifizierungsstelle in einem anderen Konto. Der Vorteil dieses Ansatzes besteht darin, dass Sie die Kontrolle über Ihr CAs Konto trennen können. Der Nachteil besteht darin, dass Sie den AWS-Managementkonsole Assistenten nicht verwenden können, um das Signieren des CA-Zertifikats einer untergeordneten Zertifizierungsstelle von Ihrer Stammzertifizierungsstelle aus zu vereinfachen.

> **Important**
>
> Wir empfehlen dringend, bei jedem Zugriff die Multi-Faktor-Authentifizierung (MFA) zu verwenden. AWS Private CA

Separate Administrator- und Ausstellerrollen

Die Rolle des CA-Administrators sollte von der Rolle der Benutzer getrennt sein, die nur Endentitätszertifikate ausstellen müssen. Wenn sich Ihr CA-Administrator und Ihr Zertifikatsaussteller in derselben Organisation befinden AWS-Konto, können Sie die Ausstellerberechtigungen einschränken, indem Sie einen IAM-Benutzer speziell für diesen Zweck erstellen.

Implementieren Sie den verwalteten Widerruf von Zertifikaten

Der verwaltete Widerruf benachrichtigt die Zertifikatsclients automatisch, wenn ein Zertifikat gesperrt wurde. Möglicherweise müssen Sie ein Zertifikat sperren, wenn dessen kryptografische Informationen kompromittiert wurden oder wenn es irrtümlich ausgestellt wurde. Clients lehnen es in der Regel ab, widerrufene Zertifikate zu akzeptieren. AWS Private CA bietet zwei Standardoptionen für den verwalteten Widerruf: das Online Certificate Status Protocol (OCSP) und Zertifikatssperrlisten (CRLs). Weitere Informationen finden Sie unter Planen Sie Ihre Methode zum Widerruf von AWS Private CA Zertifikaten.

Einschalten AWS CloudTrail

Aktivieren Sie die CloudTrail Protokollierung, bevor Sie eine private Zertifizierungsstelle erstellen und mit dem Betrieb beginnen. Mit CloudTrail können Sie einen Verlauf der AWS API-Aufrufe für Ihr Konto abrufen, um Ihre AWS Bereitstellungen zu überwachen. Dieser Verlauf umfasst API-Aufrufe AWS-Managementkonsole, die von den AWS SDKs AWS Command Line Interface, den und übergeordneten AWS Diensten getätigt wurden. Sie können auch feststellen, welche Benutzer und Konten die PCA-API-Operationen aufgerufen haben, von welcher Quell-IP-Adresse diese Aufrufe stammen und zu welchem Zeitpunkt die Aufrufe erfolgten. Sie können sie mithilfe der API CloudTrail in Anwendungen integrieren, die Erstellung von Trails für Ihr Unternehmen automatisieren, den Status Ihrer Trails überprüfen und kontrollieren, wie Administratoren die CloudTrail Anmeldung ein- und ausschalten. Weitere Informationen finden Sie unter Erstellen eines Trails. Gehen Sie zuProtokollieren von AWS Private Certificate Authority API-Aufrufen mit AWS CloudTrail, um sich Beispielpfade für AWS Private CA Operationen anzusehen.

Rotieren Sie den privaten CA-Schlüssel

Es hat sich bewährt, in regelmäßigen Abständen den privaten Schlüssel für Ihre private Zertifizierungsstelle zu aktualisieren. Sie können einen Schlüssel aktualisieren, indem Sie ein neues CA-Zertifikat importieren oder Sie können die private Zertifizierungsstelle durch eine neue ersetzen.

> **Note**
>
> Wenn Sie die CA selbst austauschen, beachten Sie, dass sich der ARN der CA ändert. Dies würde dazu führen, dass die Automatisierung, die sich auf einen fest codierten ARN stützt, fehlschlägt.

Unbenutztes löschen CAs

Sie können eine private Zertifizierungsstelle dauerhaft löschen. Sie können dies tun, wenn Sie die Zertifizierungsstelle nicht mehr benötigen oder wenn Sie sie durch eine Zertifizierungsstelle mit einem neueren privaten Schlüssel ersetzen möchten. Zum sicheren Löschen einer Zertifizierungsstelle empfehlen wir die in Löschen Ihrer privaten CA beschriebene Vorgehensweise.

> **Note**
>
> AWS stellt Ihnen eine CA in Rechnung, bis sie gelöscht wurde.

Blockieren Sie den öffentlichen Zugriff auf Ihre CRLs

AWS Private CA empfiehlt die Verwendung der Amazon S3 S3-Funktion Block Public Access (BPA) für Buckets, die Folgendes enthalten: CRLs Dadurch wird vermieden, dass Details Ihrer privaten PKI unnötigerweise potenziellen Gegnern zugänglich gemacht werden. BPA ist eine bewährte Methode für S3 und ist bei neuen Buckets standardmäßig aktiviert. In einigen Fällen ist eine zusätzliche Einrichtung erforderlich. Weitere Informationen finden Sie unter Aktivieren Sie S3 Block Public Access (BPA) mit CloudFront.

Bewährte Methoden für Amazon EKS-Anwendungen

Beachten Sie AWS Private CA bei der Bereitstellung von X.509-Zertifikaten für Amazon EKS die Empfehlungen zur Sicherung von Umgebungen mit mehreren Mandanten in den Amazon EKS Best Practices Guides. Allgemeine Informationen zur Integration AWS Private CA mit Kubernetes finden Sie unter. Kubernetes sichern mit AWS Private Certificate Authority

Verwenden Sie AWS Private CA mit dem AWS SDK for Java

Sie können die AWS Private Certificate Authority API verwenden, um programmgesteuert mit dem Dienst zu interagieren, indem Sie HTTP-Anfragen senden. Der Service gibt HTTP-Antworten zurück. Weitere Informationen finden Sie unter AWS Private Certificate Authority API-Referenz.

Zusätzlich zur HTTP-API können Sie die AWS SDKs Befehlszeilentools und zur Interaktion verwenden AWS Private CA. Dies wird anstelle der HTTP-API empfohlen. Weitere Informationen finden Sie unter Tools für Amazon Web Services. In den folgenden Themen sehen Sie, wie Sie AWS SDK for Java zum Programmieren der AWS Private CA -API verwenden.

Die DescribeCertificateAuthorityAuditReportOperationen GetCertificateAuthorityCsrGetCertificate, und unterstützen Kellner. Sie können Waiter verwenden, um den Fortschritt Ihres Codes basierend auf der Anwesenheit oder dem Zustand bestimmter Ressourcen zu steuern. Weitere Informationen finden Sie in den folgenden Themen sowie unter Kellner AWS SDK for Java im AWS Entwickler-Blog.

AWS Private CA API-Beispiele

Die folgenden Codebeispiele zeigen, wie Sie ausgewählte AWS Private CA API-Aktionen und Datentypen mit dem verwenden AWS SDK for Java.

Themen

- Programmgesteuert eine Root-CA erstellen und aktivieren
- Programmgesteuert eine untergeordnete Zertifizierungsstelle erstellen und aktivieren
- CreateCertificateAuthority
- CreateCertificateAuthorityZur Unterstützung von Active Directory verwenden
- CreateCertificateAuthorityAuditReport
- CreatePermission
- DeleteCertificateAuthority
- DeletePermission
- DeletePolicy
- DescribeCertificateAuthority
- DescribeCertificateAuthorityAuditReport
- GetCertificate
- GetCertificateAuthorityCertificate

- GetCertificateAuthorityCsr
- GetPolicy
- ImportCertificateAuthorityCertificate
- IssueCertificate
- ListCertificateAuthorities
- ListPermissions
- ListTags
- PutPolicy
- RestoreCertificateAuthority
- RevokeCertificate
- TagCertificateAuthorities
- UntagCertificateAuthority
- UpdateCertificateAuthority
- Erstellen Sie CAs Zertifikate mit benutzerdefinierten Betreffnamen
- Zertifikate mit benutzerdefinierten Erweiterungen erstellen

Programmgesteuert eine Root-CA erstellen und aktivieren

Dieses Java-Beispiel zeigt, wie Sie eine Root-CA mithilfe der folgenden AWS Private CA API-Aktionen aktivieren:

- 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);
    }
}
```

Programmgesteuert eine untergeordnete Zertifizierungsstelle erstellen und aktivieren

Dieses Java-Beispiel zeigt, wie Sie eine untergeordnete Zertifizierungsstelle mithilfe der folgenden AWS Private CA API-Aktionen aktivieren:

- 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

Das folgende Java-Beispiel zeigt, wie die CreateCerticateAuthorityOperation verwendet wird.

Mit dieser Operation wird eine private untergeordnete Zertifizierungsstelle (Certificate Authority, CA) erstellt. Sie müssen die CA-Konfiguration, die Sperrkonfiguration, den CA-Typ und ein optionales Idempotenz-Token festlegen.

Die CA-Konfiguration legt Folgendes fest:

- Den Namen des Algorithmus und die Schlüsselgröße, die zum Erstellen des privaten CA-Schlüssels verwendet werden soll.
- Die Art des Signaturalgorithmus, den die CA verwendet, um ihre eigenen Certificate Signing Request und OCSP-Antworten zu signieren CRLs
- X.500-Themeninformationen

Die CRL-Konfiguration legt Folgendes fest:

- Die Ablaufzeit der Zertifikatsperrliste in Tagen (die Gültigkeitsdauer der CRL)
- Der Amazon S3 S3-Bucket, der die CRL enthalten wird
- Einen CNAME-Alias für den S3-Bucket, der in den von der Zertifizierungsstelle ausgestellten Zertifikaten enthalten ist

Wenn diese Aktion erfolgreich ist, gibt diese Funktion den Amazon-Ressourcennamen (ARN) der Zertifizierungsstelle zurück.

```
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);
    }
}
```

Die Ausgabe sollte in etwa wie folgt aussehen:

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

CreateCertificateAuthorityZur Unterstützung von Active Directory verwenden

Das folgende Java-Beispiel zeigt, wie der CreateCerticateAuthorityVorgang verwendet wird, um eine CA zu erstellen, die im Enterprise NTAuth Store von Microsoft Active Directory (AD) installiert werden kann.

Der Vorgang erstellt eine private Stammzertifizierungsstelle (CA) mithilfe benutzerdefinierter Objektbezeichner (OIDs). Weitere Informationen und ein AWS CLI Beispiel für einen gleichwertigen Vorgang finden Sie unter Erstellen einer Zertifizierungsstelle für die Active Directory-Anmeldung.

Wenn diese Aktion erfolgreich ist, gibt diese Funktion den Amazon-Ressourcennamen (ARN) der Zertifizierungsstelle zurück.

```
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);
    }
}
```

Die Ausgabe sollte in etwa wie folgt aussehen:

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

CreateCertificateAuthorityAuditReport

Das folgende Java-Beispiel zeigt, wie die CreateCertificateAuthorityAuditReportOperation verwendet wird.

Mit dieser Operation wird ein Audit-Bericht erstellt, in dem jedes Ausstellen und jeder Widerruf eines Zertifikats aufgeführt sind. Der Bericht wird in dem Amazon S3 S3-Bucket gespeichert, den Sie bei der Eingabe angeben. Sie können einmal innerhalb von 30 Minuten einen neuen Bericht generieren.

```
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);

    }
}
```

Die Ausgabe sollte in etwa wie folgt aussehen:

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

CreatePermission

Das folgende Java-Beispiel zeigt, wie die CreatePermissionOperation verwendet wird.

Der Vorgang weist einem bestimmten AWS Dienstprinzipal Zugriffsberechtigungen von einer privaten Zertifizierungsstelle zu. Services können die Berechtigung erhalten, Zertifikate von einer privaten Zertifizierungsstelle zu erstellen und abzurufen sowie die aktiven Berechtigungen aufzulisten, die die private Zertifizierungsstelle erteilt hat. Um Zertifikate automatisch über ACM zu erneuern, müssen Sie dem ACM-Dienstprinzipal (`ListPermissions`) alle möglichen Berechtigungen (`IssueCertificate` `GetCertificate`, und) von der CA zuweisen. `acm.amazonaws.com` Sie können den ARN einer CA finden, indem Sie die ListCertificateAuthoritiesFunktion aufrufen.

Sobald eine Berechtigung erstellt wurde, können Sie sie mit der ListPermissionsFunktion überprüfen oder mit der DeletePermissionFunktion löschen.

```
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

Das folgende Java-Beispiel zeigt, wie die DeleteCertificateAuthorityOperation verwendet wird.

Dieser Vorgang löscht die private Zertifizierungsstelle (CA), die Sie mit diesem CreateCertificateAuthorityVorgang erstellt haben. Für die Operation `DeleteCertificateAuthority` müssen Sie einen ARN angeben, damit die Zertifizierungsstelle gelöscht werden kann. Sie können den ARN finden, indem Sie die ListCertificateAuthoritiesOperation aufrufen. Sie können die private Zertifizierungsstelle sofort löschen, wenn ihr Status `CREATING` oder `PENDING_CERTIFICATE` lautet. Wenn Sie das Zertifikat bereits importiert haben, können Sie sie jedoch nicht sofort löschen. Sie müssen zuerst die CA deaktivieren, indem Sie die UpdateCertificateAuthorityOperation aufrufen und den `Status` Parameter auf setzen`DISABLED`. Dann können Sie den Parameter `PermanentDeletionTimeInDays`in der Operation `DeleteCertificateAuthority`verwenden, um die Anzahl der Tage anzugeben (von 7 bis 30). Während dieses Zeitraums kann die private Zertifizierungsstelle wieder auf den Status `disabled` zurückgesetzt werden. Wenn Sie den `PermanentDeletionTimeInDays`-Parameter nicht festlegen, ist der Wiederherstellungszeitraum standardmäßig 30 Tage. Nach Ablauf dieses Zeitraums wird die private Zertifizierungsstelle dauerhaft gelöscht und kann nicht wiederhergestellt werden. Weitere Informationen finden Sie unter Wiedererstellen einer Zertifizierungsstelle.

Ein Java-Beispiel, das Ihnen zeigt, wie Sie den RestoreCertificateAuthorityVorgang verwenden, finden Sie unterRestoreCertificateAuthority.

```
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

Das folgende Java-Beispiel zeigt, wie die DeletePermissionOperation verwendet wird.

Der Vorgang löscht Berechtigungen, die eine private Zertifizierungsstelle mithilfe des Vorgangs an einen AWS Dienstprinzipal delegiert hat. CreatePermissions Sie können den ARN einer CA finden, indem Sie die ListCertificateAuthoritiesFunktion aufrufen. Sie können die Berechtigungen überprüfen, die eine CA gewährt hat, indem Sie die ListPermissionsFunktion aufrufen.

```
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

Das folgende Java-Beispiel zeigt, wie die DeletePolicyOperation verwendet wird.

Der Vorgang löscht die ressourcenbasierte Richtlinie, die an eine private Zertifizierungsstelle angehängt ist. Eine ressourcenbasierte Richtlinie wird verwendet, um die kontenübergreifende gemeinsame Nutzung der CA zu ermöglichen. Sie können den ARN einer privaten CA ermitteln, indem Sie die ListCertificateAuthoritiesAktion aufrufen.

Zu den verwandten API-Aktionen gehören PutPolicyund GetPolicy.

```
package com.amazonaws.samples;

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

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

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

public class DeletePolicy {

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

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

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

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

        // Create the request object.
```

```
        DeletePolicyRequest req = new DeletePolicyRequest();

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

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

DescribeCertificateAuthority

Das folgende Java-Beispiel zeigt, wie die DescribeCertificateAuthorityOperation verwendet wird.

Die Operation führt Informationen zu Ihrer privaten Zertifizierungsstelle (CA) auf. Sie müssen den Amazon-Ressourcennamen (ARN) der privaten Zertifizierungsstelle angeben. Die Ausgabe enthält den Status Ihrer Zertifizierungsstelle. Dies kann einer der folgenden sein:

- `CREATING`— AWS Private CA erstellt Ihre private Zertifizierungsstelle.
- `PENDING_CERTIFICATE`— Das Zertifikat steht noch aus. Sie müssen Ihre lokalen Stamm-CA oder Ihre untergeordnete CA zum Signieren Ihrer privaten CA CSR verwenden und dann in PCA importieren.
- `ACTIVE`— Ihre private CA ist aktiv.

- `DISABLED`— Ihre private CA wurde deaktiviert.
- `EXPIRED`— Ihr privates CA-Zertifikat ist abgelaufen.
- `FAILED`— Ihre private CA kann nicht erstellt werden.
- `DELETED`— Ihre private CA befindet sich noch in der Wiederherstellungszeit. Danach wird sie dauerhaft gelöscht.

```
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

Das folgende Java-Beispiel zeigt, wie die DescribeCertificateAuthorityAuditReportOperation verwendet wird.

Der Vorgang listet Informationen zu einem bestimmten Prüfbericht auf, den Sie durch den Aufruf des CreateCertificateAuthorityAuditReportVorgangs erstellt haben. Audit-Informationen werden jedes Mal erstellt, wenn der private Schlüssel der privaten CA verwendet wird. Der private Schlüssel wird verwendet, wenn Sie ein Zertifikat ausstellen, eine CRL signieren oder ein Zertifikat widerrufen.

```
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);
   }
}
```

Die Ausgabe sollte in etwa wie folgt aussehen:

```
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

Das folgende Java-Beispiel zeigt, wie die GetCertificateOperation verwendet wird.

Mit dieser Funktion wird ein Zertifikat von Ihrer privaten Zertifizierungsstelle abgerufen. Der ARN des Zertifikats wird zurückgegeben, wenn Sie den IssueCertificateVorgang aufrufen. Sie müssen sowohl den ARN Ihrer privaten Zertifizierungsstelle als auch den ARN des ausgestellten Zertifikats beim Aufrufen der Operation `GetCertificate` angeben. Sie können das Zertifikat abrufen, wenn es sich im Status `ISSUED` befindet. Sie können den CreateCertificateAuthorityAuditReportVorgang aufrufen, um einen Bericht zu erstellen, der Informationen über alle Zertifikate enthält, die von Ihrer privaten Zertifizierungsstelle ausgestellt und gesperrt wurden.

```
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);
   }
}
```

Die Ausgabe sollte eine Zertifikatskette sein, die der ähnelt, die Sie für die Zertifizierungsstelle (CA) und das Zertifikat angegeben haben.

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

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

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

GetCertificateAuthorityCertificate

Das folgende Java-Beispiel zeigt, wie die GetCertificateAuthorityCertificateOperation verwendet wird.

Mit dieser Operation werden das Zertifikat und die Zertifikatskette für Ihre private CA abgerufen. Sowohl das Zertifikat als auch die Kette sind Base64-kodierte Zeichenketten im PEM-Format. Die Zertifikatskette enthält kein CA-Zertifikat. Jedes Zertifikat in der Kette signiert das vorhergehende.

```
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);
   }
}
```

Die Ausgabe sollte ein Zertifikat und eine Zertifikatskette sein, ähnlich den Folgenden, die Sie für die Zertifizierungsstelle (CA) angegeben haben.

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

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

GetCertificateAuthorityCsr

Das folgende Java-Beispiel zeigt, wie die GetCertificateAuthorityCsrOperation verwendet wird.

Diese Operation ruft die Zertifikatsignierungsanforderung (CSR) für Ihre private CA ab. Die CSR wird erstellt, wenn Sie die CreateCertificateAuthorityOperation aufrufen. Nehmen Sie die CSR in Ihre lokale X.509-Infrastruktur auf und signieren Sie sie mit Ihrer Stammzertifizierungsstelle oder einer untergeordneten Zertifizierungsstelle. Importieren Sie dann das signierte Zertifikat zurück in ACM PCA, indem Sie den Vorgang aufrufen. ImportCertificateAuthorityCertificate Die CSR wird als Base64-kodierte Zeichenfolge im PEM-Format zurückgegeben.

```
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);
   }
```

```
}
```

Die Ausgabe sollte für die angegebene Zertifizierungsstelle (CA) ähnlich der folgenden sein. Die Zertifikatssignierungsanforderung (CSR) liegt im Base64-kodierten PEM-Format vor. Speichern Sie die CSR in einer lokalen Datei, nehmen Sie sie in Ihre lokale X.509-Infrastruktur auf und signieren Sie sie mit Ihrer Stammzertifizierungsstelle oder einer untergeordneten Zertifizierungsstelle.

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

GetPolicy

Das folgende Java-Beispiel zeigt, wie die GetPolicyOperation verwendet wird.

Der Vorgang ruft die ressourcenbasierte Richtlinie ab, die einer privaten Zertifizierungsstelle zugeordnet ist. Eine ressourcenbasierte Richtlinie wird verwendet, um die kontenübergreifende gemeinsame Nutzung der CA zu ermöglichen. Sie können den ARN einer privaten CA ermitteln, indem Sie die ListCertificateAuthoritiesAktion aufrufen.

Zu den verwandten API-Aktionen gehören PutPolicyund 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

Das folgende Java-Beispiel zeigt, wie die ImportCertificateAuthorityCertificateOperation verwendet wird.

Dieser Vorgang importiert Ihr signiertes privates CA-Zertifikat in AWS Private CA. Bevor Sie diesen Vorgang aufrufen können, müssen Sie die private Zertifizierungsstelle erstellen, indem Sie den CreateCertificateAuthorityVorgang aufrufen. Anschließend müssen Sie eine Zertifikatsignieranforderung (CSR) generieren, indem Sie den GetCertificateAuthorityCsrVorgang aufrufen. Nehmen Sie die CSR in Ihre lokale CA auf und signieren Sie diese unter Verwendung Ihrer Stammzertifizierungsstelle oder einer untergeordneten Zertifizierungsstelle. Erstellen Sie eine Zertifikatskette und kopieren Sie das signierte Zertifikat und die Zertifikatkette in Ihr Arbeitsverzeichnis.

```
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

Das folgende Java-Beispiel zeigt, wie die IssueCertificateOperation verwendet wird.

Bei diesem Vorgang wird Ihre private Zertifizierungsstelle (CA) verwendet, um ein Endzertifizierungszertifikat auszustellen. Diese Operation gibt den Amazon-Ressourcennamen (ARN) des Zertifikats zurück. Sie können das Zertifikat abrufen, indem Sie den aufrufen GetCertificateund den ARN angeben.

> **Note**
>
> Für den IssueCertificateVorgang müssen Sie eine Zertifikatsvorlage angeben. In diesem Beispiel wird die `EndEntityCertificate/V1` Vorlage verwendet. Informationen zu allen verfügbaren Vorlagen finden Sie unterVerwenden Sie Zertifikatsvorlagen AWS Private CA.

```
package com.amazonaws.samples;

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

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

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

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

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

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

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

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

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

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

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

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

      // Set the CA ARN.
```

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

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

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

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

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

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

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

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

Die Ausgabe sollte in etwa wie folgt aussehen:

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

ListCertificateAuthorities

Das folgende Java-Beispiel zeigt, wie die ListCertificateAuthoritiesOperation verwendet wird.

Bei diesem Vorgang werden die privaten Zertifizierungsstellen (CAs) aufgeführt, die Sie mithilfe des CreateCertificateAuthorityVorgangs erstellt haben.

```
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());
    }
}
```

Wenn Sie Zertifizierungsstellen auflisten möchten, sollte Ihre Ausgabe folgendermaßen oder ähnlich aussehen:

```
[{
 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

Das folgende Java-Beispiel zeigt, wie die ListPermissionsOperation verwendet wird.

In dieser Operation werden ggf. die Berechtigungen aufgelistet, die Ihrer privaten Zertifizierungsstelle zugewiesen wurden. Berechtigungen, einschließlich `IssueCertificate`GetCertificate, und`ListPermissions`, können einem AWS Dienstprinzipal mit dem CreatePermissionVorgang zugewiesen und mit dem DeletePermissionsVorgang widerrufen werden.

```
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);
   }
}
```

Wenn die angegebene private Zertifizierungsstelle einem Service-Prinzipal Berechtigungen zugewiesen hat, sollte die Ausgabe in etwa wie folgt aussehen:

```
[{
       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

Das folgende Java-Beispiel zeigt, wie die ListTagsOperation verwendet wird.

Diese Operation listet die Tags auf, sofern vorhanden, die zu Ihrer privaten Zertifizierungsstelle gehören. Tags sind Beschriftungen, mit denen Sie Ihre identifizieren und organisieren können CAs. Jedes Tag besteht aus einem Schlüssel und einem optionalen Wert. Rufen Sie den TagCertificateAuthorityVorgang auf, um Ihrer CA ein oder mehrere Tags hinzuzufügen. Rufen Sie den UntagCertificateAuthorityVorgang zum Entfernen von Tags auf.

```
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);
   }
}
```

Wenn Sie Tags auflisten, sollte Ihre Ausgabe sollte folgendermaßen oder ähnlich aussehen:

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

PutPolicy

Das folgende Java-Beispiel zeigt, wie die PutPolicyOperation verwendet wird.

Der Vorgang fügt einer privaten Zertifizierungsstelle eine ressourcenbasierte Richtlinie hinzu, wodurch die gemeinsame Nutzung von Konten ermöglicht wird. Wenn es durch eine Richtlinie autorisiert ist, kann ein Principal, der sich in einem anderen AWS Konto befindet, private Endentitätszertifikate ausstellen und verlängern, indem er eine private Zertifizierungsstelle verwendet, die ihm nicht gehört. Sie können den ARN einer privaten CA ermitteln, indem Sie die ListCertificateAuthoritiesAktion aufrufen. Beispiele für Richtlinien finden Sie in den AWS Private CA Leitlinien zu ressourcenbasierten Richtlinien.

Sobald eine Richtlinie an eine Zertifizierungsstelle angehängt ist, können Sie sie mit der GetPolicyAktion überprüfen oder sie mit der DeletePolicyAktion löschen.

```
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

Das folgende Java-Beispiel zeigt, wie die RestoreCertificateAuthorityOperation verwendet wird. Eine private Zertifizierungsstelle kann während ihres Wiederherstellungszeitraums jederzeit wiederhergestellt werden. Derzeit kann dieser Zeitraum 7 bis 30 Tage ab Löschdatum betragen. Er kann definiert werden, wenn Sie die Zertifizierungsstelle löschen. Weitere Informationen finden Sie unter Wiedererstellen einer Zertifizierungsstelle. Weitere Informationen finden Sie auch im Java-Beispiel 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

Das folgende Java-Beispiel zeigt, wie die RevokeCertificateOperation verwendet wird.

Dieser Vorgang widerruft ein Zertifikat, das Sie durch den Aufruf des IssueCertificateVorgangs ausgestellt haben. Wenn Sie bei der Erstellung oder Aktualisierung Ihrer privaten Zertifizierungsstelle eine Zertifikatssperrliste (CRL) aktiviert haben, sind Informationen zu den widerrufenen Zertifikaten in der CRL enthalten. AWS Private CA schreibt die CRL in einen Amazon S3 S3-Bucket, den Sie angeben. Weitere Informationen finden Sie in der CrlConfigurationStruktur.

```
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

Das folgende Java-Beispiel zeigt, wie die TagCertificateAuthorityOperation verwendet wird.

Diese Operation fügt Ihrer privaten CA einen oder mehrere Tags hinzu. Tags sind Beschriftungen, mit denen Sie Ihre AWS Ressourcen identifizieren und organisieren können. Jedes Tag besteht aus einem Schlüssel und einem optionalen Wert. Wenn Sie diese Operation aufrufen, geben Sie die private CA über den Amazon-Ressourcennamen (ARN) an. Sie geben den Tag mit einem Schlüssel-Wert-Paar an. Um ein bestimmtes Merkmal dieser CA zu identifizieren, können Sie ein Tag auf nur eine private CA anwenden. Oder um nach einer gemeinsamen Beziehung zwischen diesen zu filtern CAs, können Sie dasselbe Tag auf mehrere private Daten anwenden CAs. Verwenden Sie den UntagCertificateAuthorityVorgang, um ein oder mehrere Tags zu entfernen. Rufen Sie den ListTagsVorgang auf, um zu sehen, welche Tags Ihrer CA zugeordnet sind.

```
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

Das folgende Java-Beispiel zeigt, wie die UntagCertificateAuthorityOperation verwendet wird.

Diese Operation entfernt einen oder mehrere Tags aus Ihrer privaten CA. Ein Tag besteht aus einem Schlüssel-Wert-Paar. Wenn Sie den Wertteil des Tags beim Aufrufen dieser Operation nicht angeben, wird das Tag unabhängig vom Wert entfernt. Wenn Sie einen Wert angeben, wird das Tag nur entfernt, wenn es dem angegebenen Wert zugeordnet ist. Verwenden Sie den TagCertificateAuthorityVorgang, um einer privaten Zertifizierungsstelle Tags hinzuzufügen. Rufen Sie den ListTagsVorgang auf, um zu sehen, welche Tags Ihrer CA zugeordnet sind.

```
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

Das folgende Java-Beispiel zeigt, wie die UpdateCertificateAuthorityOperation verwendet wird.

Die Operation aktualisiert den Status oder die Konfiguration einer privaten Zertifizierungsstelle (Certificate Authority, CA). Ihre private CA muss sich im Status `ACTIVE` oder `DISABLED` befinden, bevor Sie sie aktualisieren können. Sie können eine private CA deaktivieren, die sich im Status `ACTIVE` befindet oder eine CA im Status `DISABLED` erneut aktivieren.

```
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;
      }
   }
```

```
}
```

Erstellen Sie CAs Zertifikate mit benutzerdefinierten Betreffnamen

Das CustomAttributeObjekt ermöglicht es Administratoren, benutzerdefinierte Objektkennungen (OIDs) an private Objekte CAs und Zertifikate zu übergeben. Benutzerdefiniert OIDs kann verwendet werden, um spezielle Hierarchien mit Subjektnamen zu erstellen, die die Struktur und die Bedürfnisse Ihrer Organisation widerspiegeln. Benutzerdefinierte Zertifikate müssen mit einer der Vorlagen erstellt werden. `ApiPassthrough` Weitere Informationen zu Vorlagen finden Sie unter AWS Private CA Vorlagensorten. Weitere Informationen zur Verwendung benutzerdefinierter Attribute finden Sie unter und. Stellen Sie private Endentitätszertifikate aus Erstellen Sie eine private CA in AWS Private CA

Sie können es nicht zusammen mit verwenden`StandardAttributes`. `CustomAttributes` Sie können den Standard jedoch OIDs als Teil eines bestehen`CustomAttributes`. Die standardmäßigen Betreffnamen OIDs sind in der folgenden Tabelle aufgeführt:

Name des Betreffs	Objekt-ID
Land	2.5.4.6
CommonName	2.5.4.3
DistinguishedNameQualifier	2.5.4,46
GenerationQualifier	2.5.4,44
GivenName	2.5.4,42
Initialen	2.5.4,43
Ort	2.5.4.7
Organisation	2.5.4.10
OrganizationalUnit	2.5.4.11
Pseudonym	2.5.4,65
SerialNumber	2.5.4.5

Name des Betreffs	Objekt-ID
Status	2.5.4.8
Nachname	2.5.4.4
Title	2.5.4.12

Themen

- CA erstellen mit CustomAttribute
- Stellen Sie ein Zertifikat aus mit CustomAttribute

CA erstellen mit 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);
    }
}
```

Stellen Sie ein Zertifikat aus mit 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);
    }
}
```

Zertifikate mit benutzerdefinierten Erweiterungen erstellen

Das CustomExtensionObjekt ermöglicht es Administratoren, benutzerdefinierte X.509-Erweiterungen in privaten Zertifikaten festzulegen. Benutzerdefinierte Zertifikate müssen mit einer der ApiPassthrough Vorlagen erstellt werden. Weitere Informationen zu Vorlagen finden Sie unter AWS Private CA Vorlagensorten. Weitere Informationen zur Verwendung benutzerdefinierter Erweiterungen finden Sie unterStellen Sie private Endentitätszertifikate aus.

Themen

- Aktivieren Sie eine untergeordnete Zertifizierungsstelle mit der Erweiterung NameConstraints
- Stellen Sie ein Zertifikat mit der Erweiterung QC Statement aus

Aktivieren Sie eine untergeordnete Zertifizierungsstelle mit der Erweiterung NameConstraints

```
package com.amazonaws.samples;

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

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

```
import java.util.Objects;

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

import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;
import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;
import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateResult;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
```

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

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

import lombok.SneakyThrows;

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

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

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

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

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

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

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

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

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

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

      return client;
  }
```

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

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

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

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

      return rootCertificate;
  }

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

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

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

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

        return subordinateCAArn;
    }

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

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

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

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
```

```
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

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

        return csr;
    }

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

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

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

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

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

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

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

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

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

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

        // Add custom extension to api-passthrough
        ApiPassthrough apiPassthrough = new ApiPassthrough();
        Extensions extensions = new Extensions();
        extensions.setCustomExtensions(Arrays.asList(customExtension));
        apiPassthrough.setExtensions(extensions);
        issueRequest.setApiPassthrough(apiPassthrough);

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

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

        return subordinateCertificateArn;
    }
```

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

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

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

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

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

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

      // Create waiter to wait on successful creation of the certificate file.
      Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
      try {
          getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
      } catch (WaiterUnrecoverableException e) {
          //Explicit short circuit when the recourse transitions into
          //an undesired state.
```

```
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

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

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

        return subordinateCertificate;
    }

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

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

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

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

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
        System.out.println("Subordinate CA certificate successfully imported.");
        System.out.println("Subordinate CA activated successfully.");
    }

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

Stellen Sie ein Zertifikat mit der Erweiterung QC Statement aus

```
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);
    }
}
```

Wird AWS Private CA zur Implementierung von Matter-Zertifikaten verwendet

Sie können die AWS Private Certificate Authority API verwenden, um Zertifikate zu erstellen, die dem Matter-Konnektivitätsstandard entsprechen. Matter spezifiziert Zertifikatkonfigurationen, die die Sicherheit und Konsistenz von IoT-Geräten (Internet of Things) auf mehreren technischen Plattformen verbessern. Weitere Informationen zu Matter finden Sie unter buildwithmatter.com.

Matter 1.2, veröffentlicht im Oktober 2023, unterstützt DAC-Widerruf mithilfe von Zertifikatssperrlisten (). CRLs Um Ihnen zu helfen, dem aktuellen Matter-Standard zu entsprechen, wenn Sie den CRL-Widerruf für CAs diese Ausgabe aktivieren, Themenzertifikate im `CrlConfiguration` Objekt, in der `CrlDistributionPointExtensionConfiguration` Struktur, auf eingestellt`OmitExtension`. `true`

In der Regel sollten Sie den CRL Distribution Point (CDP) in die ausgestellten Zertifikate CAs einbetten, sodass die vertrauenden Parteien, die die Zertifikatskette validieren, die CRL abrufen und den Zertifikatsstatus überprüfen können. In Matter wird der CDP-URI nicht in Zertifikate geschrieben. Stattdessen rufen Benutzer Daten CDPs aus dem Matter Distributed Compliance Ledger (DCL) ab, dem vertrauenswürdigen Matter-Datenspeicher. Sie müssen den CDP-URI in die Matter DCL hochladen, damit er bei der Validierung ermittelt werden kann. DACs Weitere Informationen zur Bestimmung des CDP-URI finden Sie unter. Ermitteln des CRL Distribution Point (CDP) -URI Weitere Informationen zu Matter finden Sie auf der Matter-Standard-Startseite.

Themen

- Aktivieren Sie eine Product Attestation Authority (PAA)
- Aktivieren Sie ein Product Attestation Intermediate (PAI)
- Erstellen Sie ein Device Attestation Certificate (DAC)

- Aktivieren Sie eine Root-CA für Node Operational Certificates (NOC).
- Aktivieren Sie eine untergeordnete Zertifizierungsstelle für Node Operational Certificates (NOC)
- Erstellen Sie ein Node Operational Certificate (NOC)

Aktivieren Sie eine Product Attestation Authority (PAA)

Dieses Java-Beispiel zeigt, wie Sie die Definition von Root CACertificate _ APIPassthrough /V1 Vorlage verwenden, um ein Matter Root CA (PAA) -Zertifikat für die Produktbescheinigung zu erstellen und zu installieren. Die Erweiterung AuthorityKeyIdentifier (AKI) ist optional für. PAAs Um eine AKI festzulegen, müssen Sie einen Base64-codierten AKI-Wert generieren und ihn durch eine übergeben. CustomExtension

Das Beispiel ruft die folgenden API-Aktionen auf: AWS Private CA

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Wenn Sie auf Probleme stoßen, finden Sie Beheben Sie AWS Private CA Matter-konforme Zertifikatsfehler weitere Informationen im Abschnitt Fehlerbehebung.

```
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);
    }
}
```

Aktivieren Sie ein Product Attestation Intermediate (PAI)

Dieses Java-Beispiel zeigt, wie Sie die BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1Definition Vorlage verwenden, um ein PAI-Zertifikat (Matter Subordinate CA) für die Produktbescheinigung zu erstellen und zu installieren. Sie müssen einen Base64-codierten KeyUsage Wert generieren und ihn über einen übergeben. CustomExtension

Das Beispiel ruft die folgenden API-Aktionen auf: AWS Private CA

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

Wenn Sie auf Probleme stoßen, finden Sie Beheben Sie AWS Private CA Matter-konforme Zertifikatsfehler weitere Informationen im Abschnitt Fehlerbehebung.

```
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);
    }
}
```

Erstellen Sie ein Device Attestation Certificate (DAC)

Dieses Java-Beispiel zeigt, wie Sie die Vorlage BlankEndEntityCertificate_ CriticalBasicConstraints _ APIPassthrough /V1 verwenden, um ein Matter Device Attestation-Zertifikat zu erstellen. Sie müssen einen Base64-codierten KeyUsage Wert generieren und ihn über einen übergeben. CustomExtension

Das Beispiel ruft die folgende API-Aktion auf: AWS Private CA

- IssueCertificate

Wenn Sie auf Probleme stoßen, finden Sie Beheben Sie AWS Private CA Matter-konforme Zertifikatsfehler weitere Informationen im Abschnitt Fehlerbehebung.

```
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);
    }
}
```

Aktivieren Sie eine Root-CA für Node Operational Certificates (NOC).

Dieses Java-Beispiel zeigt, wie Sie die Definition von Root CACertificate _ APIPassthrough / V1 Vorlage verwenden, um ein Matter Root-CA-Zertifikat für die Ausstellung zu erstellen und zu installieren NOCs. Die Erweiterung AuthorityKeyIdentifier (AKI) ist für NOC Root CA-Zertifikate optional. Um eine AKI festzulegen, müssen Sie einen Base64-codierten AKI-Wert generieren und ihn über einen übergeben. CustomExtension

Das Beispiel ruft die folgenden API-Aktionen auf: AWS Private CA

- CreateCertificateAuthority
- GetCertificateAuthorityCsr

- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Wenn Sie auf Probleme stoßen, finden Sie Beheben Sie AWS Private CA Matter-konforme Zertifikatsfehler weitere Informationen im Abschnitt Fehlerbehebung.

```
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);
    }
}
```

Aktivieren Sie eine untergeordnete Zertifizierungsstelle für Node Operational Certificates (NOC)

Dieses Java-Beispiel zeigt, wie die BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1Definition Vorlage verwendet wird, um ein Zertifikat der untergeordneten Matter CA auszustellen und zu installieren. NOCs Sie müssen einen Base64-codierten KeyUsage Wert generieren und ihn über einen übergeben. CustomExtension

Das Beispiel ruft die folgenden API-Aktionen auf: AWS Private CA

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

Falls Probleme auftreten, finden Sie Beheben Sie AWS Private CA Matter-konforme Zertifikatsfehler weitere Informationen im Abschnitt Fehlerbehebung.

```
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);
    }
}
```

Erstellen Sie ein Node Operational Certificate (NOC)

Dieses Java-Beispiel zeigt, wie Sie mit der Vorlage BlankEndEntityCertificate_ CriticalBasicConstraints _ APIPassthrough /V1 ein Matter Node Operational Certificate erstellen. Sie müssen einen Base64-codierten KeyUsage Wert generieren und ihn über einen übergeben. CustomExtension

Das Beispiel ruft die folgende API-Aktion auf: AWS Private CA

- IssueCertificate

Wenn Sie auf Probleme stoßen, finden Sie Beheben Sie AWS Private CA Matter-konforme Zertifikatsfehler weitere Informationen im Abschnitt Fehlerbehebung.

```
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);
    }
}
```

Wird AWS Private CA zur Implementierung von MdL-Zertifikaten verwendet

Sie können die AWS Private Certificate Authority API verwenden, um Zertifikate zu erstellen, die dem ISO/IEC-Standard für den mobilen Führerschein (MdL) entsprechen. Dieser Standard legt Schnittstellenspezifikationen für die Implementierung eines Führerscheins in Verbindung mit einem Mobilgerät fest, einschließlich Zertifikatkonfigurationen.

Themen

- Aktivieren Sie ein Zertifikat der ausstellenden Zertifizierungsstelle (IACA)
- Erstellen Sie ein Zertifikat für den Unterzeichner eines Dokuments

Aktivieren Sie ein Zertifikat der ausstellenden Zertifizierungsstelle (IACA)

Dieses Java-Beispiel zeigt, wie die BlankRootCACertificate_ PathLen 0_ /V1-Definition APIPassthrough Vorlage verwendet wird, um ein dem ISO/IEC MdL-Standard entsprechendes Zertifikat einer ausstellenden Zertifizierungsstelle (IACA) zu erstellen und zu installieren. Sie müssen Base64-kodierte Werte für`KeyUsage`,, `IssuerAlternativeName` und generieren und sie weitergeben. `CRLDistributionPoint` `CustomExtensions`

> **Note**
>
> Das IACA-Linkzertifikat richtet einen Vertrauenspfad vom alten IACA-Stammzertifikat zum neuen IACA-Stammzertifikat ein. Die ausstellende Behörde kann während des IACA-Re-Key-Vorgangs ein IACA-Linkzertifikat generieren und verteilen. Sie können kein

IACA-Linkzertifikat ausstellen, indem Sie ein IACA-Stammzertifikat mit Satz verwenden. `pathLen=0`

Das Beispiel ruft die folgenden AWS Private CA API-Aktionen auf:

- 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);
    }
}
```

Erstellen Sie ein Zertifikat für den Unterzeichner eines Dokuments

Dieses Java-Beispiel zeigt, wie Sie mit der Vorlage BlankEndEntityCertificate_ APIPassthrough /V1 ein dem ISO/IEC MdL-Standard entsprechendes Zertifikat für den Unterzeichner von Dokumenten erstellen können. Sie müssen Base64-kodierte Werte für, und generieren und sie weitergeben. `KeyUsage` `IssuerAlternativeName` `CRLDistributionPoint` `CustomExtensions`

Das Beispiel ruft die folgende API-Aktion auf: AWS Private CA

- 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);
    }
}
```

Entwickeln Sie Ihre Lösung für AWS Private CA

AWS Private CA bietet Ihnen die vollständige, cloudbasierte Kontrolle über die private PKI (Public Key Infrastructure) Ihres Unternehmens, die sich von einer Stammzertifizierungsstelle (CA) über untergeordnete CAs Zertifikate bis hin zu Endentitätszertifikaten erstreckt. Eine gründliche Planung ist unerlässlich für eine PKI, die sicher, wartbar, erweiterbar und auf die Anforderungen Ihrer Organisation abgestimmt ist. Dieser Abschnitt enthält Anweisungen zum Entwerfen einer CA-Hierarchie, zum Verwalten Ihrer privaten CA und der Lebenszyklen von privaten Endentitätszertifikaten und zum Anwenden von bewährten Methoden für die Sicherheit.

In diesem Abschnitt wird beschrieben, wie Sie sich auf AWS Private CA die Nutzung vorbereiten, bevor Sie eine private Zertifizierungsstelle (CA) einrichten. Außerdem wird die Option erklärt, Sperrunterstützung über das Online Certificate Status Protocol (OCSP) oder eine Zertifikatsperrliste (CRL) hinzuzufügen.

Darüber hinaus sollten Sie festlegen, ob Ihre Organisation es vorzieht, ihre privaten Root-CA-Anmeldeinformationen lokal zu hosten, anstatt sie mit anderen zu hosten. AWS In diesem Fall müssen Sie vor der Verwendung eine selbstverwaltete private PKI einrichten und sichern. AWS Private CA In diesem Szenario erstellen Sie dann eine untergeordnete Zertifizierungsstelle, die von einer übergeordneten Zertifizierungsstelle außerhalb von AWS Private CA unterstützt wird. AWS Private CA Weitere Informationen finden Sie unter Installation eines untergeordneten Zertifizierungsstellenzertifikats, das von einer externen übergeordneten Zertifizierungsstelle signiert wurde.

Topics

Entwerfen Sie eine CA-Hierarchie

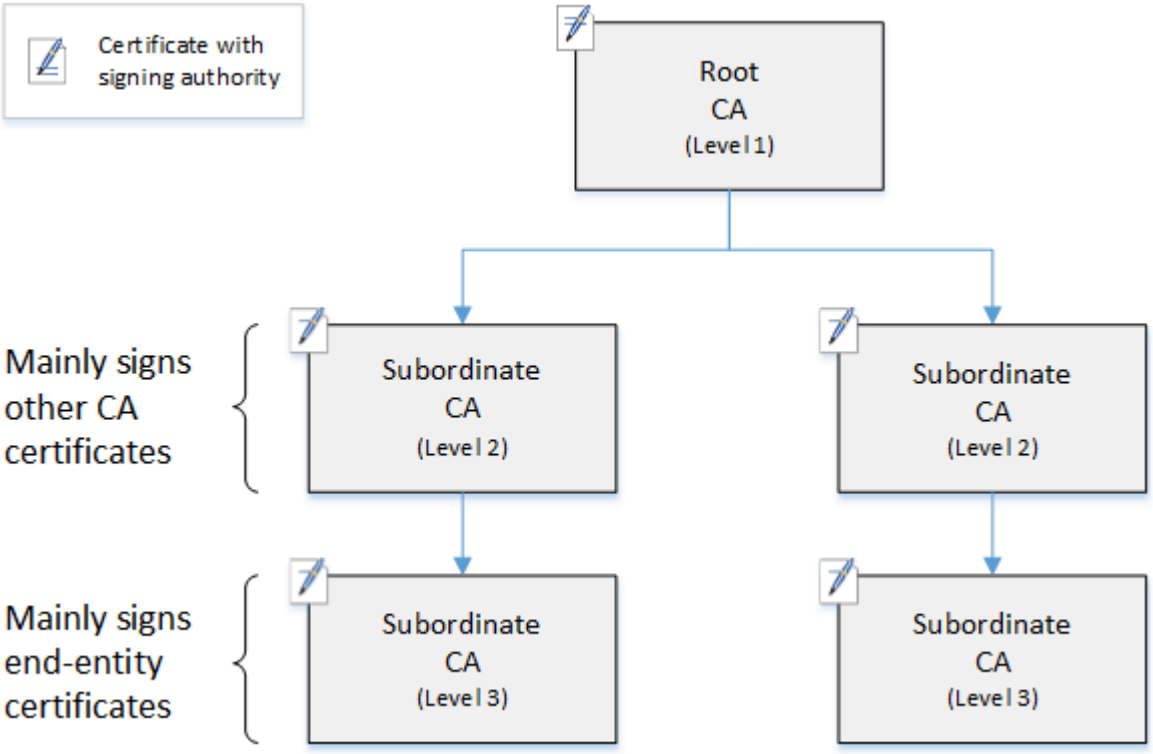
Mit AWS Private CA können Sie eine Hierarchie von Zertifizierungsstellen mit bis zu fünf Ebenen erstellen. Die Stamm-CA im oberen Bereich einer Hierarchiestruktur kann eine beliebige Anzahl

von Verzweigungen aufweisen. Die Stammzertifizierungsstelle kann in jeder Filiale bis zu vier untergeordnete CAs Ebenen haben. Sie können auch mehrere Hierarchien erstellen, jede mit einem eigenen Stamm.

Eine gut gestaltete CA-Hierarchie bietet folgende Vorteile:

- Detaillierte Sicherheitskontrollen, die für die einzelnen CAs geeignet sind
- Aufteilung von administrativen Aufgaben für besseren Lastausgleich und Sicherheit
- Verwendung von CAs mit begrenztem, widerrufbarem Vertrauen für den täglichen Betrieb
- Gültigkeitszeiträume und Zertifikatspfadbeschränkungen

Das folgende Diagramm veranschaulicht eine einfache CA-Hierarchie mit drei Ebenen.



Jede Zertifizierungsstelle in der Struktur wird durch ein X.509 v3-Zertifikat mit Signaturberechtigung (symbolisiert durch das Symbol) unterstützt. pen-and-paper Das bedeutet, dass sie andere ihnen untergeordnete Zertifikate signieren können. CAs Wenn eine CA das Zertifikat einer untergeordneten CA signiert, verleiht sie dem signierten Zertifikat eine eingeschränkte, widerrufliche Autorität. Die Stamm-CA auf Ebene 1 signiert untergeordnete CA-Zertifikate auf Ebene 2. Diese CAs wiederum signieren Zertifikate für CAs Stufe 3, die von PKI-Administratoren (Public Key Infrastructure) verwendet werden, die Endzertifikate verwalten.

Sicherheit in einer CA-Hierarchie sollte so konfiguriert werden, dass sie im oberen Bereich der Struktur am stärksten ist. Diese Anordnung schützt das Stamm-CA-Zertifikat und seinen privaten Schlüssel. Die Stammzertifizierungsstelle verankert die Vertrauensstellung für alle untergeordneten Zertifikate CAs und die untergeordneten Entitätszertifikate. Während lokaler Schaden durch die Kompromittierung eines Endentitätszertifikats entstehen kann, zerstört die Kompromittierung des Stammes die Vertrauensstellung in der gesamten PKI. Stammzertifikate und untergeordnete Zertifikate auf hoher Ebene CAs werden nur selten verwendet (normalerweise zum Signieren anderer CA-Zertifikate). Folglich werden sie streng kontrolliert und geprüft, um ein geringeres Risiko von Kompromittierungen zu gewährleisten. Auf den unteren Ebenen der Hierarchie ist die Sicherheit weniger restriktiv. Dieser Ansatz ermöglicht die routinemäßigen Verwaltungsaufgaben beim Ausstellen und Widerrufen von Endentitätszertifikaten für Benutzer, Computer-Hosts und Softwaredienste.

> **Note**
>
> Die Verwendung einer Stamm-CA zum Signieren eines untergeordneten Zertifikats ist ein seltenes Ereignis, das nur unter einer Handvoll von Umständen auftritt:
>
> - Wenn die PKI erstellt wird
> - Wenn eine CA auf hoher Ebene ersetzt werden muss
> - Wenn eine Zertifikatssperrliste (CRL) oder ein OCSP-Responder (Online Certificate Status Protocol) konfiguriert werden muss
>
> Root- und andere CAs High-Level-Zertifikate erfordern hochsichere Betriebsprozesse und Zugriffskontrollprotokolle.

Themen

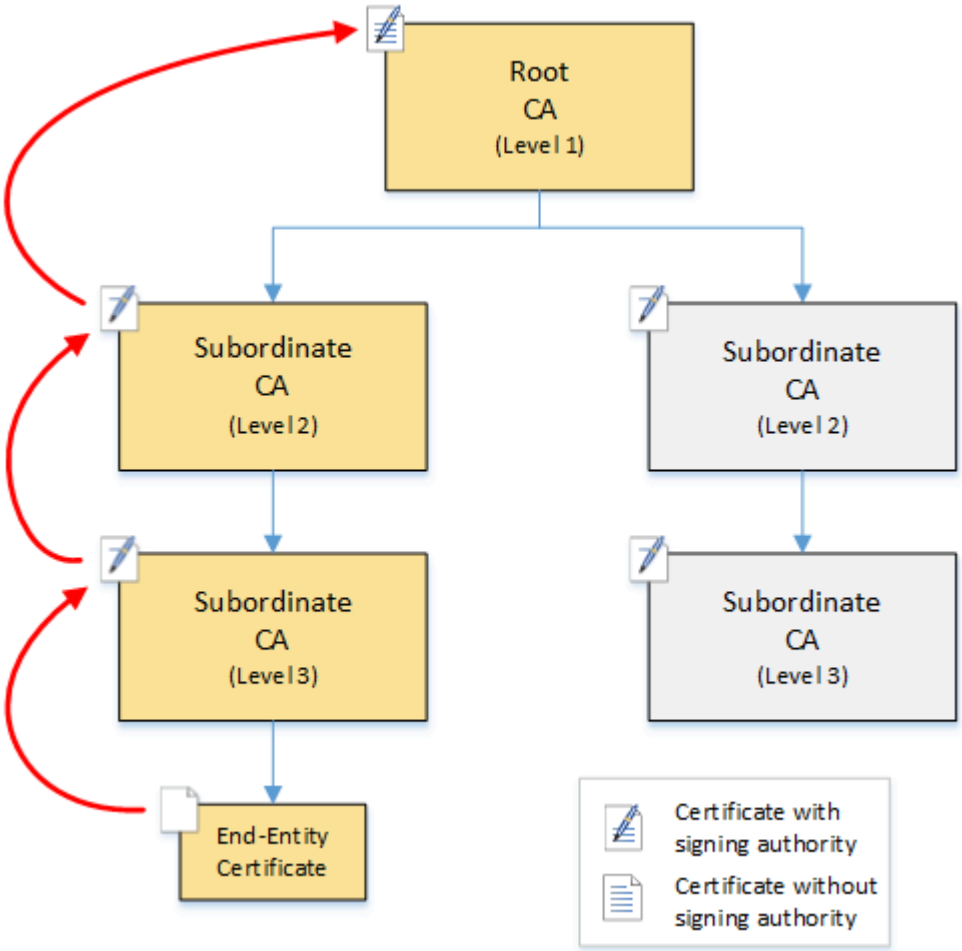
Validieren Sie die Zertifikate der Endunternehmen

Endzertifikate erhalten ihr Vertrauen aus einem Zertifizierungspfad, der über die untergeordnete Zertifizierungsstelle zurück CAs zu einer Stammzertifizierungsstelle führt. Wenn einem Webbrowser

oder einem anderen Client ein Endentitätszertifikat präsentiert wird, versucht er, eine Vertrauenskette zu erstellen. Beispielsweise überprüft er etwa, ob der definierte Name (Distinguished Name) des Ausstellers und der definierte Name des Subjekts des Zertifikats mit den entsprechenden Feldern des ausstellenden CA-Zertifikats übereinstimmen. Der Abgleich würde dann auf jeder aufeinanderfolgenden Ebene in der Hierarchie nach oben fortgesetzt, bis der Client eine vertrauenswürdige Stamm-CA erreicht, die in seinem Trust Store (Vertrauensspeicher) enthalten ist.

Der Trust Store ist eine vertrauenswürdige Bibliothek, die der Browser oder CAs das Betriebssystem enthält. Bei einer privaten PKI muss die IT Ihrer Organisation sicherstellen, dass jeder Browser oder jedes System die private Stamm-CA vorab dem Vertrauensspeicher hinzugefügt hat. Andernfalls kann der Zertifizierungspfad nicht validiert werden, was zu Client-Fehlern führt.

Das nächste Diagramm zeigt den Validierungspfad, dem ein Browser folgt, wenn ein X.509-Endentitätszertifikat vorliegt. Beachten Sie, dass das Endentitätszertifikat keine Signaturautorität besitzt und nur dazu dient, die Entität zu authentifizieren, die es besitzt.



Der Browser überprüft das Endentitätszertifikat. Der Browser stellt fest, dass das Zertifikat eine Signatur von der untergeordneten CA (Ebene 3) als vertrauenswürdige Anmeldeinformationen anbietet. Die Zertifikate für das untergeordnete Unternehmen CAs müssen in derselben PEM-Datei enthalten sein. Alternativ können sie sich auch in einer separaten Datei befinden, die die Zertifikate enthält, aus denen die Vertrauenskette besteht. Sind diese gefunden, überprüft der Browser das Zertifikat der untergeordneten CA (Ebene 3) und stellt fest, dass es eine Signatur von der untergeordneten CA (Ebene 2) bietet. Die untergeordnete CA (Ebene 2) bietet wiederum eine Signatur von der Stamm-CA (Ebene 1) als vertrauenswürdige Anmeldeinformationen. Wenn der Browser eine Kopie des privaten Stamm-CA-Zertifikats findet, das in seinem Vertrauensspeicher vorinstalliert ist, validiert er das Endentitätszertifikat als vertrauenswürdig.

In der Regel überprüft der Browser auch jedes Zertifikat anhand einer Zertifikatsperrliste (CRL). Ein abgelaufenes, gesperrtes oder falsch konfiguriertes Zertifikat wird abgelehnt und die Validierung schlägt fehl.

Planen Sie die Struktur einer CA-Hierarchie

Im Allgemeinen sollte Ihre CA-Hierarchie die Struktur Ihrer Organisation widerspiegeln. Streben Sie eine Pfadlänge (d. h. die Anzahl der CA-Ebenen) an, die nicht größer ist als für die Delegierung von Verwaltungs- und Sicherheitsrollen erforderlich. Das Hinzufügen einer CA zur Hierarchie bedeutet, die Anzahl der Zertifikate im Zertifizierungspfad zu erhöhen, was die Validierungsdauer erhöht. Wenn Sie die Pfadlänge auf ein Minimum beschränken, wird auch die Anzahl der Zertifikate reduziert, die bei der Validierung eines Endzertifikats vom Server an den Client gesendet werden.

Theoretisch kann eine Stammzertifizierungsstelle, die keinen pathLenConstraintParameter hat, eine unbegrenzte Anzahl von untergeordneten Zertifizierungsstellen autorisieren. CAs Eine untergeordnete Zertifizierungsstelle kann so viele untergeordnete Zertifizierungsstellen haben, CAs wie es ihre interne Konfiguration zulässt. AWS Private CA verwaltete Hierarchien unterstützen CA-Zertifizierungspfade mit einer Tiefe von bis zu fünf Ebenen.

Gut gestaltete CA-Strukturen haben mehrere Vorteile:

- Separate Verwaltungskontrollen für verschiedene Organisationseinheiten
- Die Fähigkeit, den Zugriff an untergeordnete Personen zu delegieren CAs
- Eine hierarchische Struktur, die höhere Ebenen durch zusätzliche Sicherheitskontrollen CAs schützt

Zwei häufige CA-Strukturen erfüllen all dies:

- Zwei CA-Ebenen: Stamm-CA und untergeordnete CA

  Dies ist die einfachste CA-Struktur, die separate Verwaltungs-, Kontroll- und Sicherheitsrichtlinien für die Stamm-CA und eine untergeordnete CA ermöglicht. Sie können restriktive Kontrollen und Richtlinien für Ihre Stamm-CA aufrechterhalten und gleichzeitig der untergeordneten CA weitreichenderen Zugriff gewähren. Letzteres wird für die Massenausgabe von Endentitätszertifikaten verwendet.

- Drei CA-Ebenen: Stamm-CA und zwei Ebenen untergeordneter CAs

  Ähnlich wie oben fügt diese Struktur eine zusätzliche CA-Ebene hinzu, um die Stamm-CA weiter von CA-Operationen auf niedriger Ebene zu trennen. Die mittlere CA-Schicht wird nur zum Signieren untergeordneter Stellen verwendet, die für CAs die Ausstellung von Endzertifikaten zuständig sind.

Zu den selteneren CA-Strukturen gehören die folgenden:

- Vier oder mehr CA-Ebenen

  Obwohl weniger häufig als Hierarchien mit drei Ebenen, sind CA-Hierarchien mit vier oder mehr Ebenen möglich und unter Umständen erforderlich, um die Delegierung administrativer Aufgaben zu ermöglichen.
- Eine CA-Ebene: nur Stamm-CA

  Diese Struktur wird häufig für die Entwicklung und das Durchführen von Tests verwendet, wenn keine vollständige Vertrauenskette erforderlich ist. Ihre Verwendung in der Produktion ist untypisch. Darüber hinaus verstößt sie gegen die bewährte Methode, separate Sicherheitsrichtlinien für die Stammzertifizierungsstelle und die Zertifizierungsstellen, die Endzertifikate ausstellen CAs , aufrechtzuerhalten.

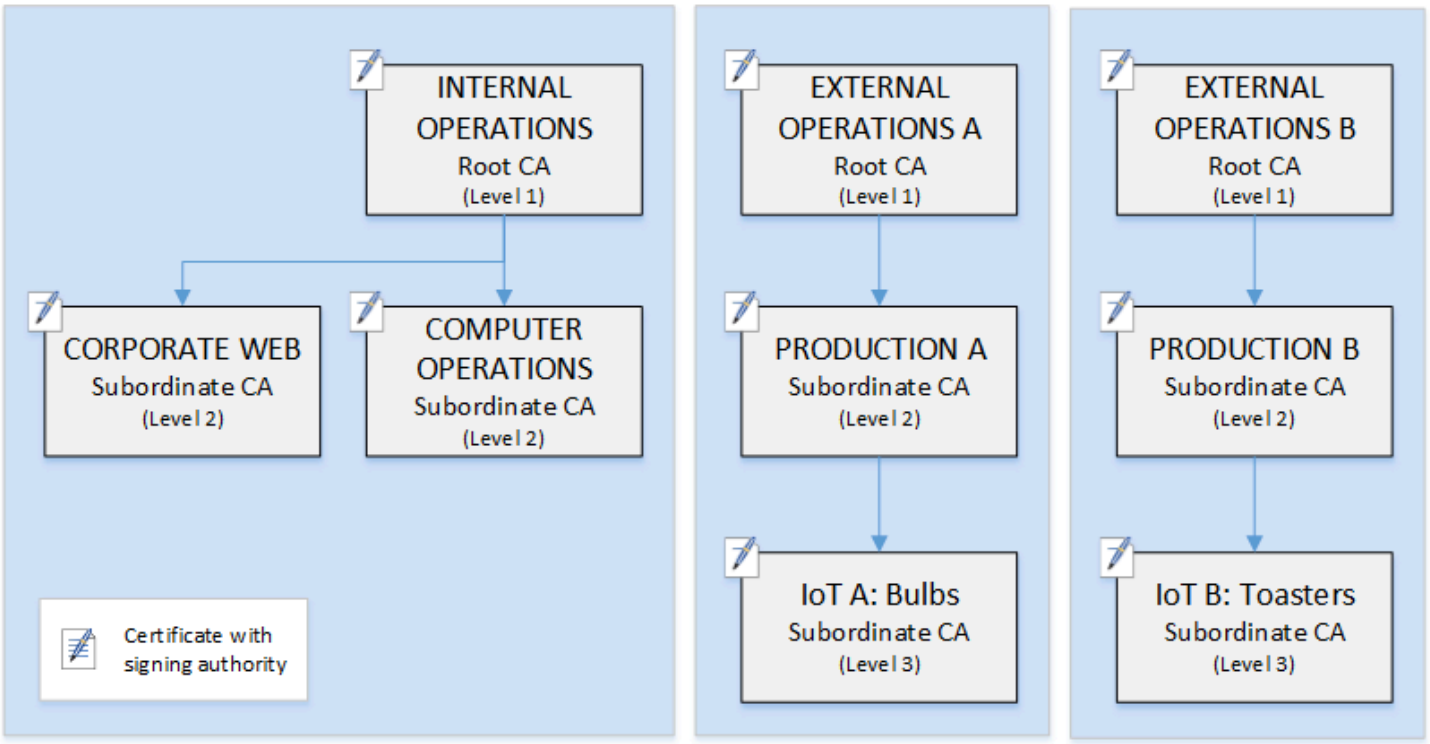
  Wenn Sie jedoch bereits Zertifikate direkt von einer Stammzertifizierungsstelle ausstellen, können Sie zu dieser migrieren. AWS Private CA Dies bietet Sicherheits- und Kontrollvorteile gegenüber der Verwendung einer Root-CA, die mit OpenSSL oder anderer Software verwaltet wird.

Beispiel für eine private PKI für einen Hersteller

In diesem Beispiel stellt ein hypothetisches Technologieunternehmen zwei IoT-Produkte (Internet of Things) her, eine intelligente Glühbirne und einen intelligenten Toaster. Während der Produktion

wird für jedes Gerät ein Endentitätszertifikat ausgestellt, damit es sicher über das Internet mit dem Hersteller kommunizieren kann. Die PKI des Unternehmens sichert auch seine Computerinfrastruktur, einschließlich der internen Website und verschiedener selbst gehosteter Computerdienste, die Finanz- und Geschäftsabläufe ausführen.

Folglich ist die CA-Hierarchie eng an diesen administrativen und operativen Aspekten des Geschäfts ausgerichtet.



Diese Hierarchie enthält drei Stamm-CAs, eine für interne Vorgänge und zwei für externe Vorgänge (eine Stamm-CA für jede Produktlinie). Es zeigt auch mehrere Zertifizierungspfade mit zwei Zertifizierungsstufen für interne Abläufe und drei Stufen für externe Abläufe.

Die Verwendung getrennter Stamm CAs - und zusätzlicher untergeordneter CA-Ebenen für externe Operationen ist eine Designentscheidung, die den Geschäfts- und Sicherheitsanforderungen gerecht wird. Mit mehreren CA-Strukturen ist die PKI zukunftssicher im Fall von Unternehmensumstrukturierungen, Veräußerungen oder Akquisitionen. Wenn Änderungen auftreten, kann eine gesamte CA-Stammhierarchie mit der Abteilung, die sie sichert, ordnungsgemäß bewegt werden. Und da es zwei Ebenen untergeordneter Zertifizierungsstellen gibt, CAs sind die Stammzertifizierungsstellen weitgehend von der Ebene 3 isoliert, die für CAs die Massensignierung der Zertifikate für Tausende oder Millionen von hergestellten Produkten zuständig ist.

Auf der internen Seite vervollständigen Internet- und interne Computervorgänge des Unternehmens eine Hierarchie mit zwei Ebenen. Diese Ebenen ermöglichen es Webadministratoren und Betriebsingenieuren, die Zertifikatausstellung unabhängig für ihre eigenen Geschäftsdomänen zu verwalten. Die Aufteilung der PKI in verschiedene funktionale Domänen ist eine bewährte Methode im Bereich Sicherheit und schützt sie vor einer Kompromittierung, die sich auf die jeweils andere auswirken könnte. Webadministratoren stellen Endentitätszertifikate zur Verwendung durch Webbrowser im gesamten Unternehmen aus und authentifizieren und verschlüsseln so die Kommunikation auf der internen Website. Betriebsingenieure stellen Endentitätszertifikate aus, die Rechenzentrums-Hosts und Computerdienste gegenseitig authentifizieren. Dieses System trägt dazu bei, vertrauliche Daten zu schützen, indem es sie im LAN verschlüsselt.

Legen Sie Längenbeschränkungen für den Zertifizierungspfad fest

Die Struktur einer CA-Hierarchie wird durch die Erweiterung der Basiseinschränkungen definiert und erzwungen, die jedes Zertifikat enthält. Die Erweiterung definiert zwei Einschränkungen:

- `cA`— Ob das Zertifikat eine Zertifizierungsstelle definiert. Wenn dieser Wert false lautet (der Standardwert), ist das Zertifikat ein Endentitätszertifikat.
- `pathLenConstraint`— Die maximale Anzahl untergeordneter Personen CAs , die in einer gültigen Vertrauenskette existieren können. Das Zertifikat der Endeinheit wird nicht gezählt, da es sich nicht um ein CA-Zertifikat handelt.

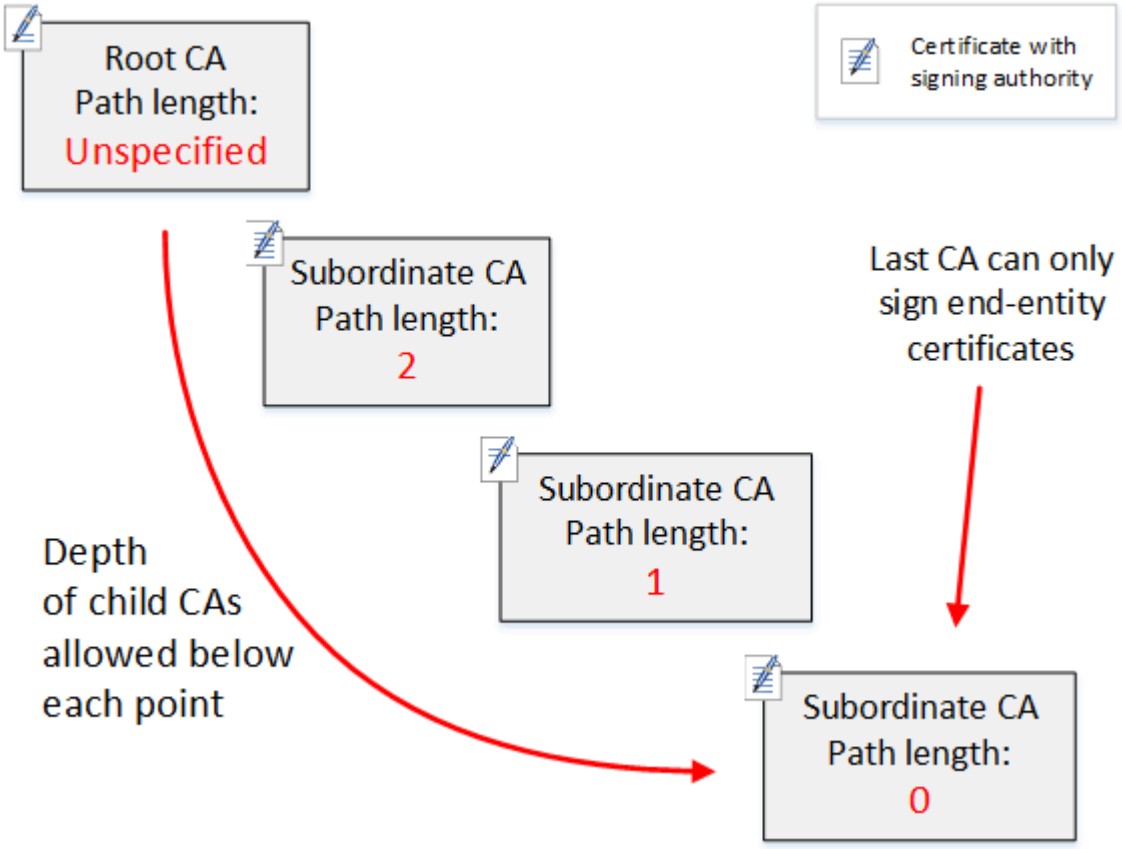
Ein Stamm-CA-Zertifikat erfordert maximale Flexibilität und enthält keine Pfadlängenbeschränkung. Dadurch kann der Stamm einen Zertifizierungspfad beliebiger Länge definieren.

> **Note**
>
> AWS Private CA begrenzt den Zertifizierungspfad auf fünf Stufen.

Untergeordnete Elemente CAs haben `pathLenConstraint` Werte, die gleich oder größer als Null sind, abhängig von der Position in der Hierarchie, der Platzierung und den gewünschten Funktionen. In einer Hierarchie mit drei CAs ist beispielsweise keine Pfadbeschränkung für die Stammzertifizierungsstelle angegeben. Die erste untergeordnete Zertifizierungsstelle hat eine Pfadlänge von 1 und kann daher eine untergeordnete CAs Zertifizierungsstelle signieren. Jedes dieser untergeordneten CAs Elemente muss unbedingt den `pathLenConstraint` Wert Null haben. Dies bedeutet, dass sie Endentitätszertifikate signieren, jedoch keine zusätzlichen CA-Zertifikate

ausstellen können. Die Beschränkung der Fähigkeit, Neues zu schaffen, CAs ist eine wichtige Sicherheitskontrolle.

Das folgende Diagramm veranschaulicht diese Verbreitung von eingeschränkter Autorität in der Hierarchie.



In dieser Hierarchie mit vier Ebenen ist die Stamm-CA nicht eingeschränkt (wie immer). Die erste untergeordnete Zertifizierungsstelle hat jedoch einen `pathLenConstraint` Wert von 2, wodurch ihr untergeordnetes Objekt daran CAs gehindert wird, mehr als zwei Stufen tiefer vorzudringen. Folglich muss der Einschränkungswert für einen gültigen Zertifizierungspfad in den nächsten beiden Ebenen auf Null verringert werden. Wenn ein Webbrowser ein Endentitätszertifikat aus diesem Zweig vorfindet, das eine Pfadlänge größer als vier hat, schlägt die Validierung fehl. Ein solches Zertifikat könnte auf eine versehentlich erstellte CA, eine falsch konfigurierte CA oder eine nicht autorisierte Ausstellung zurückzuführen sein.

Verwalten Sie die Pfadlänge mit Vorlagen

AWS Private CA stellt Vorlagen für die Ausstellung von Stamm-, untergeordneten und Endentitätszertifikaten bereit. Diese Vorlagen fassen bewährte Methoden für die grundlegenden

Einschränkungswerte, einschließlich der Pfadlänge, zusammen. Die Vorlagen umfassen die folgenden:

- Root /V1 CACertificate
- Untergeordnet _ 0/V1 CACertificate PathLen
- Untergeordnet _ 1/V1 CACertificate PathLen
- Untergeordnet _ 2/V1 CACertificate PathLen
- Untergeordnet _ 3/V1 CACertificate PathLen
- EndEntityCertificate/V1

Die `IssueCertificate`-API gibt einen Fehler zurück, wenn Sie versuchen, eine CA mit einer Pfadlänge zu erstellen, die größer oder gleich der Pfadlänge des ausstellenden CA-Zertifikats ist.

Weitere Informationen zu Zertifikatvorlagen finden Sie unter Verwenden Sie Zertifikatsvorlagen AWS Private CA.

Automatisieren Sie die Einrichtung der CA-Hierarchie mit AWS CloudFormation

Wenn Sie sich für ein Design für Ihre CA-Hierarchie entschieden haben, können Sie es testen und mithilfe einer AWS CloudFormation Vorlage in Produktion nehmen. Ein Beispiel für eine solche Vorlage finden Sie unter Deklarieren einer privaten CA-Hierarchie im CloudFormation - Benutzerhandbuch.

Verwalten Sie den privaten CA-Lebenszyklus

CA-Zertifikate haben eine feste Lebensdauer bzw. einen festen Gültigkeitszeitraum. Wenn ein CA-Zertifikat abläuft, werden alle Zertifikate, die direkt oder indirekt von einer CAs untergeordneten Stelle in der CA-Hierarchie ausgestellt wurden, ungültig. Sie können den Ablauf von CA-Zertifikaten vermeiden, indem Sie im Voraus planen.

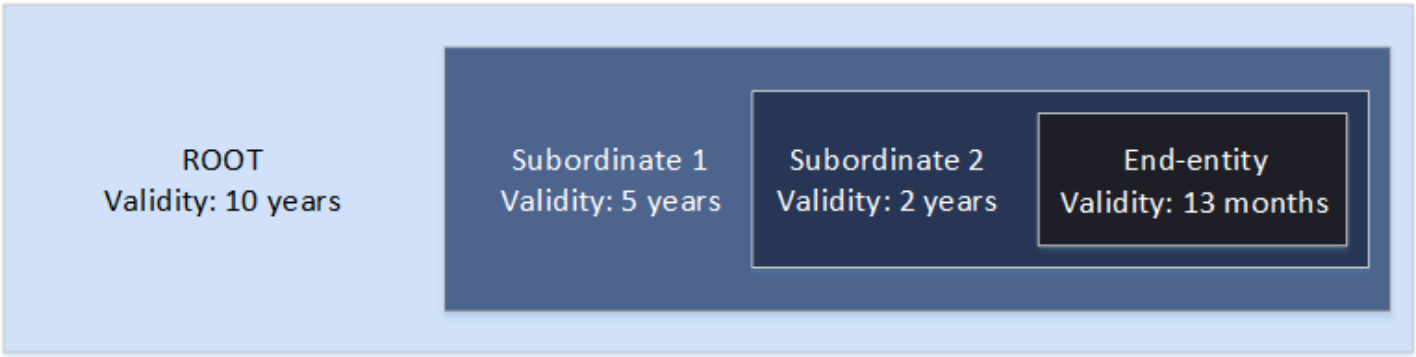
Wählen Sie Gültigkeitszeiträume

Der Gültigkeitszeitraum eines X.509-Zertifikats ist ein erforderliches grundlegendes Zertifikatfeld. Er bestimmt den Zeitraum, in dem die ausstellende CA bescheinigt, dass das Zertifikat vertrauenswürdig ist, mit Ausnahme einer Sperre. (Ein Stammzertifikat, das selbst signiert wird, bescheinigt seinen eigenen Gültigkeitszeitraum.)

AWS Private CA und AWS Certificate Manager unterstützen Sie bei der Konfiguration der Gültigkeitszeiträume von Zertifikaten, wobei die folgenden Einschränkungen gelten:

- Ein von verwaltetes Zertifikat AWS Private CA muss eine Gültigkeitsdauer haben, die kürzer oder gleich der Gültigkeitsdauer der Zertifizierungsstelle ist, die es ausgestellt hat. Mit anderen Worten, untergeordnete Zertifikate CAs und Endentitätszertifikate können ihre übergeordneten Zertifikate nicht überdauern. Der Versuch, die `IssueCertificate`-API zum Ausstellen eines CA-Zertifikats mit einem Gültigkeitszeitraum größer oder gleich der CA der übergeordneten CA zu verwenden, schlägt fehl.
- Zertifikate, die von ausgestellt und verwaltet werden AWS Certificate Manager (diejenigen, für die ACM den privaten Schlüssel generiert), haben eine Gültigkeitsdauer von 13 Monaten (395 Tagen). ACM verwaltet den Verlängerungsprozess für diese Zertifikate. Wenn Sie Zertifikate direkt AWS Private CA ausstellen, können Sie einen beliebigen Gültigkeitszeitraum wählen.

Das folgende Diagramm zeigt eine typische Konfiguration von verschachtelten Gültigkeitszeiträumen. Das Stammzertifikat hat die längste Lebensdauer, Endzertifikate sind relativ kurzlebig, und untergeordnete Zertifikate liegen CAs zwischen diesen Extremen.



Bestimmen Sie beim Planen der CA-Hierarchie die optimale Lebensdauer Ihrer CA-Zertifikate. Arbeiten Sie ab der gewünschten Lebensdauer der Endentitätszertifikate, die Sie ausstellen möchten, rückwärts.

Endentitätszertifikate

Endentitätszertifikate sollten über einen dem Anwendungsfall entsprechenden Gültigkeitszeitraum verfügen. Eine kurze Lebensdauer minimiert das Risiko für ein Zertifikat in dem Fall, dass sein privater Schlüssel verloren geht oder gestohlen wird. Kurze Lebensdauern bedeuten jedoch häufige Erneuerungen. Wenn ein abgelaufenes Zertifikat nicht erneuert wird, kann es zu Ausfallzeiten kommen.

Die verteilte Verwendung von Endentitätszertifikaten kann auch logistische Probleme darstellen, wenn es zu einer Sicherheitsverletzung kommt. In Ihrer Planung sollten Erneuerungs- und Verteilungszertifikate, das Sperren von kompromittierten Zertifikaten und die Schnelligkeit der Verbreitung von Sperren auf Clients, die auf die Zertifikate angewiesen sind, berücksichtigt werden.

Die Standardgültigkeitsdauer für ein über ACM ausgestelltes Endzertifizierungszertifikat beträgt 13 Monate (395 Tage). In können Sie die `IssueCertificate` API verwenden AWS Private CA, um einen beliebigen Gültigkeitszeitraum anzuwenden, sofern dieser kürzer ist als der der ausstellenden Zertifizierungsstelle.

Untergeordnete CA-Zertifikate

Untergeordnete CA-Zertifikate sollten wesentlich längere Gültigkeitszeiträume haben als die von ihnen ausgestellten Zertifikate. Ein guter Bereich für die Gültigkeit eines CA-Zertifikats ist das zwei- bis fünffache des Zeitraums eines untergeordneten CA-Zertifikats oder eines Endentitätszertifikats. Angenommen, Sie haben eine CA-Hierarchie mit zwei Ebenen (Stamm-CA und eine untergeordnete CA). Wenn Sie Endentitätszertifikate mit einer Laufzeit von einem Jahr ausstellen möchten, können Sie die Lebensdauer der untergeordneten ausstellenden CA auf drei Jahre konfigurieren. Dies ist die Standardgültigkeitsdauer für ein untergeordnetes CA-Zertifikat in AWS Private CA. Untergeordnete CA-Zertifikate können geändert werden, ohne das Stamm-CA-Zertifikat zu ersetzen.

Stammzertifikate

Änderungen an einem Stamm-CA-Zertifikat wirken sich auf die gesamte PKI (Public Key-Infrastruktur) aus und erfordern, dass Sie alle abhängigen Client-Betriebssystem- und Browser-Vertrauensspeicher aktualisieren müssen. Um die Auswirkungen auf die Betriebsabläufe zu minimieren, sollten Sie für das Stammzertifikat einen langen Gültigkeitszeitraum wählen. Die AWS Private CA Standardeinstellung für Stammzertifikate beträgt zehn Jahre.

Die CA-Nachfolge verwalten

Sie haben zwei Möglichkeiten, die CA-Abfolge zu verwalten: Ersetzen Sie die alte CA oder stellen Sie die CA erneut mit einem neuen Gültigkeitszeitraum aus.

Ersetzen Sie eine alte CA

Um eine alte CA zu ersetzen, erstellen Sie eine neue CA und verketten Sie sie mit derselben übergeordneten CA. Anschließend stellen Sie Zertifikate von der neuen CA aus.

Zertifikate, die von der neuen CA ausgestellt wurden, verfügen über eine neue CA-Kette. Sobald die neue CA eingerichtet ist, können Sie die alte CA deaktivieren, um zu verhindern, dass sie

neue Zertifikate ausstellt. Wenn die alte Zertifizierungsstelle deaktiviert ist, unterstützt sie den Widerruf alter Zertifikate, die von der Zertifizierungsstelle ausgestellt wurden. Falls sie entsprechend konfiguriert ist, validiert sie weiterhin Zertifikate mithilfe von and/or OCSP-Zertifikatssperrlisten (CRLs). Wenn das letzte von der alten CA ausgestellte Zertifikat abläuft, können Sie die alte CA löschen. Sie können einen Auditbericht für alle von der CA ausgestellten Zertifikate generieren, um zu bestätigen, dass alle ausgestellten Zertifikate abgelaufen sind. Wenn die alte Zertifizierungsstelle über eine untergeordnete Zertifizierungsstelle verfügt CAs, müssen Sie diese ebenfalls ersetzen, da die untergeordnete Zertifizierungsstelle zur gleichen Zeit oder vor ihrer übergeordneten Zertifizierungsstelle CAs abläuft. Ersetzen Sie zunächst die höchste CA in der Hierarchie, die ersetzt werden muss. Erstellen Sie dann auf jeder nachfolgenden untergeordneten Ebene eine neue untergeordnete CAs Ersatzinstanz.

AWS empfiehlt, dass Sie bei Bedarf eine Generierungs-ID der Zertifizierungsstelle in die Namen von CAs aufnehmen. Nehmen wir beispielsweise an, dass Sie die CA der ersten Generation „Corporate Root CA“ nennen. Wenn Sie die Zertifizierungsstelle der zweiten Generation erstellen, nennen Sie sie „Corporate Root CA G2". Diese einfache Benennungskonvention kann dazu beitragen, Verwechslungen zu vermeiden, wenn beide CAs noch nicht abgelaufen sind.

Diese Methode der CA-Abfolge wird bevorzugt, da dabei der private Schlüssel der CA rotiert. Das Rotieren des privaten Schlüssels ist eine bewährte Methode für CA-Schlüssel. Die Rotationshäufigkeit sollte proportional zur Häufigkeit der Schlüsselnutzung sein: Zertifikate CAs, die mehr ausstellen, sollten häufiger rotiert werden.

> **Note**
>
> Private Zertifikate, die über ACM ausgestellt wurden, können nicht erneuert werden, wenn Sie die CA austauschen. Wenn Sie ACM für die Ausstellung und Verlängerung verwenden, müssen Sie das CA-Zertifikat erneut ausstellen, um die Lebensdauer der Zertifizierungsstelle zu verlängern.

Eine alte CA erneut ausstellen

Wenn sich eine CA dem Ablauf nähert, besteht eine alternative Methode zur Verlängerung ihrer Lebensdauer darin, das CA-Zertifikat mit einem neuen Ablaufdatum erneut auszustellen. Bei der Neuausstellung bleiben alle CA-Metadaten erhalten und die vorhandenen privaten und öffentlichen Schlüssel werden beibehalten. In diesem Szenario bleiben die bestehende Zertifikatskette und die noch nicht abgelaufenen, von der Zertifizierungsstelle ausgestellten Endzertifikate gültig, bis sie

ablaufen. Die Ausstellung neuer Zertifikate kann auch ohne Unterbrechung fortgesetzt werden. Um eine Zertifizierungsstelle mit einem neu ausgestellten Zertifikat zu aktualisieren, folgen Sie den üblichen Installationsverfahren, die unter beschrieben sind. Installation des CA-Zertifikats

> **Note**
>
> Wir empfehlen, eine auslaufende Zertifizierungsstelle zu ersetzen, anstatt ihr Zertifikat erneut auszustellen, da die Umstellung auf ein neues key pair Sicherheitsvorteile bietet.

Widerrufen Sie eine CA

Sie widerrufen eine CA, indem Sie ihr zugrundeliegendes Zertifikat widerrufen. Dadurch werden auch effektiv alle von der CA ausgestellten Zertifikate gesperrt. Sperrinformationen werden über OCSP oder eine CRL an die Clients verteilt. Sie sollten ein CA-Zertifikat nur dann widerrufen, wenn Sie alle von der Endeinheit ausgestellten Zertifikate und untergeordneten Zertifizierungsstellenzertifikate widerrufen möchten.

Planen Sie Ihre Methode zum Widerruf von AWS Private CA Zertifikaten

Bei der Planung Ihrer privaten PKI sollten Sie überlegen AWS Private CA, wie Sie mit Situationen umgehen sollen, in denen Endgeräte einem ausgestellten Zertifikat nicht mehr vertrauen sollen, z. B. wenn der private Schlüssel eines Endpunkts offengelegt wird. Die gängigen Lösungsansätze für dieses Problem bestehen darin, kurzlebige Zertifikate zu verwenden oder den Widerruf von Zertifikaten zu konfigurieren. Kurzlebige Zertifikate laufen in einem so kurzen Zeitraum (in Stunden oder Tagen) ab, dass ein Widerruf keinen Sinn macht. Das Zertifikat wird in etwa der gleichen Zeit ungültig, die benötigt wird, um einen Endpunkt über den Widerruf zu benachrichtigen. In diesem Abschnitt werden die Widerrufsoptionen für AWS Private CA Kunden beschrieben, einschließlich Konfiguration und bewährten Methoden.

Kunden, die nach einer Sperrmethode suchen, können zwischen Online Certificate Status Protocol (OCSP), Zertifikatssperrlisten (CRLs) oder beidem wählen.

> **Note**
>
> Wenn Sie Ihre Zertifizierungsstelle erstellen, ohne den Widerruf zu konfigurieren, können Sie sie jederzeit später konfigurieren. Weitere Informationen finden Sie unter Aktualisieren Sie eine private Zertifizierungsstelle in AWS Private Certificate Authority.

- Online Certificate Status Protocol (OCSP)

  AWS Private CA bietet eine vollständig verwaltete OCSP-Lösung, mit der Endgeräte darüber informiert werden, dass Zertifikate gesperrt wurden, ohne dass Kunden die Infrastruktur selbst betreiben müssen. Kunden können OCSP für neue oder bestehende Geräte CAs mit einem einzigen Vorgang über die AWS Private CA Konsole, die API, die CLI oder über CloudFormation die Konsole aktivieren. Während CRLs OCSP-Speicher- und Verarbeitungsanforderungen auf dem Endgerät gespeichert und verarbeitet werden und veralten können, werden OCSP-Speicher- und Verarbeitungsanforderungen synchron auf dem Responder-Backend behandelt.

  Wenn Sie OCSP für eine Zertifizierungsstelle aktivieren, AWS Private CA wird die URL des OCSP-Responders in die AIA-Erweiterung (Authority Information Access) jedes neu ausgestellten Zertifikats aufgenommen. Die Erweiterung ermöglicht es Clients wie Webbrowsern, den Responder abzufragen und festzustellen, ob ein Zertifikat der Endeinheit oder einer untergeordneten Zertifizierungsstelle vertrauenswürdig ist. Der Responder gibt eine Statusmeldung zurück, die kryptografisch signiert ist, um ihre Authentizität sicherzustellen.

  Der AWS Private CA OCSP-Responder entspricht RFC 5019.

  Überlegungen zu OCSP

  - OCSP-Statusmeldungen werden mit demselben Signaturalgorithmus signiert, für den die ausstellende Zertifizierungsstelle konfiguriert wurde. CAs Die in der AWS Private CA Konsole erstellten Dateien verwenden standardmäßig den SHA256 WITRSA-Signaturalgorithmus. Andere unterstützte Algorithmen finden Sie in der CertificateAuthorityConfigurationAPI-Dokumentation.
  - APIPassthrough und CSRPassthrough Zertifikatsvorlagen funktionieren nicht mit der AIA-Erweiterung, wenn der OCSP-Responder aktiviert ist.
  - Der Endpunkt des verwalteten OCSP-Dienstes ist über das öffentliche Internet zugänglich. Kunden, die OCSP nutzen, aber keinen öffentlichen Endpunkt bevorzugen, müssen ihre eigene OCSP-Infrastruktur betreiben.
- Sperrlisten für Zertifikate () CRLs

Eine Zertifikatssperrliste (Certificate Revocation List, CRL) ist eine Datei, die eine Liste von Zertifikaten enthält, die vor ihrem geplanten Ablaufdatum gesperrt wurden. Die CRL enthält eine Liste von Zertifikaten, denen nicht mehr vertraut werden sollte, den Grund für den Widerruf und andere relevante Informationen.

Wenn Sie Ihre Zertifizierungsstelle (CA) konfigurieren, können Sie wählen, ob eine vollständige oder partitionierte CRL AWS Private CA erstellt werden soll. Ihre Wahl bestimmt die maximale Anzahl von Zertifikaten, die die Zertifizierungsstelle ausstellen und widerrufen kann. Weitere Informationen finden Sie unter AWS Private CA -Kontingente.

Überlegungen zur CRL

- Überlegungen zu Speicher und Bandbreite: Aufgrund der lokalen Download- und Verarbeitungsanforderungen ist mehr Speicher CRLs erforderlich als bei OCSP. CRLsKönnte jedoch die Netzwerkbandbreite im Vergleich zu OCSP reduzieren, indem Sperrlisten zwischengespeichert werden, anstatt den Status pro Verbindung zu überprüfen. Bei Geräten mit beschränktem Speicherplatz, wie z. B. bestimmten IoT-Geräten, sollten Sie die Verwendung partitionierter Geräte in Betracht ziehen. CRLs
- Änderung des CRL-Typs: Wenn Sie von einer vollständigen zu einer partitionierten CRL wechseln, werden bei Bedarf neue Partitionen AWS Private CA erstellt und allen Partitionen, einschließlich der ursprünglichen, die IDP-Erweiterung hinzugefügt. CRLs Der Wechsel von partitioniert zu vollständig aktualisiert nur eine einzige CRL und verhindert den future Widerruf von Zertifikaten, die mit früheren Partitionen verknüpft sind.

> **Note**
>
> Sowohl bei OCSP als auch bei der CRLs Statusänderung kommt es zu einer gewissen Verzögerung zwischen dem Widerruf und der Verfügbarkeit der Statusänderung.
>
> - Bei OCSP-Antworten kann es bis zu 60 Minuten dauern, bis der neue Status angezeigt wird, wenn Sie ein Zertifikat widerrufen. Im Allgemeinen unterstützt OCSP tendenziell eine schnellere Verteilung von Sperrinformationen, da OCSP-Antworten, im Gegensatz zu CRLs denen, die von Clients tagelang zwischengespeichert werden können, in der Regel nicht von Clients zwischengespeichert werden.

- Eine Zertifikatssperrliste wird in der Regel etwa 30 Minuten nach Widerrufen eines Zertifikats aktualisiert. Falls eine CRL-Aktualisierung aus irgendeinem Grund fehlschlägt, werden alle 15 Minuten weitere AWS Private CA Versuche unternommen.

Allgemeine Anforderungen für Sperrkonfigurationen

Die folgenden Anforderungen gelten für alle Sperrkonfigurationen.

- Eine Konfiguration zur Deaktivierung CRLs oder OCSP darf nur den `Enabled=False` Parameter enthalten und schlägt fehl, wenn andere Parameter wie `CustomCname` oder enthalten `ExpirationInDays` sind.
- In einer CRL-Konfiguration muss der `S3BucketName` Parameter den Benennungsregeln von Amazon Simple Storage Service für Buckets entsprechen.
- Eine Konfiguration, die einen benutzerdefinierten Canonical Name (CNAME) -Parameter für CRLs oder OCSP enthält, muss den Einschränkungen von RFC7230 für die Verwendung von Sonderzeichen in einem CNAME entsprechen.
- In einer CRL- oder OCSP-Konfiguration darf der Wert von CNAME kein Protokollpräfix wie „http://“ oder „https://“ enthalten.

Themen

Richten Sie eine CRL ein für AWS Private CA

Bevor Sie im Rahmen der Erstellung der Zertifizierungsstelle eine Zertifikatssperrliste (CRL) konfigurieren können, müssen Sie möglicherweise vorher einige Einstellungen vornehmen. In diesem Abschnitt werden die Voraussetzungen und Optionen erläutert, mit denen Sie vertraut sein sollten, bevor Sie eine Zertifizierungsstelle mit angehängter CRL erstellen.

Informationen zur Verwendung des Online Certificate Status Protocol (OCSP) als Alternative oder Ergänzung zu einer CRL finden Sie unter und. Certificate revocation options Passen Sie die OCSP-URL an für AWS Private CA

Themen

- CRL-Typen
- CRL-Struktur
- Zugriffsrichtlinien für CRLs in Amazon S3
- Aktivieren Sie S3 Block Public Access (BPA) mit CloudFront
- Ermitteln des CRL Distribution Point (CDP) -URI
-

CRL-Typen

- Vollständig — Die Standardeinstellung. AWS Private CA verwaltet eine einzige, unpartitionierte CRL-Datei für alle nicht abgelaufenen Zertifikate, die von einer Zertifizierungsstelle ausgestellt wurden und gesperrt wurden. Jedes ausgestellte Zertifikat ist AWS Private CA über seine CDP-Erweiterung (CRL Distribution Point) an eine bestimmte CRL gebunden, wie in RFC 5280 definiert. Sie können bis zu 1 Million private Zertifikate für jede Zertifizierungsstelle haben, wenn die vollständige CRL aktiviert ist. Weitere Informationen finden Sie in den AWS Private CA Kontingenten.
- Partitioniert — Im Vergleich zu vollständig CRLs, partitioniert erhöht CRLs sich die Anzahl der Zertifikate, die Ihre private Zertifizierungsstelle ausstellen kann, erheblich und erspart Ihnen das häufige Wechseln Ihrer Zertifikate. CAs

> **Important**
>
> Wenn Sie partitioniert verwenden CRLs, müssen Sie überprüfen, ob die der CRL zugeordnete IDP-URI (Issuing Distribution Point) mit der CDP-URI des Zertifikats übereinstimmt, um sicherzustellen, dass die richtige CRL abgerufen wurde. AWS Private CA markiert die IDP-Erweiterung als kritisch. Ihr Client muss sie verarbeiten können.

CRL-Struktur

Jede CRL ist eine DER-codierte Datei. Verwenden Sie einen Befehl, der dem folgenden ähnelt, um die Datei herunterzuladen und mit OpenSSL anzuzeigen:

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

CRLs haben das folgende Format:

```
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**
>
> Die CRL wird erst in Amazon S3 hinterlegt, nachdem ein Zertifikat ausgestellt wurde, das darauf verweist. Davor war nur eine `acm-pca-permission-test-key` Datei im Amazon S3 S3-Bucket sichtbar.

Zugriffsrichtlinien für CRLs in Amazon S3

Wenn Sie eine CRL erstellen möchten, müssen Sie einen Amazon S3 S3-Bucket vorbereiten, in dem sie gespeichert werden kann. AWS Private CA hinterlegt die CRL automatisch in dem von Ihnen Amazon S3 S3-Bucket und aktualisiert sie regelmäßig. Weitere Informationen finden Sie unter Bucket erstellen.

Ihr S3-Bucket muss durch eine beigefügte IAM-Berechtigungsrichtlinie gesichert sein. Autorisierte Benutzer und Dienstprinzipale benötigen die `Put` Erlaubnis, Objekte im Bucket platzieren AWS Private CA zu dürfen, und die `Get` Erlaubnis, sie abzurufen.

> **Note**
>
> Die Konfiguration der IAM-Richtlinie hängt von den AWS-Regionen beteiligten Personen ab. Regionen lassen sich in zwei Kategorien einteilen:
>
> - Standardmäßig aktivierte Regionen — Regionen, die standardmäßig für alle aktiviert sind. AWS-Konten
> - Standardmäßig deaktivierte Regionen — Regionen, die standardmäßig deaktiviert sind, aber vom Kunden manuell aktiviert werden können.
>
> Weitere Informationen und eine Liste der standardmäßig deaktivierten Regionen finden Sie unter Verwalten. AWS-Regionen Eine Erläuterung von Service Principals im Kontext von IAM finden Sie unter AWS Service Principals in Opt-in-Regionen.
> Bei der Konfiguration CRLs als Methode zum Widerruf von Zertifikaten AWS Private CA wird eine CRL erstellt und in einem S3-Bucket veröffentlicht. Für den S3-Bucket ist eine IAM-Richtlinie erforderlich, die es dem AWS Private CA Dienstprinzipal ermöglicht, in den Bucket zu schreiben. Der Name des Service Principal variiert je nach den verwendeten Regionen, und es werden nicht alle Möglichkeiten unterstützt.

PCA	S3	Dienstauftraggeber
Beide in derselben Region		`acm-pca.amazonaws.com`
Aktiviert	Enabled	`acm-pca.amazonaws.com`
Disabled	Aktiviert	`acm-pca.Region.amazonaws.com`
Enabled	Disabled	Nicht unterstützt

Die Standardrichtlinie gilt `SourceArn` nicht für die CA. Wir empfehlen Ihnen, eine weniger freizügige Richtlinie wie die folgende anzuwenden, die den Zugriff sowohl auf ein bestimmtes AWS Konto als auch auf eine bestimmte private Zertifizierungsstelle beschränkt. Alternativ können Sie den Bedingungsschlüssel aws: SourceOrg ID verwenden, um den Zugriff auf eine bestimmte Organisation in einzuschränken. AWS Organizations Weitere Informationen zu Bucket-Richtlinien finden Sie unter Bucket-Richtlinien für Amazon Simple Storage Service.

Wenn Sie sich dafür entscheiden, die Standardrichtlinie zuzulassen, können Sie sie später jederzeit ändern.

Aktivieren Sie S3 Block Public Access (BPA) mit CloudFront

Neue Amazon S3 S3-Buckets werden standardmäßig mit aktivierter Funktion Block Public Access (BPA) konfiguriert. BPA gehört zu den bewährten Sicherheitsmethoden von Amazon S3 und ist eine Reihe von Zugriffskontrollen, mit denen Kunden den Zugriff auf Objekte in ihren S3-Buckets und auf die Buckets insgesamt optimieren können. Wenn BPA aktiv und korrekt konfiguriert ist, haben nur autorisierte und authentifizierte AWS Benutzer Zugriff auf einen Bucket und seinen Inhalt.

AWS empfiehlt die Verwendung von BPA für alle S3-Buckets, um zu verhindern, dass vertrauliche Informationen potenziellen Gegnern zugänglich gemacht werden. Zusätzliche Planung ist jedoch erforderlich, wenn Ihre PKI-Clients Daten CRLs über das öffentliche Internet abrufen (d. h., wenn Sie nicht bei einem Konto angemeldet sind). AWS In diesem Abschnitt wird beschrieben, wie Sie eine private PKI-Lösung mithilfe von Amazon CloudFront, einem Content Delivery Network (CDN), für die Bereitstellung konfigurieren, CRLs ohne dass ein authentifizierter Client-Zugriff auf einen S3-Bucket erforderlich ist.

> **Note**
>
> Bei der Nutzung CloudFront fallen zusätzliche Kosten für Ihr Konto an. AWS Weitere Informationen finden Sie unter CloudFront Amazon-Preise.
> Wenn Sie Ihre CRL in einem S3-Bucket mit aktiviertem BPA speichern und diese nicht verwenden, müssen Sie eine weitere CDN-Lösung entwickeln CloudFront, um sicherzustellen, dass Ihr PKI-Client Zugriff auf Ihre CRL hat.

Für BPA einrichten CloudFront

Erstellen Sie eine CloudFront Distribution, die Zugriff auf Ihren privaten S3-Bucket hat und nicht authentifizierte CRLs Clients bedienen kann.

Um eine CloudFront Distribution für die CRL zu konfigurieren

1. Erstellen Sie eine neue CloudFront Distribution, indem Sie das Verfahren unter Creating a Distribution im Amazon CloudFront Developer Guide verwenden.

   Wenden Sie beim Abschluss des Verfahrens die folgenden Einstellungen an:

   - Wählen Sie unter Origin Domain Name Ihren S3-Bucket aus.
   - Wählen Sie Ja für „Bucket-Zugriff einschränken“.
   - Wähle „Neue Identität erstellen“ für Origin Access Identity aus.
   - Wähle Ja, Bucket-Richtlinie aktualisieren unter Leseberechtigungen für Bucket gewähren aus.

   > **Note**
   >
   > In diesem Verfahren wird Ihre Bucket-Richtlinie so CloudFront geändert, dass sie auf Bucket-Objekte zugreifen kann. Erwägen Sie, diese Richtlinie so zu bearbeiten, dass nur auf Objekte im `crl` Ordner zugegriffen werden kann.

2. Suchen Sie nach der Initialisierung der Distribution ihren Domänennamen in der CloudFront Konsole und speichern Sie ihn für den nächsten Vorgang.

   > **Note**
   >
   > Wenn Ihr S3-Bucket in einer anderen Region als us-east-1 neu erstellt wurde, erhalten Sie möglicherweise einen temporären HTTP 307-Umleitungsfehler, wenn Sie über auf

Ihre veröffentlichte Anwendung zugreifen. CloudFront Es kann mehrere Stunden dauern, bis die Adresse des Buckets weitergegeben wird.

Richten Sie Ihre CA für BPA ein

Fügen Sie bei der Konfiguration Ihrer neuen CA den Alias zu Ihrer CloudFront Distribution hinzu.

Um Ihre CA mit einem CNAME zu konfigurieren für CloudFront

- Erstellen Sie Ihre CA mitErstellen Sie eine private CA in AWS Private CA.

  Wenn Sie das Verfahren ausführen, `revoke_config.txt` sollte die Sperrdatei die folgenden Zeilen enthalten, um ein nichtöffentliches CRL-Objekt anzugeben und eine URL zum Verteilungsendpunkt in bereitzustellen: CloudFront

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

  Wenn Sie anschließend Zertifikate mit dieser Zertifizierungsstelle ausstellen, enthalten sie einen Block wie den folgenden:

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

> **Note**
>
> Wenn Sie über ältere Zertifikate verfügen, die von dieser Zertifizierungsstelle ausgestellt wurden, können diese nicht auf die CRL zugreifen.

Ermitteln des CRL Distribution Point (CDP) -URI

Wenn Sie den CRL Distribution Point (CDP) -URI in Ihrem Workflow verwenden müssen, können Sie entweder ein Zertifikat ausstellen, indem Sie den CRL-URI auf diesem Zertifikat verwenden oder die

folgende Methode verwenden. Dies funktioniert nur, wenn der Vorgang abgeschlossen ist. CRLs An Partitionen wird CRLs eine zufällige GUID angehängt.

Wenn Sie den S3-Bucket als CRL Distribution Point (CDP) für Ihre CA verwenden, kann der CDP-URI eines der folgenden Formate haben.

- `http://amzn-s3-demo-bucket.s3.region-code.amazonaws.com/crl/CA-ID.crl`
- `http://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/crl/CA-ID.crl`

Wenn Sie Ihre CA mit einem benutzerdefinierten CNAME konfiguriert haben, enthält der CDP-URI den CNAME, zum Beispiel `http://alternative.example.com/crl/CA-ID.crl`

AWS Private CA Schreibt CDP-Erweiterungen standardmäßig mit regionalen Endpunkten, die nur verfügbar sind. IPv4 `amazonaws.com` Um CRLs over zu verwenden IPv6, führen Sie einen der folgenden Schritte aus, sodass dieser Punkt auf CDPs die URLs Dual-Stack-Endpunkte von S3 geschrieben wird:

- Legen Sie Ihren benutzerdefinierten CRL-Namen auf die S3-Dualstack-Endpunktdomäne fest. Beispiel: `bucketname.s3.dualstack.region-code.amazonaws.com`
- Richten Sie Ihren eigenen CNAME-DNS-Eintrag ein, der auf den entsprechenden S3-Dualstack-Endpunkt verweist, und verwenden Sie ihn dann als Ihren benutzerdefinierten CRL-Namen

Passen Sie die OCSP-URL an für AWS Private CA

> **Note**
>
> Dieses Thema richtet sich an Kunden, die die öffentliche URL des OCSP-Responder-Endpunkts (Online Certificate Status Protocol) für Branding- oder andere Zwecke anpassen möchten. Wenn Sie die Standardkonfiguration von AWS Private CA verwaltetem OCSP verwenden möchten, können Sie dieses Thema überspringen und den Konfigurationsanweisungen unter Sperre konfigurieren folgen.

Wenn Sie OCSP für aktivieren, enthält jedes Zertifikat AWS Private CA, das Sie ausstellen, standardmäßig die URL für den AWS OCSP-Responder. Auf diese Weise können Clients, die eine kryptografisch sichere Verbindung anfordern, OCSP-Validierungsanfragen direkt an diese senden.

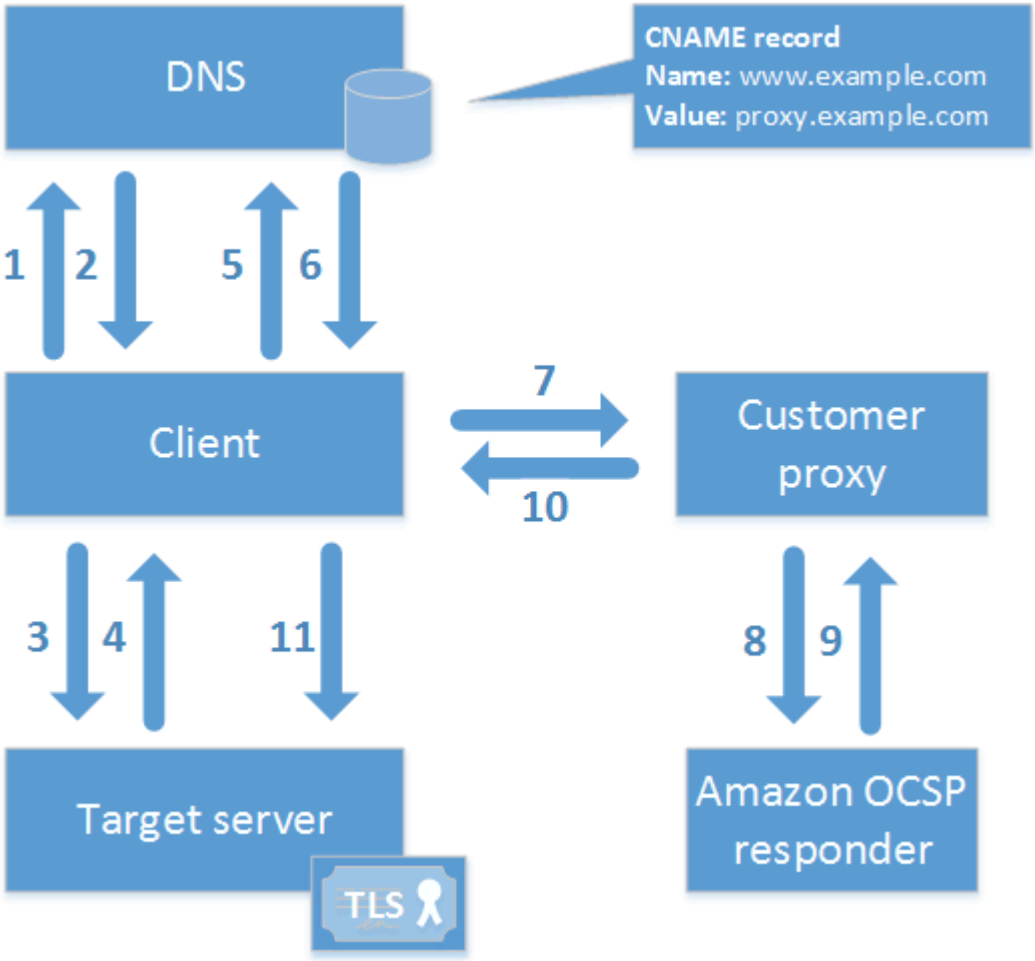
AWS In einigen Fällen kann es jedoch vorzuziehen sein, in Ihren Zertifikaten eine andere URL anzugeben, während Sie letztendlich OCSP-Abfragen an senden. AWS

> **Note**
>
> Informationen zur Verwendung einer Zertifikatssperrliste (CRL) als Alternative oder Ergänzung zu OCSP finden Sie unter Sperrung konfigurieren und Planung einer Zertifikatssperrliste (CRL).

Bei der Konfiguration einer benutzerdefinierten URL für OCSP sind drei Elemente erforderlich.

- CA-Konfiguration — Geben Sie in der `RevocationConfiguration` für Ihre CA eine benutzerdefinierte OCSP-URL an, wie unter beschriebenBeispiel 2: Erstellen Sie eine CA mit aktiviertem OCSP und einem benutzerdefinierten CNAME. Erstellen Sie eine private CA in AWS Private CA
- DNS — Fügen Sie Ihrer Domain-Konfiguration einen CNAME-Eintrag hinzu, um die in den Zertifikaten angezeigte URL einer Proxyserver-URL zuzuordnen. Weitere Informationen finden Sie unter Beispiel 2: Erstellen Sie eine CA mit aktiviertem OCSP und einem benutzerdefinierten CNAME in Erstellen Sie eine private CA in AWS Private CA.
- Proxyserver weiterleiten — Richten Sie einen Proxyserver ein, der den empfangenen OCSP-Verkehr transparent an den OCSP-Responder AWS weiterleiten kann.

Das folgende Diagramm zeigt, wie diese Elemente zusammenarbeiten.



Wie im Diagramm dargestellt, umfasst der benutzerdefinierte OCSP-Validierungsprozess die folgenden Schritte:

1. Der Client fragt DNS für die Zieldomäne ab.
2. Der Client empfängt die Ziel-IP.
3. Der Client öffnet eine TCP-Verbindung mit dem Ziel.
4. Der Client erhält das Ziel-TLS-Zertifikat.
5. Der Client fragt DNS für die im Zertifikat aufgeführte OCSP-Domäne ab.
6. Der Client erhält eine Proxy-IP.
7. Der Client sendet eine OCSP-Anfrage an den Proxy.
8. Der Proxy leitet die Anfrage an den OCSP-Responder weiter.
9. Der Responder gibt den Zertifikatsstatus an den Proxy zurück.
10. Der Proxy leitet den Zertifikatsstatus an den Client weiter.

11. Wenn das Zertifikat gültig ist, beginnt der Client mit dem TLS-Handshake.

> **Tip**
>
> Dieses Beispiel kann mit Amazon CloudFront und Amazon Route 53 implementiert werden, nachdem Sie eine CA wie oben beschrieben konfiguriert haben.
>
> 1. Erstellen Sie in CloudFront eine Distribution und konfigurieren Sie sie wie folgt:
>   - Erstellen Sie einen alternativen Namen, der Ihrem benutzerdefinierten CNAME entspricht.
>   - Binden Sie Ihr Zertifikat daran.
>   - `ocsp.acm-pca.<region>.amazonaws.com`Als Ursprung festlegen.
>   - Verwenden Sie den Dualstack-Endpunkt, um IPv6 Verbindungen zu verwenden `acm-pca-ocsp.<region>.api.aws`
>   - Wenden Sie die Richtlinie an`Managed-CachingDisabled`.
>   - Stellen Sie die Viewer-Protokollrichtlinie auf HTTP und HTTPS ein.
>   - Stellen Sie die zulässigen HTTP-Methoden auf GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE ein.
> 2. Erstellen Sie in Route 53 einen DNS-Eintrag, der Ihren benutzerdefinierten CNAME der URL der CloudFront Distribution zuordnet.

Verwenden Sie OCSP über IPv6

Die AWS Private CA Standard-OCSP-Responder-URL ist -only. IPv4 Um OCSP over zu verwenden IPv6, konfigurieren Sie eine benutzerdefinierte OCSP-URL für Ihre CA. Die URL kann entweder sein:

- Der FQDN des Dual-Stack-PCA-OCSP-Responders, der das folgende Format hat `acm-pca-ocsp.region-name.api.aws`
- Ein CNAME-Eintrag, den Sie so konfiguriert haben, dass er auf den Dual-Stack-OCSP-Responder verweist, wie oben beschrieben.

AWS Private CA Verstehen Sie die CA-Modi

AWS Private CA unterstützt die Erstellung einer Zertifizierungsstelle (CA) in einem von zwei Modi. Die Modi „Allzweck-Zertifikat“ und „Kurzlebiges Zertifikat“ wirken sich auf die zulässige Gültigkeitsdauer der von der Zertifizierungsstelle ausgestellten Zertifikate aus.

> **Note**
>
> AWS Private CA führt keine Gültigkeitsprüfungen für Root-CA-Zertifikate durch.

Universell einsetzbar (Standard)

In diesem Modus kann die CA Zertifikate mit beliebiger Gültigkeitsdauer ausstellen. Die meisten Anwendungen verwenden Zertifikate dieses Typs. In der Regel spezifiziert die CA auch einen Sperrmechanismus.

Kurzlebiges Zertifikat

In diesem Modus wird eine Zertifizierungsstelle definiert, die ausschließlich Zertifikate mit einer maximalen Gültigkeitsdauer von sieben Tagen ausstellt. Diese kurzlebigen Zertifikate laufen so schnell ab, dass sie ohne einen Sperrmechanismus bereitgestellt werden können. Für einige Anwendungen ist es sinnvoller, häufig kurzlebige Zertifikate bereitzustellen, als den Netzwerk- und Verarbeitungsaufwand durch den Widerruf auf sich zu nehmen.

Kurzlebige Zertifikate müssen die letzte Zertifizierungsstelle in der Zertifikatshierarchie sein. Es entsteht ein erheblicher Mehraufwand, da die private CA alle sieben Tage erneuert werden muss.

CAs Der kurzlebige Zertifikatsmodus kostet weniger als der allgemeine CAs Zertifikatsmodus. Weitere Informationen finden Sie unter AWS Private Certificate Authority – Preise.

Um eine Zertifizierungsstelle zu erstellen, die kurzlebige Zertifikate ausstellt, setzen Sie den `UsageMode` Parameter mithilfe der Prozedur zum Erstellen einer Zertifizierungsstelle auf kurzlebiges Zertifikat.

> **Note**
>
> AWS Certificate Manager kann keine Zertifikate ausstellen, die von einer privaten Zertifizierungsstelle im kurzlebigen Modus signiert wurden.

Die Verwendung von kurzlebigen Zertifikaten wird von den folgenden AWS Diensten unterstützt:

- Amazon AppStream
- Amazon WorkSpaces

Planen Sie für Resilienz in AWS Private CA

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Redundanz und Disaster Recovery

Berücksichtigen Sie Redundanz und DR bei der Planung Ihrer CA-Hierarchie. AWS Private CA ist in mehreren Regionen verfügbar, sodass Sie Redundanzen CAs in mehreren Regionen einrichten können. Für den AWS Private CA Service gilt ein Service Level Agreement (SLA) mit einer Verfügbarkeit von 99,9%. Es gibt mindestens zwei Ansätze, die Sie für Redundanz und Notfallwiederherstellung in Betracht ziehen können. Sie können Redundanz an der Stamm-CA oder an der höchsten untergeordneten CA konfigurieren. Jeder Ansatz hat Vor- und Nachteile.

1. Aus Redundanz- und Notfallwiederherstellungsgründen können Sie zwei Roots CAs in zwei verschiedenen AWS Regionen einrichten. Bei dieser Konfiguration arbeitet jede Stammzertifizierungsstelle unabhängig in einer AWS Region, sodass Sie im Falle eines Notfalls in einer Region geschützt sind. Das Erstellen redundanter Root-Zertifikate erhöht CAs jedoch die betriebliche Komplexität: Sie müssen beide Root-CA-Zertifikate an die Vertrauensspeicher der Browser und Betriebssysteme in Ihrer Umgebung verteilen.
2. Sie können auch redundante untergeordnete Zertifikate CAs einrichten, die in jeder Ihrer AWS Regionen bereitgestellt werden, und diese an dieselbe eindeutige Stammzertifizierungsstelle in einer einzigen AWS Region anhängen. Der Vorteil dieses Ansatzes besteht darin, dass Sie nur ein einzelnes Stamm-CA-Zertifikat an die Vertrauensspeicher in Ihrer Umgebung verteilen müssen.

Die Einschränkung besteht darin, dass Sie für den Fall einer Katastrophe, die sich auf die AWS Region auswirkt, in der sich Ihre Stammzertifizierungsstelle befindet, nicht über eine redundante Stammzertifizierungsstelle verfügen.

Zertifizierungsstellen in AWS Private CA

Mit AWS Private Certificate Authority ihr können Sie eine vollständig AWS gehostete Hierarchie von Stamm- und untergeordneten Zertifizierungsstellen (CAs) für den internen Gebrauch in Ihrer Organisation erstellen. Um den Widerruf von Zertifikaten zu verwalten, können Sie das Online Certificate Status Protocol (OCSP), Zertifikatssperrlisten (CRLs) oder beides aktivieren. AWS Private CA speichert und verwaltet Ihre CA-Zertifikate und OCSP-Antworten CRLs, und die privaten Schlüssel für Ihre Root-Autoritäten werden sicher von gespeichert. AWS

> **Note**
>
> Die OCSP-Implementierung in unterstützt AWS Private CA keine OCSP-Anforderungserweiterungen. Wenn Sie eine OCSP-Batchanfrage mit mehreren Zertifikaten einreichen, verarbeitet der AWS OCSP-Responder nur das erste Zertifikat in der Warteschlange und löscht die anderen. Es kann bis zu einer Stunde dauern, bis ein Widerruf in den OCSP-Antworten erscheint.

Sie können AWS Private CA über die AWS-Managementkonsole AWS CLI, und die AWS Private CA API darauf zugreifen. In den folgenden Themen sehen Sie, wie Sie die Konsole und die CLI verwenden. Weitere Informationen zur API finden Sie in der AWS Private Certificate Authority API-Referenz. Java-Beispiele, die Ihnen verdeutlichen, wie Sie die API verwenden, finden Sie unter Verwenden Sie AWS Private CA mit dem AWS SDK for Java.

Nachdem Sie eine aktive private Zertifizierungsstelle erstellt und den Zugriff darauf konfiguriert haben, können Sie Zertifikate ausstellen und abrufen, wie unter beschriebenZertifikate ausstellen und verwalten in AWS Private CA.

Themen

- Fügen Sie Tags für Ihre private CA hinzu
- Verstehen Sie den AWS Private CA CA-Status
- Aktualisieren Sie eine private Zertifizierungsstelle in AWS Private Certificate Authority
- Löschen Ihrer privaten CA
- Stellen Sie eine private CA wieder her
- Verwenden Sie extern signierte private CA-Zertifikate

Zur Verwendung eingerichtet AWS Private CA

Wenn Sie noch kein Kunde von Amazon Web Services (AWS) sind, müssen Sie sich registrieren, um Amazon Web Services () nutzen zu können AWS Private CA. Ihr Konto hat automatisch Zugriff auf alle verfügbaren Services, aber es werden Ihnen nur Services, die Sie nutzen, in Rechnung gestellt.

Topics

- Melden Sie sich an für ein AWS-Konto
- Erstellen eines Benutzers mit Administratorzugriff
- Installieren Sie das AWS Command Line Interface

Melden Sie sich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Benutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Du kannst jederzeit deine aktuellen Kontoaktivitäten einsehen und dein Konto verwalten, indem du zu https://aws.amazon.com/gehst und Mein Konto auswählst.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS-Managementkonsoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung -Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center -Benutzerhandbuch.
2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie IAM-Identity-Center-Verzeichnis im Benutzerhandbuch unter Benutzerzugriff mit der Standardeinstellung konfigurieren.AWS IAM Identity Center

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Access-Portal.

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

   Anweisungen hierzu finden Sie unter Berechtigungssatz erstellen im AWS IAM Identity Center - Benutzerhandbuch.
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Eine genaue Anleitung finden Sie unter Gruppen hinzufügen im AWS IAM Identity Center - Benutzerhandbuch.

Installieren Sie das AWS Command Line Interface

Wenn Sie das nicht installiert haben, es AWS CLI aber verwenden möchten, folgen Sie den Anweisungen unter AWS Command Line Interface. In diesem Handbuch gehen wir davon aus, dass Sie Ihren Endpunkt, Ihre Region und Ihre Authentifizierungsdetails konfiguriert haben, und wir lassen diese Parameter in den Beispielbefehlen weg.

Erstellen Sie eine private CA in AWS Private CA

Sie können die Verfahren in diesem Abschnitt verwenden, um entweder eine Stamm CAs - oder eine untergeordnete CAs Struktur zu erstellen. Das Ergebnis ist eine überprüfbare Hierarchie von Vertrauensbeziehungen, die Ihren organisatorischen Anforderungen entspricht. Sie können eine Zertifizierungsstelle erstellen AWS-Managementkonsole, indem Sie den PCA-Teil von oder verwenden. AWS CLI AWS CloudFormation

Informationen zum Aktualisieren der Konfiguration einer Zertifizierungsstelle, die Sie bereits erstellt haben, finden Sie unterAktualisieren Sie eine private Zertifizierungsstelle in AWS Private Certificate Authority.

Informationen zur Verwendung einer Zertifizierungsstelle zum Signieren von Endentitätszertifikaten für Ihre Benutzer, Geräte und Anwendungen finden Sie unterStellen Sie private Endentitätszertifikate aus.

> **Note**
>
> Ihrem Konto wird ab dem Zeitpunkt, zu dem Sie sie erstellen, ein monatlicher Preis für jede private CA in Rechnung gestellt.
> Die neuesten AWS Private CA Preisinformationen finden Sie unter AWS Private Certificate Authority Preise. Sie können auch den AWS Preisrechner verwenden, um die Kosten zu schätzen.

Themen

- CLI-Beispiele für die Erstellung einer privaten CA

Console

So erstellen Sie eine private CA über die -Konsole

1. Gehen Sie wie folgt vor, um eine private Zertifizierungsstelle mit dem zu erstellen AWS-Managementkonsole.

   Um mit der Verwendung der Konsole zu beginnen

   Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Konsole unter`https://console.aws.amazon.com/acm-pca/home`.

   - Wenn du die Konsole in einer Region öffnest, in der du kein Privatkonto hast CAs, wird die Einführungsseite angezeigt. Wählen Sie Create a Private CA aus.
   - Wenn Sie die Konsole in einer Region öffnen, in der Sie bereits eine Zertifizierungsstelle erstellt haben, wird die Seite Private Zertifizierungsstellen mit einer Liste Ihrer Zertifizierungsstellen geöffnet CAs. Wählen Sie Create CA aus.

2.

Wählen Sie unter Modusoptionen den Ablaufmodus der Zertifikate aus, die Ihre CA ausstellt.

- Allgemein — Stellt Zertifikate aus, die mit einem beliebigen Ablaufdatum konfiguriert werden können. Dies ist die Standardeinstellung.
- Kurzlebiges Zertifikat — Stellt Zertifikate mit einer maximalen Gültigkeitsdauer von sieben Tagen aus. Eine kurze Gültigkeitsdauer kann in einigen Fällen einen Sperrmechanismus ersetzen.

3. Wählen Sie im Abschnitt Typoptionen der Konsole den Typ der privaten Zertifizierungsstelle aus, die Sie erstellen möchten.

- Wenn Sie Root wählen, wird eine neue CA-Hierarchie eingerichtet. Diese CA wird durch ein selbstsigniertes Zertifikat gesichert. Sie dient als ultimative Signaturautorität für andere Zertifikate CAs und Endentitätszertifikate in der Hierarchie.
- Wenn Sie Untergeordnet wählen, wird eine Zertifizierungsstelle erstellt, die von einer übergeordneten Zertifizierungsstelle signiert werden muss, die ihr in der Hierarchie übergeordnet ist. Untergeordnete Entitäten CAs werden in der Regel verwendet, um weitere untergeordnete Entitätszertifikate zu erstellen CAs oder Endentitätszertifikate für Benutzer, Computer und Anwendungen auszustellen.

> **Note**
>
> AWS Private CA bietet einen automatisierten Signaturprozess, wenn die übergeordnete Zertifizierungsstelle Ihrer untergeordneten Zertifizierungsstelle ebenfalls von gehostet wird. AWS Private CA Sie müssen lediglich die zu verwendende übergeordnete Zertifizierungsstelle auswählen.
> Ihre untergeordnete Zertifizierungsstelle muss möglicherweise von einem externen Vertrauensdienstanbieter signiert werden. Falls ja, AWS Private CA stellt er Ihnen eine Certificate Signing Request (CSR) zur Verfügung, die Sie herunterladen und verwenden müssen, um ein signiertes CA-Zertifikat zu erhalten. Weitere Informationen finden Sie unter Installieren Sie ein Zertifikat einer untergeordneten Zertifizierungsstelle, das von einer externen übergeordneten Zertifizierungsstelle signiert wurde.

4.

Konfigurieren Sie unter Optionen für den definierten Betreffnamen den Betreffnamen Ihrer privaten Zertifizierungsstelle. Sie müssen einen Wert für mindestens eine der folgenden Optionen eingeben:

- Organisation (O) — Zum Beispiel ein Firmenname
- Organisationseinheit (OU) — Zum Beispiel eine Abteilung innerhalb eines Unternehmens
- Ländername (C) — Ein aus zwei Buchstaben bestehender Ländercode
- Name des Bundesstaates oder der Provinz — Vollständiger Name eines Bundesstaates oder einer Provinz
- Ortsname — Der Name einer Stadt
- Allgemeiner Name (CN) — Eine für Menschen lesbare Zeichenfolge zur Identifizierung der CA.

> **Note**
>
> Sie können den Betreffnamen eines Zertifikats weiter anpassen, indem Sie bei der Ausstellung eine APIPassthrough Vorlage verwenden. Weitere Informationen und ein ausführliches Beispiel finden Sie unterStellen Sie mithilfe einer APIPassthrough Vorlage ein Zertifikat mit einem benutzerdefinierten Betreffnamen aus.

Da das Hintergrundzertifikat selbst signiert ist, sind die Betreffinformationen, die Sie für eine private Zertifizierungsstelle angeben, wahrscheinlich spärlicher als die Informationen, die eine öffentliche Zertifizierungsstelle enthalten würde. Weitere Informationen zu den einzelnen Werten, aus denen sich ein definierter Name für den Betreffenden zusammensetzt, finden Sie in RFC 5280.

5. Wählen Sie unter Optionen für den Schlüsselalgorithmus den Schlüsselalgorithmus und die Stärke des Algorithmus aus. Der Standardwert ist RSA 2048. Sie können aus den folgenden Algorithmen wählen:

   - ML-DSA-44
   - ML-DSA-65
   - ML-DSA-87
   - RSA 2048

- RSA 3072
- RSA 4096
- ECDSA P256
- ECDSA P384
- ECDSA P512

6. Unter Zertifikatssperroptionen können Sie zwischen zwei Methoden wählen, um den Sperrstatus mit Clients zu teilen, die Ihre Zertifikate verwenden:

   - Aktivieren Sie die CRL-Verteilung
   - Schalten Sie OCSP ein

   Sie können eine, keine oder beide dieser Sperroptionen für Ihre CA konfigurieren. Der verwaltete Widerruf ist zwar optional, wird aber als bewährte Methode empfohlen. Bevor Sie diesen Schritt abschließen, finden Sie unter Planen Sie Ihre Methode zum Widerruf von AWS Private CA Zertifikaten Informationen zu den Vorteilen der einzelnen Methoden, zur eventuell erforderlichen vorläufigen Einrichtung und zu zusätzlichen Sperrfunktionen.

   > **Note**
   >
   > Wenn Sie Ihre Zertifizierungsstelle erstellen, ohne den Widerruf zu konfigurieren, können Sie sie jederzeit später konfigurieren. Weitere Informationen finden Sie unter Aktualisieren Sie eine private Zertifizierungsstelle in AWS Private Certificate Authority.

   Gehen Sie wie folgt vor, um die Optionen für den Widerruf von Zertifikaten zu konfigurieren.

   a. Wählen Sie unter Zertifikatssperroptionen die Option CRL-Verteilung aktivieren aus.

   b. Wählen Sie unter S3-Bucket-URI einen vorhandenen Bucket aus der Liste aus.

      Wenn Sie einen vorhandenen Bucket angeben, müssen Sie sicherstellen, dass BPA für das Konto und den Bucket deaktiviert ist. Andernfalls schlägt der Vorgang zum Erstellen der CA fehl. Wenn die Zertifizierungsstelle erfolgreich erstellt wurde, müssen Sie ihr dennoch manuell eine Richtlinie hinzufügen, bevor Sie mit der Generierung beginnen können CRLs. Verwenden Sie eines der unter beschriebenen

RichtlinienmusterZugriffsrichtlinien für CRLs in Amazon S3. Weitere Informationen finden Sie unter Hinzufügen einer Bucket-Richtlinie mithilfe der Amazon S3 S3-Konsole.

c. Erweitern Sie die CRL-Einstellungen für zusätzliche Konfigurationsoptionen.

- Wählen Sie Partitionierung aktivieren, um die Partitionierung von zu aktivieren. CRLs Wenn Sie die Partitionierung nicht aktivieren, gilt für Ihre CA die maximale Anzahl von gesperrten Zertifikaten. Weitere Informationen finden Sie unter AWS Private Certificate Authority -Kontingente. Weitere Informationen zu partitionierten Dateien finden Sie unter CRLs CRL-Typen.
- Fügen Sie einen benutzerdefinierten CRL-Namen hinzu, um einen Alias für Ihren Amazon S3 S3-Bucket zu erstellen. Dieser Name ist in Zertifikaten enthalten, die von der Zertifizierungsstelle in der Erweiterung „CRL Distribution Points“ ausgestellt wurden, die in RFC 5280 definiert ist. Um CRLs over zu verwenden IPv6, setzen Sie dies auf den DualStack-S3-Endpunkt Ihres Buckets, wie unter Using over beschrieben. CRLs IPv6
- Fügen Sie einen benutzerdefinierten Pfad hinzu, um einen DNS-Alias für den Dateipfad in Ihrem Amazon S3 S3-Bucket zu erstellen.
- Geben Sie die Gültigkeitsdauer in Tagen ein, in der Ihre CRL gültig bleiben soll. Der Standardwert lautet 7 Tage. Im Online-Bereich CRLs ist eine Gültigkeitsdauer von 2-7 Tagen üblich. AWS Private CA versucht, die CRL in der Mitte des angegebenen Zeitraums zu regenerieren.

7. Wählen Sie für Optionen zum Widerruf von Zertifikaten die Option OCSP einschalten aus.

- Im Feld Benutzerdefinierter OCSP-Endpunkt — optional können Sie einen vollqualifizierten Domainnamen (FQDN) für einen Nicht-Amazon-OCSP-Endpunkt angeben. Um OCSP over zu verwenden IPv6, setzen Sie dieses Feld auf einen Dual-Stack-Endpunkt, wie unter OCSP Over verwenden beschrieben. IPv6

  Wenn Sie in diesem Feld einen FQDN angeben, wird der FQDN anstelle der Standard-URL für den OCSP-Responder in die Authority Information Access-Erweiterung jedes ausgestellten Zertifikats AWS Private CA eingefügt. AWS Wenn ein Endpunkt ein Zertifikat empfängt, das den benutzerdefinierten FQDN enthält, fragt er diese Adresse nach einer OCSP-Antwort ab. Damit dieser Mechanismus funktioniert, müssen Sie zwei zusätzliche Maßnahmen ergreifen:

  - Verwenden Sie einen Proxyserver, um den Datenverkehr, der bei Ihrem benutzerdefinierten FQDN eingeht, an den AWS OCSP-Responder weiterzuleiten.

- Fügen Sie Ihrer DNS-Datenbank einen entsprechenden CNAME-Eintrag hinzu.

> **Tip**
>
> Weitere Hinweise zur Implementierung einer vollständigen OCSP-Lösung mit einem benutzerdefinierten CNAME finden Sie unter. Passen Sie die OCSP-URL an für AWS Private CA

Hier ist zum Beispiel ein CNAME-Eintrag für benutzerdefiniertes OCSP, wie er in Amazon Route 53 erscheinen würde.

Datensatzname	Typ	Routing-Richtlinie	Unterscheidungsmerkmal	Bewerten/Weiterleiten des Datenverkehrs an
alternative.example.com	CNAME	Einfach	-	proxy.example.com

> **Note**
>
> Der Wert des CNAME-Werts darf kein Protokollpräfix wie „http://“ oder „https://“ enthalten.

8. Unter Tags hinzufügen können Sie optional Ihre CA taggen. Tags sind Schlüssel-Wert-Paare, die als Metadaten zum Identifizieren und Organisieren von AWS -Ressourcen dienen. Eine Liste der AWS Private CA Tag-Parameter und Anweisungen zum Hinzufügen von Tags CAs nach der Erstellung finden Sie unterFügen Sie Tags für Ihre private CA hinzu.

> **Note**
>
> Um während des Erstellungsvorgangs Tags an eine private CA anzuhängen, muss ein CA-Administrator der `CreateCertificateAuthority` Aktion zunächst

eine Inline-IAM-Richtlinie zuordnen und das Tagging explizit zulassen. Weitere Informationen finden Sie unter Tag-on-create: Hinzufügen von Tags an eine CA zum Zeitpunkt der Erstellung.

9. Unter den CA-Berechtigungsoptionen können Sie optional automatische Verlängerungsberechtigungen an den AWS Certificate Manager Service Principal delegieren. ACM kann private Endentitätszertifikate, die von dieser CA generiert wurden, nur automatisch erneuern, wenn diese Berechtigung erteilt wird. Sie können jederzeit Verlängerungsberechtigungen mit dem AWS Private CA CreatePermissionAPI- oder create-permission-CLI-Befehl zuweisen.

   Standardmäßig werden diese Berechtigungen aktiviert.

   > **Note**
   >
   > AWS Certificate Manager unterstützt die automatische Verlängerung von kurzlebigen Zertifikaten nicht.

10. Bestätigen Sie unter Preisgestaltung, dass Sie die Preisgestaltung für eine private Zertifizierungsstelle verstanden haben.

    > **Note**
    >
    > Die neuesten AWS Private CA Preisinformationen finden Sie unter AWS Private Certificate Authority Preisgestaltung. Sie können auch den AWS Preisrechner verwenden, um die Kosten zu schätzen.

11. Wählen Sie Create CA, nachdem Sie alle eingegebenen Informationen auf Richtigkeit überprüft haben. Die Detailseite für die Zertifizierungsstelle wird geöffnet und ihr Status wird als Ausstehendes Zertifikat angezeigt.

    > **Note**
    >
    > Wenn Sie sich auf der Detailseite befinden, können Sie die Konfiguration Ihrer Zertifizierungsstelle abschließen, indem Sie Aktionen, CA-Zertifikat installieren

wählen. Sie können auch später zur Liste der privaten Zertifizierungsstellen zurückkehren und den Installationsvorgang abschließen, der für Ihren Fall gilt:

- Installieren Sie ein Root-CA-Zertifikat
- Installieren Sie ein untergeordnetes CA-Zertifikat, das von gehostet wird AWS Private CA
- Installieren Sie ein Zertifikat einer untergeordneten Zertifizierungsstelle, das von einer externen übergeordneten Zertifizierungsstelle signiert wurde

CLI

Verwenden Sie den Befehl create-certificate-authority, um eine private CA zu erstellen. Sie müssen die CA-Konfiguration (mit Informationen zum Algorithmus und zum Betreffnamen), die Sperrkonfiguration (wenn Sie OCSP and/or als CRL verwenden möchten) und den Typ der Zertifizierungsstelle (Stamm oder untergeordnet) angeben. Die Details der Konfiguration und der Sperrkonfiguration sind in zwei Dateien enthalten, die Sie als Argumente für den Befehl angeben. Optional können Sie auch den CA-Nutzungsmodus (für die Ausstellung von Standard- oder kurzlebigen Zertifikaten) konfigurieren, Tags anhängen und ein Idempotenz-Token bereitstellen.

Wenn Sie eine CRL konfigurieren, müssen Sie über einen gesicherten Amazon S3 S3-Bucket verfügen, bevor Sie den create-certificate-authority Befehl ausgeben. Weitere Informationen finden Sie unter Zugriffsrichtlinien für CRLs in Amazon S3.

Die CA-Konfigurationsdatei spezifiziert die folgenden Informationen:

- Den Namen des Algorithmus
- Die Schlüsselgröße, die beim Erstellen eines privaten Schlüssels für die CA verwendet werden soll
- Die Art des Signaturalgorithmus, den die CA verwendet, um ihre eigenen Certificate Signing Request- CRLs und OCSP-Antworten zu signieren
- X.500-Themeninformationen

Die Sperrkonfiguration für OCSP definiert ein `OcspConfiguration` Objekt mit den folgenden Informationen:

- Das `Enabled` Flag ist auf „true“ gesetzt.

- (Optional) Ein benutzerdefinierter CNAME, der als Wert für `OcspCustomCname` deklariert wurde.

Die Sperrkonfiguration für eine CRL definiert ein `CrlConfiguration` Objekt mit den folgenden Informationen:

- Das auf „true“ gesetzte `Enabled` Flag.
- Der CRL-Ablaufzeitraum in Tagen (der Gültigkeitszeitraum der CRL).
- Der Amazon S3 S3-Bucket, der die CRL enthalten wird.
- (Optional) Ein ObjectAclS3-Wert, der bestimmt, ob die CRL öffentlich zugänglich ist. In dem hier vorgestellten Beispiel ist der öffentliche Zugriff gesperrt. Weitere Informationen finden Sie unter Aktivieren Sie S3 Block Public Access (BPA) mit CloudFront.
- (Optional) Ein CNAME-Alias für den S3-Bucket, der in den von der CA ausgestellten Zertifikaten enthalten ist. Wenn die CRL nicht öffentlich zugänglich ist, deutet dies auf einen Vertriebsmechanismus wie Amazon CloudFront hin.
- (Optional) Ein `CrlDistributionPointExtensionConfiguration` Objekt mit den folgenden Informationen:
  - Die `OmitExtension` Markierung ist auf „wahr“ oder „falsch“ gesetzt. Dies steuert, ob der Standardwert für die CDP-Erweiterung in ein von der CA ausgestelltes Zertifikat geschrieben wird. Weitere Hinweise zur CDP-Erweiterung finden Sie unter. Ermitteln des CRL Distribution Point (CDP) -URI A CustomCname kann nicht gesetzt werden, wenn OmitExtension es „wahr“ ist.
- (Optional) Ein benutzerdefinierter Pfad für die CRL im S3-Bucket.
- (Optional) Ein CrlTypeWert, der bestimmt, ob die CRL vollständig oder partitioniert sein wird. Wenn nicht angegeben, wird die CRL standardmäßig auf abgeschlossen gesetzt.

> **Note**
>
> Sie können beide Sperrmechanismen auf derselben Zertifizierungsstelle aktivieren, indem Sie sowohl ein `OcspConfiguration` Objekt als auch ein `CrlConfiguration` Objekt definieren. Wenn Sie keinen --revocation-configuration Parameter angeben, sind beide Mechanismen standardmäßig deaktiviert. Wenn Sie später Unterstützung bei der Sperrvalidierung benötigen, finden Sie weitere Informationen unterAktualisierung einer CA (CLI).

Im folgenden Abschnitt finden Sie CLI-Beispiele.

CLI-Beispiele für die Erstellung einer privaten CA

Bei den folgenden Beispielen wird davon ausgegangen, dass Sie Ihr `.aws` Konfigurationsverzeichnis mit einer gültigen Standardregion, einem Endpunkt und Anmeldeinformationen eingerichtet haben. Informationen zur Konfiguration Ihrer AWS CLI Umgebung finden Sie unter Einstellungen für Konfiguration und Anmeldeinformationsdatei. Aus Gründen der besseren Lesbarkeit stellen wir in den Beispielbefehlen die CA-Konfiguration und die Sperreingabe als JSON-Dateien bereit. Ändern Sie die Beispieldateien nach Bedarf für Ihre Verwendung.

Alle Beispiele verwenden die folgende `ca_config.txt` Konfigurationsdatei, sofern nicht anders angegeben.

Datei: ca_config.txt

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

Beispiel 1: Erstellen Sie eine CA mit aktiviertem OCSP

In diesem Beispiel aktiviert die Sperrdatei die standardmäßige OCSP-Unterstützung, die den AWS Private CA Responder verwendet, um den Zertifikatsstatus zu überprüfen.

Datei: revoke_config.txt für OCSP

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

Befehl

```
$ 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
```

Bei Erfolg gibt dieser Befehl den Amazon-Ressourcennamen (ARN) der neuen CA aus.

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

Befehl

```
$ 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
```

Bei Erfolg gibt dieser Befehl den Amazon-Ressourcennamen (ARN) der CA aus.

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

Verwenden Sie den folgenden Befehl, um die Konfiguration Ihrer CA zu überprüfen.

```
$ 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
```

Diese Beschreibung sollte den folgenden Abschnitt enthalten.

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

Beispiel 2: Erstellen Sie eine CA mit aktiviertem OCSP und einem benutzerdefinierten CNAME

In diesem Beispiel ermöglicht die Sperrdatei die benutzerdefinierte OCSP-Unterstützung. Der `OcspCustomCname` Parameter verwendet einen vollqualifizierten Domänennamen (FQDN) als Wert.

Wenn Sie in diesem Feld einen FQDN angeben, wird der FQDN anstelle der Standard-URL für den OCSP-Responder in die Authority Information Access-Erweiterung jedes ausgestellten Zertifikats AWS Private CA eingefügt. AWS Wenn ein Endpunkt ein Zertifikat empfängt, das den benutzerdefinierten FQDN enthält, fragt er diese Adresse nach einer OCSP-Antwort ab. Damit dieser Mechanismus funktioniert, müssen Sie zwei zusätzliche Maßnahmen ergreifen:

- Verwenden Sie einen Proxyserver, um den Datenverkehr, der bei Ihrem benutzerdefinierten FQDN eingeht, an den AWS OCSP-Responder weiterzuleiten.
- Fügen Sie Ihrer DNS-Datenbank einen entsprechenden CNAME-Eintrag hinzu.

> **Tip**
>
> Weitere Hinweise zur Implementierung einer vollständigen OCSP-Lösung mit einem benutzerdefinierten CNAME finden Sie unter. Passen Sie die OCSP-URL an für AWS Private CA

Hier ist zum Beispiel ein CNAME-Eintrag für benutzerdefiniertes OCSP, wie er in Amazon Route 53 erscheinen würde.

Datensatzname	Typ	Routing-Richtlinie	Unterscheidungsmerkmal	Bewerten/Weiterleiten des Datenverkehrs an
alternative.example.com	CNAME	Einfach	-	proxy.example.com

> **Note**
>
> Der Wert des CNAME-Werts darf kein Protokollpräfix wie „http://“ oder „https://“ enthalten.

Datei: revoke_config.txt für OCSP

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

Befehl

```
$ 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
```

Bei Erfolg gibt dieser Befehl den Amazon-Ressourcennamen (ARN) der CA aus.

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

Verwenden Sie den folgenden Befehl, um die Konfiguration Ihrer CA zu überprüfen.

```
$ 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
```

Diese Beschreibung sollte den folgenden Abschnitt enthalten.

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

Beispiel 3: Erstellen Sie eine CA mit einer angehängten CRL

In diesem Beispiel definiert die Sperrkonfiguration CRL-Parameter.

Datei: revoke_config.txt

```
{
   "CrlConfiguration":{
      "Enabled":true,
      "ExpirationInDays":7,
      "S3BucketName":"amzn-s3-demo-bucket"
   }
}
```

Befehl

```
$ 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
```

Bei Erfolg gibt dieser Befehl den Amazon-Ressourcennamen (ARN) der CA aus.

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

Verwenden Sie den folgenden Befehl, um die Konfiguration Ihrer CA zu überprüfen.

```
$ 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
```

Diese Beschreibung sollte den folgenden Abschnitt enthalten.

```
"RevocationConfiguration": {
   ...
   "CrlConfiguration": {
      "Enabled": true,
      "ExpirationInDays": 7,
      "S3BucketName": "amzn-s3-demo-bucket"
   },
   ...
}
```

Beispiel 4: Erstellen Sie eine CA mit einer angehängten CRL und aktiviertem benutzerdefinierten CNAME

In diesem Beispiel definiert die Sperrkonfiguration CRL-Parameter, die einen benutzerdefinierten CNAME enthalten.

Datei: revoke_config.txt

```
{
   "CrlConfiguration":{
      "Enabled":true,
      "ExpirationInDays":7,
      "CustomCname": "alternative.example.com",
      "S3BucketName":"amzn-s3-demo-bucket"
   }
}
```

Befehl

```
$ 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
```

Bei Erfolg gibt dieser Befehl den Amazon-Ressourcennamen (ARN) der CA aus.

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

Verwenden Sie den folgenden Befehl, um die Konfiguration Ihrer CA zu überprüfen.

```
$ 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
```

Diese Beschreibung sollte den folgenden Abschnitt enthalten.

```
"RevocationConfiguration": {
   ...
   "CrlConfiguration": {
      "Enabled": true,
      "ExpirationInDays": 7,
      "CustomCname": "alternative.example.com",
      "S3BucketName": "amzn-s3-demo-bucket",
   ...
   }
}
```

Beispiel 5: Erstellen Sie eine CA und geben Sie den Nutzungsmodus an

In diesem Beispiel wird der CA-Nutzungsmodus bei der Erstellung einer CA angegeben. Falls nicht angegeben, ist der Parameter für den Verwendungsmodus standardmäßig auf GENERAL_PURPOSE voreingestellt. In diesem Beispiel ist der Parameter auf

SHORT_LIVED_CERTIFICATE gesetzt, was bedeutet, dass die CA Zertifikate mit einer maximalen Gültigkeitsdauer von sieben Tagen ausstellt. In Situationen, in denen es unpraktisch ist, den Widerruf zu konfigurieren, läuft ein kurzlebiges Zertifikat, das kompromittiert wurde, im Rahmen des normalen Betriebs schnell ab. Folglich fehlt in dieser Beispielzertifizierungsstelle ein Sperrmechanismus.

> **Note**
>
> AWS Private CA führt keine Gültigkeitsprüfungen für Root-CA-Zertifikate durch.

```
$ 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
```

Verwenden Sie den describe-certificate-authorityBefehl in AWS CLI , um Details zur resultierenden Zertifizierungsstelle anzuzeigen, wie im folgenden Befehl dargestellt:

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

Beispiel 6: Erstellen Sie eine Zertifizierungsstelle für die Active Directory-Anmeldung

Sie können eine private Zertifizierungsstelle erstellen, die für die Verwendung im Enterprise NTAuth Store von Microsoft Active Directory (AD) geeignet ist und dort Kartenanmelde- oder Domänencontrollerzertifikate ausstellen kann. Informationen zum Importieren eines CA-Zertifikats in AD finden Sie unter So importieren Sie Zertifikate von Drittanbieter-Zertifizierungsstellen (CA) in den Enterprise Store. NTAuth

Das Microsoft-Tool certutil kann verwendet werden, um CA-Zertifikate in AD zu veröffentlichen, indem die Option aufgerufen wird. -dspublish Einem mit Certutil in AD veröffentlichten Zertifikat wird in der gesamten Gesamtstruktur vertraut. Mithilfe von Gruppenrichtlinien können Sie das Vertrauen auch auf eine Teilmenge der gesamten Gesamtstruktur beschränken, z. B. auf eine einzelne Domäne oder eine Gruppe von Computern in einer Domäne. Damit die Anmeldung funktioniert, muss die ausstellende Zertifizierungsstelle ebenfalls im NTAuth Store veröffentlicht sein. Weitere Informationen finden Sie unter Verteilen von Zertifikaten an Client-Computer mithilfe von Gruppenrichtlinien.

In diesem Beispiel wird die folgende `ca_config_AD.txt` Konfigurationsdatei verwendet.

Datei: 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"
         }
      ]
   }
}
```

Befehl

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

Bei Erfolg gibt dieser Befehl den Amazon-Ressourcennamen (ARN) der CA aus.

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

Verwenden Sie den folgenden Befehl, um die Konfiguration Ihrer CA zu überprüfen.

```
$ 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
```

Diese Beschreibung sollte den folgenden Abschnitt enthalten.

```
...

"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"
      }
   ]
}
...
```

Beispiel 7: Erstellen Sie eine Themen-CA mit angehängter CRL und ohne CDP-Erweiterung in ausgestellten Zertifikaten

Sie können eine private Zertifizierungsstelle erstellen, die für die Ausstellung von Zertifikaten für den Matter Smart Home-Standard geeignet ist. In diesem Beispiel `ca_config_PAA.txt` definiert die CA-Konfiguration eine Matter Product Attestation Authority (PAA) mit der Vendor-ID (VID) auf. FFF1

Datei: 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"
      }
    ]
  }
}
```

Die Sperrkonfiguration aktiviert und konfiguriert die CA so CRLs, dass die Standard-CDP-URL in allen ausgestellten Zertifikaten weggelassen wird.

Datei: revoke_config.txt

```
{
   "CrlConfiguration":{
      "Enabled":true,
      "ExpirationInDays":7,
      "S3BucketName":"amzn-s3-demo-bucket",
   "CrlDistributionPointExtensionConfiguration":{
  "OmitExtension":true
   }
   }
}
```

Befehl

```
$ 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
```

Bei Erfolg gibt dieser Befehl den Amazon-Ressourcennamen (ARN) der CA aus.

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

Verwenden Sie den folgenden Befehl, um die Konfiguration Ihrer CA zu überprüfen.

```
$ 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
```

Diese Beschreibung sollte den folgenden Abschnitt enthalten.

```
"RevocationConfiguration": {
   ...
   "CrlConfiguration": {
      "Enabled": true,
      "ExpirationInDays": 7,
      "S3BucketName": "amzn-s3-demo-bucket",
   "CrlDistributionPointExtensionConfiguration":{
  "OmitExtension":true
   }
   },
   ...
}
...
```

Installation des CA-Zertifikats

Führen Sie die folgenden Verfahren durch, um ein Zertifikat für Ihre private Zertifizierungsstelle (Certificate Authority, CA) zu erstellen und zu installieren. Ihre CA ist dann einsatzbereit.

AWS Private CA unterstützt drei Szenarien für die Installation eines CA-Zertifikats:

- Installation eines Zertifikats für eine Stammzertifizierungsstelle, gehostet von AWS Private CA
- Installieren eines untergeordneten CA-Zertifikats, dessen übergeordnete Zertifizierungsstelle von AWS Private CA gehostet wird
- Installieren eines untergeordneten CA-Zertifikats, dessen übergeordnete Zertifizierungsstelle extern gehostet wird

In den folgenden Abschnitten werden die Verfahren für jedes Szenario beschrieben. Die Konsolenprozeduren beginnen auf der Konsolenseite Private CAs.

Kompatible Signaturalgorithmen

Die Unterstützung von Signaturalgorithmen für CA-Zertifikate hängt vom Signaturalgorithmus der übergeordneten Zertifizierungsstelle und von der ab AWS-Region. Die folgenden Einschränkungen gelten sowohl für die Konsole als auch für den AWS CLI Betrieb.

- Eine übergeordnete Zertifizierungsstelle mit dem RSA-Schlüsselalgorithmus kann Zertifikate mit den folgenden Signaturalgorithmen ausstellen:
  - SHA256 RSA

- SHA384 RSA
- SHA512 RSA

- In einer älteren Version AWS-Region kann eine übergeordnete Zertifizierungsstelle mit dem EDCSA-Schlüsselalgorithmus Zertifikate mit den folgenden Signaturalgorithmen ausstellen:
  - SHA256 ECDSA
  - SHA384 ECDSA
  - SHA512 ECDSA

  Zu den älteren Versionen gehören AWS-Regionen :

Name der Region	Geografischer Standort
eu-north-1	Europa (Stockholm)
me-south-1	Middle East (Bahrain)
ap-south-1	Asien-Pazifik (Mumbai)
eu-west-3	Europa (Paris)
us-east-2	USA Ost (Ohio)
af-south-1	Afrika (Kapstadt)
eu-west-1	Europa (Irland)
eu-central-1	Europa (Frankfurt)
sa-east-1	Südamerika (São Paulo)
ap-east-1	Asien-Pazifik (Hongkong)

Name der Region	Geografischer Standort
us-east-1	USA Ost (Nord-Virginia)
ap-northeast-2	Asien-Pazifik (Seoul)
eu-west-2	Europa (London)
ap-northeast-1	Asien-Pazifik (Tokio)
us-gov-east-1	AWS GovCloud (US-Ost)
us-gov-west-1	AWS GovCloud (US-West)
us-west-2	USA West (Oregon)
us-west-1	USA West (Nordkalifornien)
ap-southeast-1	Asien-Pazifik (Singapur)
ap-southeast-2	Asien-Pazifik (Sydney)

- In einem nicht älteren System AWS-Region gelten für EDCSA die folgenden Regeln:
  - Eine übergeordnete Zertifizierungsstelle mit dem EC_Prime256V1-Signaturalgorithmus kann Zertifikate mit ECDSA P256 ausstellen.
  - Eine übergeordnete Zertifizierungsstelle mit dem EC_SECP384R1-Signaturalgorithmus kann Zertifikate mit ECDSA P384 ausstellen.
- In jedem Fall gelten die folgenden Regeln für EDCSA: AWS-Region
  - Eine übergeordnete Zertifizierungsstelle mit dem EC_SECP521R1-Signaturalgorithmus kann Zertifikate mit ECDSA P521 ausstellen.

Installieren Sie ein Root-CA-Zertifikat

Sie können ein Root-CA-Zertifikat über den AWS-Managementkonsole oder den installieren AWS CLI.

Um ein Zertifikat für Ihre private Root-CA (Konsole) zu erstellen und zu installieren

1. (Optional) Wenn Sie sich noch nicht auf der Detailseite der Zertifizierungsstelle befinden, öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause. Wählen Sie auf der Seite Private Zertifizierungsstellen eine Stammzertifizierungsstelle mit dem Status Ausstehendes Zertifikat oder Aktiv aus.
2. Wählen Sie Aktionen, CA-Zertifikat installieren, um die Seite Root-CA-Zertifikat installieren zu öffnen.
3. Geben Sie unter Parameter für das Root-CA-Zertifikat angeben die folgenden Zertifikatsparameter an:

   - Gültigkeit — Gibt das Ablaufdatum und die Uhrzeit für das CA-Zertifikat an. Die AWS Private CA Standardgültigkeitsdauer für ein Root-CA-Zertifikat beträgt 10 Jahre.
   - Signaturalgorithmus — Gibt den Signaturalgorithmus an, der verwendet werden soll, wenn die Stammzertifizierungsstelle neue Zertifikate ausstellt. Die verfügbaren Optionen variieren je nachdem AWS-Region , wo Sie die Zertifizierungsstelle erstellen. Weitere Informationen finden Sie unter Kompatible SignaturalgorithmenUnterstützte kryptografische Algorithmen in AWS Private Certificate Authority, und SigningAlgorithmin CertificateAuthorityConfiguration.
     - SHA256 RSA
     - SHA384 RSA
     - SHA512 RSA

   Überprüfen Sie Ihre Einstellungen auf Richtigkeit und wählen Sie dann Bestätigen und installieren. AWS Private CA exportiert eine CSR für Ihre CA, generiert ein Zertifikat mithilfe einer Root-CA-Zertifikatsvorlage und signiert das Zertifikat selbst. AWS Private CA importiert dann das selbstsignierte Root-CA-Zertifikat.
4. Auf der Detailseite für die Zertifizierungsstelle wird der Status der Installation (erfolgreich oder fehlgeschlagen) oben angezeigt. Wenn die Installation erfolgreich war, zeigt die neu abgeschlossene Stammzertifizierungsstelle im Bereich Allgemein den Status Aktiv an.

Um ein Zertifikat für Ihre private Stammzertifizierungsstelle zu erstellen und zu installieren (AWS CLI)

1. Generieren Sie eine Zertifikatsignieranforderung (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
```

Die resultierende Dateica.csr, eine im Base64-Format codierte PEM-Datei, hat das folgende Aussehen.

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

Sie können OpenSSL verwenden, um den Inhalt der CSR anzuzeigen und zu überprüfen.

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

Dies führt zu einer Ausgabe, die der folgenden ähnelt.

```
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. Verwenden Sie die CSR aus dem vorherigen Schritt als Argument für den `--csr` Parameter und stellen Sie das Stammzertifikat aus.

> **Note**
>
> Wenn Sie AWS CLI Version 1.6.3 oder höher verwenden, verwenden Sie das Präfix, `fileb://` wenn Sie die erforderliche Eingabedatei angeben. Dadurch wird sichergestellt, dass die AWS Private CA Base64-kodierten Daten korrekt analysiert werden.

```
$ 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. Rufen Sie das Stammzertifikat ab.

```
$ 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
```

Die resultierende Datei`cert.pem`, eine im Base64-Format codierte PEM-Datei, hat das folgende Aussehen.

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

Sie können OpenSSL verwenden, um den Inhalt des Zertifikats anzuzeigen und zu überprüfen.

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

Dies führt zu einer Ausgabe, die der folgenden ähnelt.

```
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. Importieren Sie das Root-CA-Zertifikat, um es auf der CA zu installieren.

> **Note**
>
> Wenn Sie AWS CLI Version 1.6.3 oder höher verwenden, verwenden Sie das Präfix, `fileb://` wenn Sie die erforderliche Eingabedatei angeben. Dadurch wird sichergestellt, dass die AWS Private CA Base64-kodierten Daten korrekt analysiert werden.

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

Überprüfen Sie den neuen Status der CA.

```
$ aws acm-pca describe-certificate-authority \
 --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
 --output json
```

Der Status wird jetzt als AKTIV angezeigt.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-05T14:24:12.867000-08:00",
        "LastStateChangeAt": "2021-03-08T12:37:14.235000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T07:46:27-08:00",
        "NotAfter": "2022-03-08T08:46:27-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
```

```
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "CustomCname": "alternative.example.com",
                "S3BucketName": "amzn-s3-demo-bucket"
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

Installieren Sie ein untergeordnetes CA-Zertifikat, das von gehostet wird AWS Private CA

Sie können das verwenden AWS-Managementkonsole , um ein Zertifikat für Ihre AWS Private CA gehostete untergeordnete Zertifizierungsstelle zu erstellen und zu installieren.

Um ein Zertifikat für Ihre AWS Private CA gehostete untergeordnete Zertifizierungsstelle zu erstellen und zu installieren

1. (Optional) Wenn Sie sich noch nicht auf der Detailseite der Zertifizierungsstelle befinden, öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause. Wählen Sie auf der Seite Private Zertifizierungsstellen eine untergeordnete Zertifizierungsstelle mit dem Status Ausstehendes Zertifikat oder Aktiv aus.
2. Wählen Sie Aktionen, CA-Zertifikat installieren, um die Seite Untergeordnetes CA-Zertifikat installieren zu öffnen.
3. Wählen Sie auf der Seite Zertifikat der untergeordneten Zertifizierungsstelle installieren unter Typ der Zertifizierungsstelle auswählen aus, dass ein Zertifikat installiert werden AWS Private CAsoll, das von verwaltet wird. AWS Private CA

4. Wählen Sie unter Übergeordnete Zertifizierungsstelle auswählen eine Zertifizierungsstelle aus der Liste Übergeordnete private Zertifizierungsstelle aus. Die Liste wird so gefiltert, CAs dass sie angezeigt wird, die die folgenden Kriterien erfüllen:

   - Sie sind berechtigt, die CA zu verwenden.
   - Die CA würde sich nicht selbst signieren.
   - Die CA befindet sich im Status`ACTIVE`.
   - Der CA-Modus ist`GENERAL_PURPOSE`.

5. Geben Sie unter Geben Sie die untergeordneten CA-Zertifikatsparameter an die folgenden Zertifikatsparameter an:

   - Gültigkeit — Gibt das Ablaufdatum und die Uhrzeit für das CA-Zertifikat an.
   - Signaturalgorithmus — Gibt den Signaturalgorithmus an, der verwendet werden soll, wenn die Stammzertifizierungsstelle neue Zertifikate ausstellt. Folgende Optionen sind vorhanden:
     - SHA256 RSA
     - SHA384 RSA
     - SHA512 RSA
   - Pfadlänge — Die Anzahl der Vertrauensebenen, die die untergeordnete Zertifizierungsstelle beim Signieren neuer Zertifikate hinzufügen kann. Eine Pfadlänge von Null (Standard) bedeutet, dass nur Endentitätszertifikate und keine CA-Zertifikate erstellt werden können. Eine Pfadlänge von eins oder mehreren bedeutet, dass die untergeordnete Zertifizierungsstelle Zertifikate ausstellen kann, um ihr weitere CAs untergeordnete Zertifizierungsstellen zu erstellen.
   - Vorlagen-ARN — Zeigt den ARN der Konfigurationsvorlage für dieses CA-Zertifikat an. Die Vorlage ändert sich, wenn Sie die angegebene Pfadlänge ändern. Wenn Sie ein Zertifikat mit dem CLI-Befehl issue-certificate oder der IssueCertificateAPI-Aktion erstellen, müssen Sie den ARN manuell angeben. Informationen zu verfügbaren CA-Zertifikatvorlagen finden Sie unter Verwenden Sie Zertifikatsvorlagen AWS Private CA.

6. Überprüfen Sie Ihre Einstellungen auf Richtigkeit und wählen Sie dann Bestätigen und installieren aus. AWS Private CA exportiert eine CSR, generiert ein Zertifikat mithilfe einer untergeordneten Zertifizierungsstelle und signiert das Zertifikat mit der ausgewählten übergeordneten Zertifizierungsstelle. AWS Private CA importiert dann das signierte Zertifikat der untergeordneten Zertifizierungsstelle.

7. Auf der Detailseite für die Zertifizierungsstelle wird der Status der Installation (erfolgreich oder fehlgeschlagen) oben angezeigt. Wenn die Installation erfolgreich war, zeigt die neu abgeschlossene untergeordnete Zertifizierungsstelle im Bereich Allgemein den Status Aktiv an.

Installieren Sie ein Zertifikat einer untergeordneten Zertifizierungsstelle, das von einer externen übergeordneten Zertifizierungsstelle signiert wurde

Nachdem Sie, wie unter beschrieben, eine untergeordnete private Zertifizierungsstelle erstellt habenErstellen Sie eine private CA in AWS Private CA, haben Sie die Möglichkeit, sie zu aktivieren, indem Sie ein von einer externen Signaturstelle signiertes CA-Zertifikat installieren. Um Ihr Zertifikat einer untergeordneten Zertifizierungsstelle mit einer externen Zertifizierungsstelle zu signieren, müssen Sie zunächst einen externen Vertrauensdienstanbieter als Ihre Signaturbehörde einrichten oder die Nutzung eines Drittanbieters vereinbaren.

> **Note**
>
> Verfahren zur Einrichtung oder Beschaffung eines externen Vertrauensdienstanbieters fallen nicht in den Geltungsbereich dieses Handbuchs.

Nachdem Sie eine untergeordnete Zertifizierungsstelle erstellt haben und Zugriff auf eine externe Signaturstelle haben, führen Sie die folgenden Aufgaben aus:

1. Besorgen Sie sich eine Zertifikatsignieranforderung (CSR) von. AWS Private CA
2. Reichen Sie die CSR an Ihre externe Signaturbehörde ein und erhalten Sie ein signiertes CA-Zertifikat zusammen mit allen Kettenzertifikaten.
3. Importieren Sie das CA-Zertifikat und die Kette in AWS Private CA , um Ihre untergeordnete Zertifizierungsstelle zu aktivieren.

Die detaillierten Schritte finden Sie unter Verwenden Sie extern signierte private CA-Zertifikate.

Steuern Sie den Zugriff auf die private CA

Jeder Benutzer mit den erforderlichen Berechtigungen für eine private Zertifizierungsstelle AWS Private CA kann diese Zertifizierungsstelle verwenden, um andere Zertifikate zu signieren. Der Besitzer der Zertifizierungsstelle kann Zertifikate ausstellen oder die erforderlichen Berechtigungen

für die Ausstellung von Zertifikaten an einen AWS Identity and Access Management (IAM-) Benutzer delegieren, der sich in derselben befindet. AWS-KontoEin Benutzer, der in einem anderen AWS Konto ansässig ist, kann auch Zertifikate ausstellen, wenn er vom Eigentümer der Zertifizierungsstelle im Rahmen einer ressourcenbasierten Richtlinie autorisiert wurde.

Autorisierte Benutzer, unabhängig davon, ob es sich um ein einzelnes Konto oder um ein Konto mit mehreren Konten handelt, können unsere Ressourcen bei der Ausstellung von Zertifikaten verwenden AWS Private CA . AWS Certificate Manager Zertifikate, die über den AWS Private CA IssueCertificateAPI-Befehl oder den CLI-Befehl issue-certificate ausgestellt wurden, werden nicht verwaltet. Solche Zertifikate erfordern eine manuelle Installation auf den Zielgeräten und eine manuelle Verlängerung, wenn sie ablaufen. Zertifikate, die über die ACM-Konsole, die RequestCertificateACM-API oder den CLI-Befehl request-certificate ausgestellt wurden, werden verwaltet. Solche Zertifikate können problemlos in Diensten installiert werden, die in ACM integriert sind. Wenn der CA-Administrator dies zulässt und das Konto des Ausstellers über eine dienstbezogene Rolle für ACM verfügt, werden verwaltete Zertifikate nach Ablauf automatisch erneuert.

Themen

- Erstellen Sie Einzelkontoberechtigungen für einen IAM-Benutzer
- Fügen Sie eine Richtlinie für den kontoübergreifenden Zugriff bei

Erstellen Sie Einzelkontoberechtigungen für einen IAM-Benutzer

Wenn der Zertifizierungsstellenadministrator (d. h. der Besitzer der Zertifizierungsstelle) und der Zertifikatsaussteller in einem einzigen AWS Konto ansässig sind, besteht eine bewährte Methode darin, die Rollen des Ausstellers und des Administrators zu trennen, indem ein AWS Identity and Access Management (IAM-) Benutzer mit eingeschränkten Rechten erstellt wird. Informationen zur Verwendung von IAM mit AWS Private CA sowie Beispielberechtigungen finden Sie unter. Identity and Access Management (IAM) für AWS Private Certificate Authority

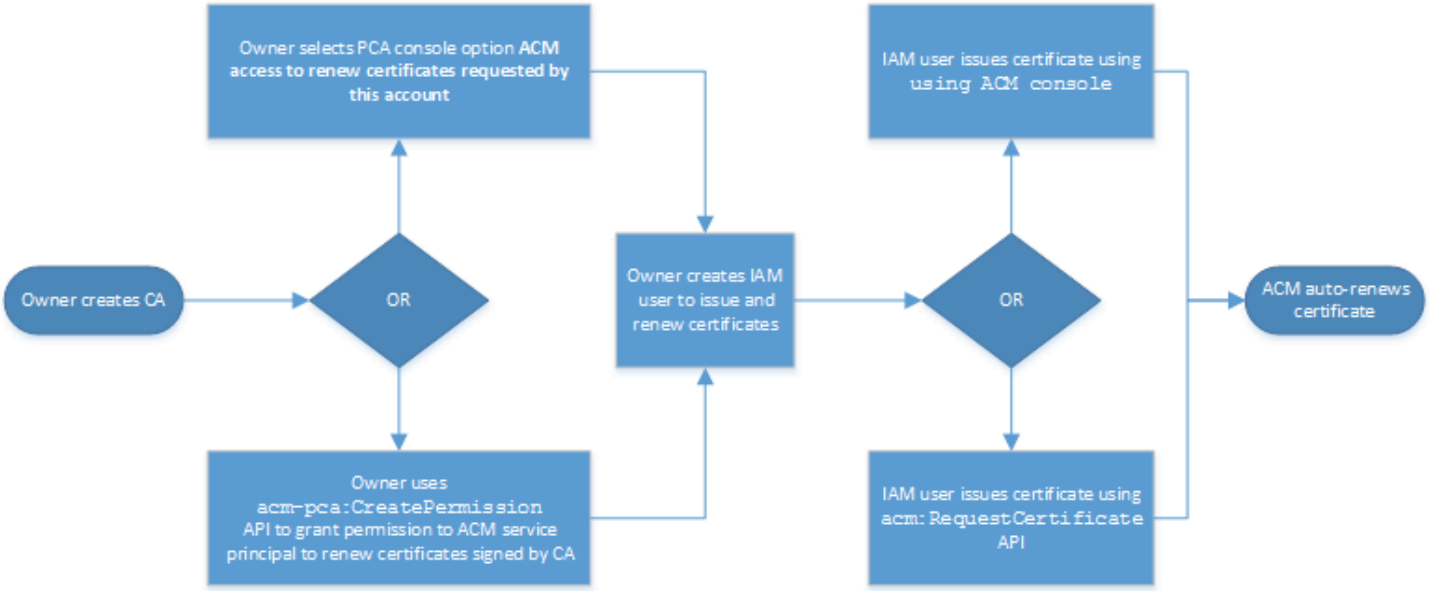
Fall 1 für ein einzelnes Konto: Ausstellung eines nicht verwalteten Zertifikats

In diesem Fall erstellt der Kontoinhaber eine private Zertifizierungsstelle und anschließend einen IAM-Benutzer mit der Berechtigung, von der privaten Zertifizierungsstelle signierte Zertifikate auszustellen. Der IAM-Benutzer stellt ein Zertifikat aus, indem er die AWS Private CA `IssueCertificate` API aufruft.

Auf diese Weise ausgestellte Zertifikate werden nicht verwaltet, was bedeutet, dass ein Administrator sie exportieren und auf Geräten installieren muss, auf denen sie verwendet werden sollen. Sie müssen außerdem manuell erneuert werden, wenn sie ablaufen. Die Ausstellung eines Zertifikats mithilfe dieser API erfordert eine Zertifikatsignieranforderung (CSR) und ein key pair, die außerhalb AWS Private CA von OpenSSL oder einem ähnlichen Programm generiert werden. Weitere Informationen finden Sie in der `IssueCertificate` Dokumentation.

Fall 2: Einzelkonto: Ausstellung eines verwalteten Zertifikats über ACM

Dieser zweite Fall beinhaltet API-Operationen sowohl von ACM als auch von PCA. Der Kontoinhaber erstellt wie zuvor einen privaten CA- und IAM-Benutzer. Der Kontoinhaber erteilt dem ACM-Dienstprinzipal dann die Erlaubnis, alle von dieser CA signierten Zertifikate automatisch zu erneuern. Der IAM-Benutzer stellt das Zertifikat erneut aus, diesmal jedoch durch Aufrufen der `RequestCertificate` ACM-API, die für die CSR und die Schlüsselgenerierung zuständig ist. Wenn das Zertifikat abläuft, automatisiert ACM den Verlängerungsablauf.



Der Kontoinhaber hat die Möglichkeit, während oder nach der Erstellung der Zertifizierungsstelle oder mithilfe der `CreatePermission` PCA-API über die Verwaltungskonsole Verlängerungsberechtigungen zu erteilen. Die aus diesem Workflow erstellten verwalteten Zertifikate können mit AWS Diensten verwendet werden, die in ACM integriert sind.

Der folgende Abschnitt enthält Verfahren zur Erteilung von Verlängerungsberechtigungen.

Weisen Sie ACM Berechtigungen zur Zertifikatsverlängerung zu

Mit Managed Renewal in AWS Certificate Manager (ACM) können Sie den Prozess der Zertifikatserneuerung sowohl für öffentliche als auch für private Zertifikate automatisieren. Damit ACM die von einer privaten Zertifizierungsstelle generierten Zertifikate automatisch erneuern kann, muss dem ACM-Dienstprinzipal von der CA selbst alle möglichen Berechtigungen erteilt werden. Wenn diese Verlängerungsberechtigungen für ACM nicht vorhanden sind, muss der Eigentümer der Zertifizierungsstelle (oder ein bevollmächtigter Vertreter) jedes private Zertifikat manuell neu ausstellen, wenn es abläuft.

> **Important**
>
> Diese Verfahren für die Zuweisung von Verlängerungsberechtigungen gelten nur, wenn sich der Eigentümer der Zertifizierungsstelle und der Zertifikatsaussteller im selben Konto befinden. AWS Informationen zu kontenübergreifenden Szenarien finden Sie unter. Fügen Sie eine Richtlinie für den kontoübergreifenden Zugriff bei

Erneuerungsberechtigungen können während der Erstellung einer privaten CA delegiert oder jederzeit geändert werden, sofern sich die CA im `ACTIVE`-Status befindet.

Sie können private CA-Berechtigungen über die AWS Private CA -Konsole, die AWS Command Line Interface (AWS CLI) oder die AWS Private CA -API verwalten:

So weisen Sie ACM (Konsole) private CA-Berechtigungen zu

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause.
2. Wählen Sie auf der Seite Private Zertifizierungsstellen Ihre private Zertifizierungsstelle aus der Liste aus.
3. Wählen Sie Aktionen, CA-Berechtigungen konfigurieren aus.
4. Wählen Sie ACM-Zugriff autorisieren aus, um die von diesem Konto angeforderten Zertifikate zu erneuern.
5. Wählen Sie Speichern.

Um ACM-Berechtigungen in () zu verwalten AWS Private CA AWS CLI

Verwenden Sie den Befehl create-permission, um ACM Berechtigungen zuzuweisen. Sie müssen die erforderlichen Berechtigungen (`IssueCertificate`, und`ListPermissions`) zuweisen`GetCertificate`, damit ACM Ihre Zertifikate automatisch erneuern kann.

```
$ 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
```

Verwenden Sie den Befehl list-permissions, um die von einer CA delegierten Berechtigungen aufzulisten.

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

Verwenden Sie den Befehl delete-permission, um die einem Dienstprinzipal von einer Zertifizierungsstelle zugewiesenen Berechtigungen zu widerrufen. AWS

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

Fügen Sie eine Richtlinie für den kontoübergreifenden Zugriff bei

Wenn sich der Zertifizierungsstellenadministrator und der Zertifikatsaussteller in unterschiedlichen AWS Konten befinden, muss sich der Zertifizierungsstellenadministrator den Zugriff auf die Zertifizierungsstelle teilen. Dies wird erreicht, indem eine ressourcenbasierte Richtlinie an die Zertifizierungsstelle angehängt wird. Die Richtlinie gewährt einem bestimmten Prinzipal, bei dem es sich um einen AWS Kontoinhaber, einen IAM-Benutzer, eine ID oder eine AWS Organizations Organisationseinheits-ID handeln kann, Erteilungsberechtigungen.

Ein CA-Administrator kann Richtlinien auf folgende Weise anhängen und verwalten:

- In der Managementkonsole mithilfe von AWS Resource Access Manager (RAM), einer Standardmethode für die gemeinsame Nutzung von AWS Ressourcen zwischen Konten. Wenn Sie eine CA-Ressource AWS RAM mit einem Principal in einem anderen Konto gemeinsam nutzen,

wird die erforderliche ressourcenbasierte Richtlinie automatisch an die CA angehängt. Weitere Informationen zu RAM finden Sie im AWS RAM Benutzerhandbuch.

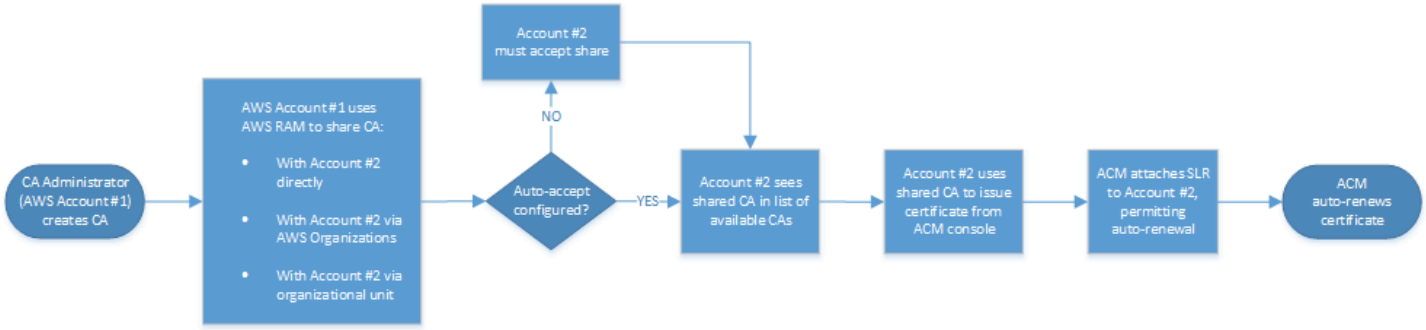
> **Note**
>
> Sie können die RAM-Konsole ganz einfach öffnen, indem Sie eine Zertifizierungsstelle und dann Aktionen, Ressourcenfreigaben verwalten auswählen.

- Programmgesteuert, mithilfe der PCA APIs PutPolicy, und GetPolicy. DeletePolicy
- Manuell mithilfe der PCA-Befehle put-policy, get-policy und delete-policy in der. AWS CLI

Nur für die Konsolenmethode ist RAM-Zugriff erforderlich.

Kontoübergreifender Fall 1: Ausstellung eines verwalteten Zertifikats über die Konsole

In diesem Fall verwendet der CA-Administrator AWS Resource Access Manager (AWS RAM), um den CA-Zugriff mit einem anderen AWS Konto zu teilen, sodass dieses Konto verwaltete ACM-Zertifikate ausstellen kann. Das Diagramm zeigt, dass AWS RAM die Zertifizierungsstelle direkt mit dem Konto oder indirekt über eine AWS Organizations ID, der das Konto angehört, gemeinsam genutzt werden kann.



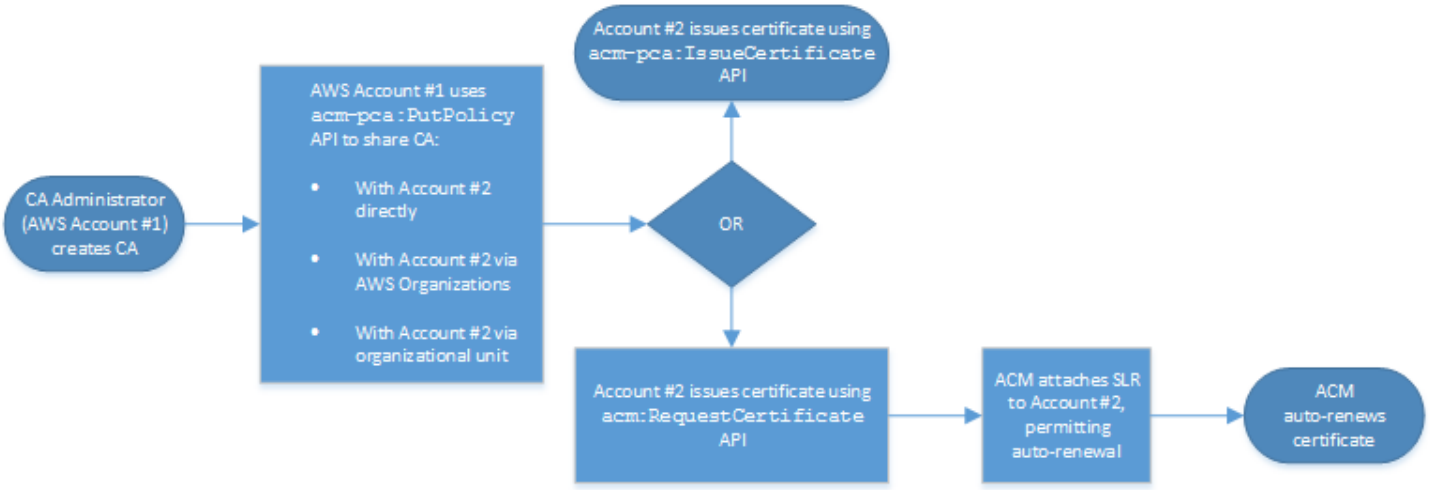
Nachdem RAM eine Ressource gemeinsam genutzt hat AWS Organizations, muss der Principal des Empfängers die Ressource akzeptieren, damit sie wirksam wird. Der Empfänger kann so konfigurieren AWS Organizations , dass angebotene Shares automatisch akzeptiert werden.

> **Note**
>
> Das Empfängerkonto ist für die Konfiguration der automatischen Verlängerung in ACM verantwortlich. In der Regel installiert ACM bei der ersten Verwendung einer gemeinsam genutzten Zertifizierungsstelle eine dienstbezogene Rolle, die es ermöglicht, unbeaufsichtigte Zertifikatsanrufe zu tätigen. AWS Private CA Schlägt dies fehl (in der Regel aufgrund einer

fehlenden Berechtigung), werden die Zertifikate der Zertifizierungsstelle nicht automatisch erneuert. Nur der ACM-Benutzer kann das Problem lösen, nicht der CA-Administrator. Weitere Informationen finden Sie unter Verwenden einer Service Linked Role (SLR) mit ACM.

Kontoübergreifender Fall 2: Ausstellung verwalteter und nicht verwalteter Zertifikate mithilfe der API oder CLI

In diesem zweiten Fall werden die Optionen für die gemeinsame Nutzung und Ausstellung veranschaulicht, die mithilfe der AWS Certificate Manager API und möglich sind. AWS Private CA All diese Operationen können auch mit den entsprechenden AWS CLI Befehlen ausgeführt werden.



Da die API-Operationen in diesem Beispiel direkt verwendet werden, hat der Zertifikatsaussteller die Wahl zwischen zwei API-Vorgängen, um ein Zertifikat auszustellen. Die PCA-API-Aktion `IssueCertificate` führt zu einem nicht verwalteten Zertifikat, das nicht automatisch erneuert wird und exportiert und manuell installiert werden muss. Die ACM-API-Aktion RequestCertificateführt zu einem verwalteten Zertifikat, das einfach auf integrierten ACM-Diensten installiert werden kann und automatisch erneuert wird.

> **Note**
>
> Das Empfängerkonto ist für die Konfiguration der automatischen Verlängerung in ACM verantwortlich. In der Regel installiert ACM bei der ersten Verwendung einer gemeinsam genutzten Zertifizierungsstelle eine dienstbezogene Rolle, über die Zertifikate unbeaufsichtigt abgerufen werden können. AWS Private CA Schlägt dies fehl (in der Regel aufgrund einer fehlenden Berechtigung), werden die Zertifikate der Zertifizierungsstelle nicht automatisch

verlängert, und nur der ACM-Benutzer kann das Problem lösen, nicht der CA-Administrator. Weitere Informationen finden Sie unter Verwenden einer Service Linked Role (SLR) mit ACM.

Liste privat CAs

Sie können die AWS Private CA Konsole verwenden oder private Daten AWS CLI auflisten CAs , deren Eigentümer Sie sind oder auf die Sie Zugriff haben.

Um die verfügbaren Produkte CAs über die Konsole aufzulisten

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause.
2. Überprüfen Sie die Informationen in der Liste der privaten Zertifizierungsstellen. Sie können CAs mithilfe der Seitenzahlen oben rechts durch mehrere Seiten navigieren. Jede Zertifizierungsstelle belegt eine Zeile, für die jeweils einige oder alle der folgenden Spalten angezeigt werden:

- Betreff — Zusammenfassung der Informationen zu den eindeutigen Namen der Zertifizierungsstelle.
- Id — 32-Byte-Hexadezimaler eindeutiger Bezeichner der CA.
- Status — Status der Zertifizierungsstelle. Mögliche Werte sind Erstellt, Ausstehendes Zertifikat, Aktiv, Gelöscht, Deaktiviert, Abgelaufen und Fehlgeschlagen.
- Typ — Der Typ der Zertifizierungsstelle. Mögliche Werte sind Root und Subordinate.
- Modus — Der Modus der CA. Mögliche Werte sind General-Purpose (stellt Zertifikate aus, die mit einem beliebigen Ablaufdatum konfiguriert werden können) und Kurzlebiges Zertifikat (stellt Zertifikate mit einer maximalen Gültigkeitsdauer von sieben Tagen aus). Eine kurze Gültigkeitsdauer kann in einigen Fällen einen Sperrmechanismus ersetzen. Die Standardeinstellung ist Allzweck.
- Besitzer — Das AWS Konto, dem die CA gehört. Dies kann Ihr Konto oder ein Konto sein, für das Verwaltungsberechtigungen der Zertifizierungsstelle an Sie delegiert wurden.
- Schlüsselalgorithmus — Der von der CA unterstützte Public-Key-Algorithmus. Mögliche Werte sind ML_DSA_44, ML_DSA_65, ML_DSA_87, RSA_2048, RSA_3072, RSA_4096, EC_Prime256V1, EC_SECP384R1 und EC_SECP521R1.
- Signaturalgorithmus — Der Algorithmus, den die CA zum Signieren von Zertifikatsanfragen verwendet. (Nicht zu verwechseln mit dem `SigningAlgorithm` Parameter, der zum Signieren

von Zertifikaten verwendet wird, wenn sie ausgestellt werden.) Mögliche Werte sind ML_DSA_44, ML_DSA_65, ML_DSA_87, WITRSA, WITRSA, WITRSA, WITHECDSA, WITHECDSA und WITHECDSA. SHA256 SHA384 SHA512 SHA256 SHA384 SHA512

> **Note**
>
> Sie können die Spalten, die Sie anzeigen möchten, sowie andere Einstellungen anpassen, indem Sie das Einstellungssymbol in der oberen rechten Ecke der Konsole auswählen.

Zur Liste verfügbarer Dateien verwenden Sie CAs AWS CLI

Verwenden Sie den list-certificate-authoritiesBefehl, um die verfügbaren Artikel aufzulisten, CAs wie im folgenden Beispiel gezeigt:

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

Die vom System zurückgegebenen Informationen ähneln den Folgenden:

```
{
   "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
            }
```

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

Eine private Zertifizierungsstelle anzeigen

Sie können die ACM-Konsole oder die verwenden AWS CLI , um detaillierte Metadaten zu einer privaten Zertifizierungsstelle anzuzeigen und mehrere Werte nach Bedarf zu ändern. Ausführliche Informationen zur Aktualisierung finden Sie CAs unterAktualisieren Sie eine private Zertifizierungsstelle in AWS Private Certificate Authority.

So zeigen Sie CA-Details in der Konsole an

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause.
2. Sehen Sie sich die Liste der privaten Zertifizierungsstellen an. CAs Mithilfe der Seitenzahlen oben rechts können Sie durch mehrere Seiten navigieren.
3. Um detaillierte Metadaten für eine aufgelistete Zertifizierungsstelle anzuzeigen, wählen Sie das Optionsfeld neben der Zertifizierungsstelle aus, die Sie überprüfen möchten. Dadurch wird ein Detailbereich mit den folgenden Registerkarten geöffnet:

   - Registerkarte „Betreff“ — Informationen zum eindeutigen Namen der Zertifizierungsstelle. Weitere Informationen finden Sie unter Subject distinguished name. Zu den angezeigten Feldern gehören:
     - Betreff — Zusammenfassung der angegebenen Namensinformationsfelder
     - Organisation (O) — Zum Beispiel ein Firmenname
     - Organisationseinheit (OU) — Zum Beispiel eine Abteilung innerhalb eines Unternehmens
     - Ländername (C) — Ein aus zwei Buchstaben bestehender Ländercode
     - Name des Bundesstaates oder der Provinz — Vollständiger Name eines Bundesstaates oder einer Provinz
     - Ortsname — Der Name einer Stadt
     - Allgemeiner Name (CN) — Eine für Menschen lesbare Zeichenfolge zur Identifizierung der CA.
   - Registerkarte CA-Zertifikat — Informationen zur Gültigkeit des CA-Zertifikats

- Gültig bis — Datum und Uhrzeit, bis das CA-Zertifikat gültig ist
- Läuft ab in — Die Anzahl der Tage bis zum Ablauf

- Registerkarte „Sperrkonfiguration“ — Ihre aktuellen Optionen für den Widerruf von Zertifikaten. Wählen Sie Bearbeiten, um zu aktualisieren.
  - Verteilung der Zertifikatsperrliste (CRL) — Status „Aktiviert“ oder „Deaktiviert“
  - Online Certificate Status Protocol (OCSP) — Status „Aktiviert“ oder „Deaktiviert“
- Registerkarte „Berechtigungen“ — Ihre aktuelle Auswahl an Berechtigungen zur Zertifikatserneuerung für diese Zertifizierungsstelle AWS Certificate Manager (ACM). Wählen Sie Bearbeiten, um zu aktualisieren.
- ACM-Autorisierung für Verlängerungen — Status „Autorisiert“ oder „Nicht autorisiert“
- Registerkarte „Tags“ — Ihre aktuelle Zuweisung von anpassbaren Labels für diese CA. Wählen Sie „Zu aktualisierende Tags verwalten“ aus.
- Registerkarte „Resource Shares“ — Ihre aktuelle Zuweisung von Ressourcenfreigaben für diese CA durch AWS Resource Access Manager (RAM). Wählen Sie zu aktualisierende Ressourcenfreigaben verwalten aus.
  - Name — Name der Ressourcenfreigabe
  - Status — Status der Ressourcenfreigabe

4. Wählen Sie das ID-Feld der Zertifizierungsstelle aus, die Sie überprüfen möchten, um den Bereich Allgemein zu öffnen. Der eindeutige 32-Byte-Hexadezimalbezeichner der CA wird oben angezeigt. Der Bereich enthält die folgenden zusätzlichen Informationen:

   - Status — CA-Status. Mögliche Werte sind Erstellt, Ausstehendes Zertifikat, Aktiv, Gelöscht, Deaktiviert, Abgelaufen und Fehlgeschlagen.
   - ARN — Der Amazon-Ressourcenname für die CA.
   - Besitzer — Das AWS Konto, dem die CA gehört. Dies kann Ihr Konto (Self) oder ein Konto sein, für das Verwaltungsberechtigungen für die Zertifizierungsstelle an Sie delegiert wurden.
   - CA-Typ — Der Typ der CA. Mögliche Werte sind Root und Subordinate.
   - Erstellt am — Datum und Uhrzeit der Erstellung der CA.
   - Ablaufdatum — Datum und Uhrzeit, an dem das CA-Zertifikat abläuft.
   - Modus — Der Modus der CA. Mögliche Werte sind General-Purpose (Zertifikate, die mit einem beliebigen Ablaufdatum konfiguriert werden können) und Kurzlebiges Zertifikat (Zertifikate mit einer maximalen Gültigkeitsdauer von sieben Tagen). Eine kurze Gültigkeitsdauer kann in einigen Fällen einen Sperrmechanismus ersetzen. Die Standardeinstellung ist Allzweck.

- Schlüsselalgorithmus — Der von der CA unterstützte Public-Key-Algorithmus. Mögliche Werte sind ML-DSA-44, ML-DSA-65, ML-DSA-87, RSA 2048, RSA 3072, RSA 4096, ECDSA P256, ECDSA P384 und ECDSA P521.
- Signaturalgorithmus — Der Algorithmus, den die CA verwendet, um ihre eigenen Certificate Signing Request und OCSP-Antworten zu signieren (nicht zu verwechseln mit dem in der API verwendeten Parameter). CRLs `SigningAlgorithm` IssueCertificate Mögliche Werte sind ML-DSA-44, ML-DSA-65, ML-DSA-87, RSA, RSA und SHA256 RSA, ECDSA, ECDSA, ECDSA SHA384 SHA512 SHA256 SHA384 SHA512
- Wichtiger Standard für Speichersicherheit — Grad der Konformität mit den Federal Information Processing Standards (FIPS). Sie können aus folgenden Werten wählen: FIPS 140-2 Level 2 oder höher, FIPS 140-2 Level 3 oder höher und CCPC Level 1 oder höher. AWS Dieser Parameter variiert je nach Region.

  > **Note**
  >
  > Ab dem 26. Januar 2023 schützt AWS Private CA alle privaten CA-Schlüssel in Regionen außerhalb China mithilfe von Hardware-Sicherheitsmodulen (HSMs), die FIPS PUB 140-2 Level 3 entsprechen.

Um CA-Details anzuzeigen und zu ändern, verwenden Sie AWS CLI

Verwenden Sie den describe-certificate-authorityBefehl in AWS CLI , um Details zu einer Zertifizierungsstelle anzuzeigen, wie im folgenden Befehl dargestellt:

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

Die vom System zurückgegebenen Informationen ähneln den Folgenden:

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

Hinweise zum Aktualisieren einer privaten Zertifizierungsstelle über die Befehlszeile finden Sie unterAktualisierung einer CA (CLI).

Fügen Sie Tags für Ihre private CA hinzu

Tags sind Wörter oder Ausdrücke, die in Form von Metadaten zum Identifizieren und Organisieren von AWS -Ressourcen verwendet werden. Jedes Tag besteht aus einem Schlüssel und einem Wert. Sie können die AWS Private CA Konsole AWS Command Line Interface (AWS CLI) oder die PCA-API verwenden, um private CAs Tags hinzuzufügen, anzuzeigen oder zu entfernen.

Sie können jederzeit benutzerdefinierte Tags für Ihre private CA hinzufügen oder entfernen. Sie könnten beispielsweise private CAs mit Schlüssel-Wert-Paaren wie `Environment=Prod` oder kennzeichnen, `Environment=Beta` um zu identifizieren, für welche Umgebung die CA bestimmt ist. Weitere Informationen finden Sie unter Private CA erstellen.

> **Note**
>
> Um einer privaten CA während des Erstellungsvorgangs Tags hinzuzufügen, muss ein CA-Administrator der `CreateCertificateAuthority` Aktion zunächst eine Inline-IAM-Richtlinie zuordnen und das Tagging explizit zulassen. Weitere Informationen finden Sie unter Tag-on-create: Hinzufügen von Tags an eine CA zum Zeitpunkt der Erstellung.

Tagging wird auch von anderen AWS Ressourcen unterstützt. Sie können verschiedenen Ressourcen dasselbe Tag zuweisen, um anzuzeigen, dass diese Ressourcen miteinander verknüpft sind. Sie können beispielsweise Ihrer CA, dem Elastic Load Balancing Load Balancer und anderen verwandten Ressourcen ein Tag zuweisen. `Website=example.com` Weitere Informationen zum Markieren von AWS Ressourcen finden Sie unter Taggen Ihrer EC2 Amazon-Ressourcen im EC2 Amazon-Benutzerhandbuch.

Für Tags gelten die folgenden grundlegenden Einschränkungen: AWS Private CA

- Die maximale Anzahl von Tags für jede private CA ist 50.
- Die maximale Länge eines Tag-Schlüssels beträgt 128 Zeichen.
- Die maximale Länge eines Tag-Wertes beträgt 256 Zeichen.
- Der Tag-Schlüssel und -Wert können die folgenden Zeichen enthalten: A-Z, a-z und .:+= @_%- (Bindestrich).
- Bei Tag-Schlüsseln und -Werten wird zwischen Groß- und Kleinschreibung unterschieden.
- Die Präfixe `aws:` und `rds:` sind für die Verwendung von AWS reserviert. Sie können keine Tags hinzufügen, bearbeiten oder löschen, deren Schlüssel mit `aws:` oder `rds:` beginnt. Standard-Tags, die mit Ihrem Kontingent beginnen `aws:` und `rds:` nicht auf Ihr tags-per-resource Kontingent angerechnet werden.
- Wenn Sie planen, Ihr Tagging-Schema für mehrere Dienste und Ressourcen zu verwenden, denken Sie daran, dass andere Dienste möglicherweise andere Einschränkungen für zulässige Zeichen haben. Weitere Informationen finden Sie in der Dokumentation des jeweiligen Services.
- AWS Private CA Tags sind nicht für die Verwendung in den Resource Groups und im Tag-Editor in verfügbar AWS-Managementkonsole.

Sie können eine private CA aus der AWS Private CA -Konsole, der AWS Command Line Interface (AWS CLI) oder der AWS Private CA -API mit einem Tag versehen.

So versehen Sie eine private CA mit Tags (Konsole)

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause.
2. Wählen Sie auf der Seite Private Zertifizierungsstellen Ihre private Zertifizierungsstelle aus der Liste aus.
3. Wählen Sie im Detailbereich unter der Liste die Registerkarte Tags aus. Eine Liste der vorhandenen Tags wird angezeigt.

4. Wählen Sie Tags verwalten aus.
5. Wählen Sie Neues Tag hinzufügen aus.
6. Geben Sie einen Schlüssel und ein Wertpaar ein.
7. Wählen Sie Speichern.

So versehen Sie eine private CA mit Tags (AWS CLI)

Verwenden Sie den tag-certificate-authorityBefehl, um Ihrer privaten CA Tags hinzuzufügen.

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

Verwenden Sie den Befehl list-tags, um die Tags für eine private CA aufzulisten.

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

Verwenden Sie den untag-certificate-authorityBefehl, um Tags aus einer privaten CA zu entfernen.

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

Verstehen Sie den AWS Private CA CA-Status

Der Status einer Zertifizierungsstelle, die anhand der AWS Private CA Ergebnisse einer Benutzeraktion oder in einigen Fällen einer Serviceaktion verwaltet wird. Beispielsweise ändert sich der Status einer Zertifizierungsstelle, wenn sie abläuft. Die Statusoptionen, die CA-Administratoren zur Verfügung stehen, hängen vom aktuellen Status der CA ab.

AWS Private CA kann die folgenden Statuswerte melden. Die Tabelle zeigt die CA-Funktionen, die in den einzelnen Bundesstaaten verfügbar sind.

> **Note**
>
> Für alle Statuswerte außer `DELETED` und `FAILED` wird Ihnen die CA in Rechnung gestellt.

Status	Zertifikate ausstellen	Bestätigen Sie Zertifikate mit OCSP	Generieren CRLs	Audits generieren	Sie können das CA-Zertifikat aktualisieren	Zertifikate können widerrufen werden	Ihnen wird die CA in Rechnung gestellt
`CREATING`— Die CA wird gerade erstellt.	Nein	Nein	Nein	Nein	Nein	Nein	Ja
`PENDING_CERTIFICATE` — Die CA wurde erstellt und benötigt ein Zertifikat, um betriebsbereit zu sein. *	Nein	Nein	Nein	Nein	Nein	Nein	Ja
`ACTIVE`	Ja	Ja	Ja	Ja	Ja	Ja	Ja
`DISABLED`— Sie haben die CA manuell deaktiviert.	Nein	Ja	Ja	Ja	Nein	Ja	Ja
`EXPIRED`— Das CA-Zertifikat ist abgelaufen. **	Nein	Nein	Nein	Nein	Ja	Nein	Ja
`FAILED`	Die `CreateCertificateAuthority` Aktion ist fehlgeschlagen. Dies kann aufgrund eines Netzwerkausfalls, eines AWS Back-End-Fehlers oder anderer Fehler auftreten. Eine fehlgeschlagene CA kann nicht wiederhergestellt werden. Löschen Sie die CA und erstellen Sie eine neue.						Nein

Status	Zertifikate ausstellen	Bestätigen Sie Zertifikate mit OCSP	Generieren CRLs	Audits generieren	Sie können das CA-Zertifikat aktualisieren	Zertifikate können widerrufen werden	Ihnen wird die CA in Rechnung gestellt
`DELETED`	Ihre CA befindet sich innerhalb des Wiederherstellungszeitraums, der zwischen 7 und 30 Tagen dauern kann. Nach diesem Zeitraum wird sie endgültig gelöscht. • Wenn Sie die `RestoreCertificateAuthority` -API in einer CA mit dem `DELETED`-Status und einem abgelaufenen Zertifikat aufrufen, wird die CA auf `EXPIRED` gesetzt. • Weitere Informationen zum Löschen einer CA finden Sie unter Löschen Ihrer privaten CA.						Nein

Um die Aktivierung abzuschließen, müssen Sie eine CSR generieren, ein signiertes CA-Zertifikat von einer Zertifizierungsstelle abrufen und das Zertifikat in diese importieren. AWS Private CA Die CSR kann entweder an Ihre neue Zertifizierungsstelle (zur Selbstsignierung) oder an eine lokale Stammzertifizierungsstelle oder untergeordnete Zertifizierungsstelle übermittelt werden. Weitere Informationen finden Sie unter Installation des CA-Zertifikats.

Sie können den Status einer abgelaufenen CA nicht direkt ändern. Wenn Sie ein neues Zertifikat für die Zertifizierungsstelle importieren, wird der Status auf AWS Private CA zurückgesetzt, es `ACTIVE` sei denn, er wurde vor Ablauf des Zertifikats auf `DISABLED` festgelegt.

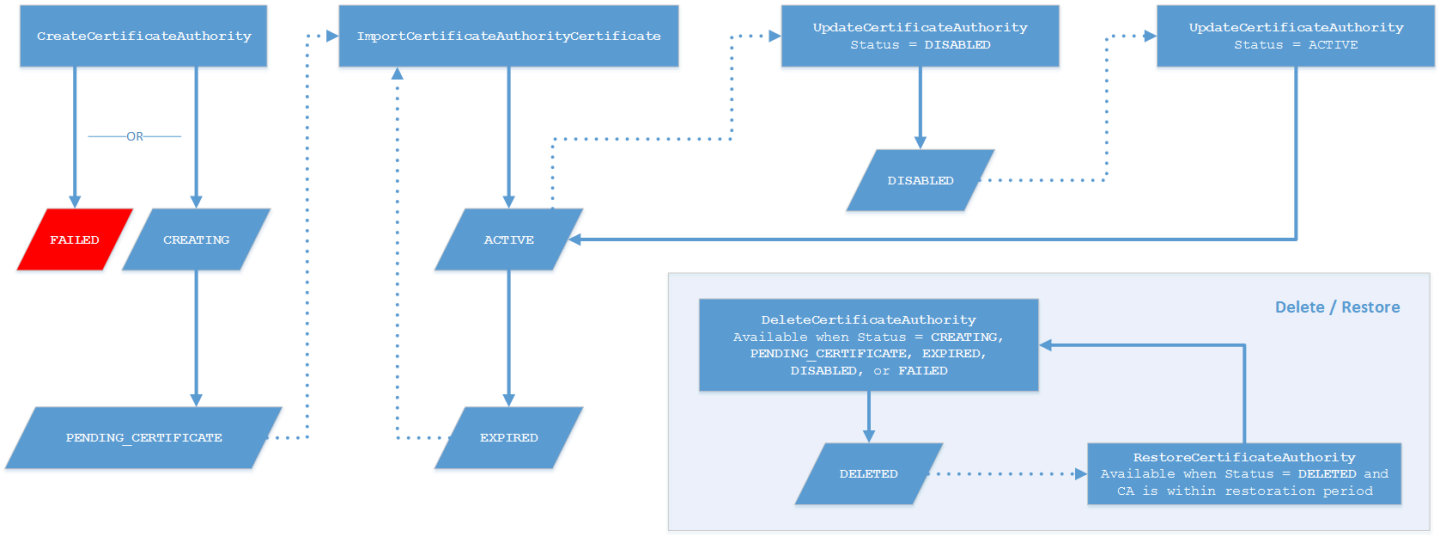
Zusätzliche Überlegungen zu abgelaufenen CA-Zertifikaten:

- CA-Zertifikate werden nicht automatisch erneuert. Informationen zur Automatisierung der Verlängerung durch finden Sie AWS Certificate Manager unterWeisen Sie ACM Berechtigungen zur Zertifikatsverlängerung zu.
- Wenn Sie versuchen, ein neues Zertifikat mit einer abgelaufenen CA auszustellen, wird die `IssueCertificate`-API `InvalidStateException` zurückgegeben. Eine abgelaufene Stamm-CA muss ein neues Stamm-CA-Zertifikat selbst signieren, bevor sie neue untergeordnete Zertifikate ausstellen kann.

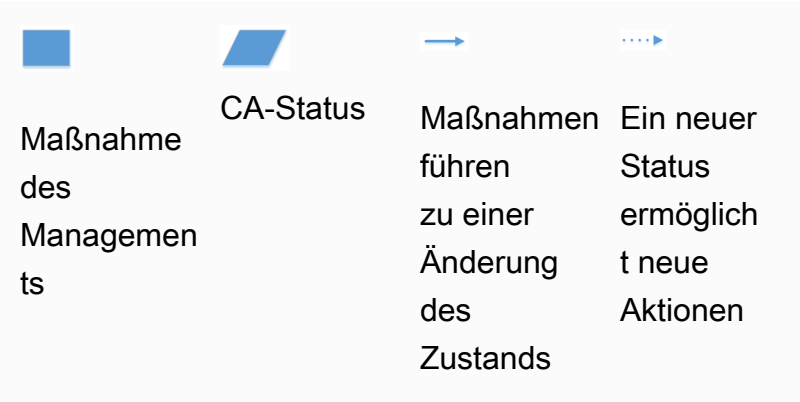
- The `ListCertificateAuthorities`und `DescribeCertificateAuthority` APIs gibt den Status zurück, `EXPIRED` ob das CA-Zertifikat abgelaufen ist, unabhängig davon, ob der CA-Status auf `ACTIVE` oder `DISABLED` gesetzt ist. Wenn jedoch die abgelaufene CA auf `DELETED` festgelegt wurde, ist der zurückgegebene Status `DELETED`.
- Die `UpdateCertificateAuthority`-API kann den Status einer abgelaufenen CA nicht aktualisieren.
- Die `RevokeCertificate` API kann nicht verwendet werden, um abgelaufene Zertifikate, einschließlich eines CA-Zertifikats, zu widerrufen.

Beziehung zwischen CA-Status und CA-Lebenszyklus

Das folgende Diagramm veranschaulicht den Lebenszyklus der CA als Interaktion von Verwaltungsaktionen mit dem Status der CA.



Schlüssel des Diagramms



Oben im Diagramm werden Verwaltungsaktionen über die AWS Private CA -Konsole, die CLI oder die API angewendet. Die Aktionen führen die CA durch Erstellung, Aktivierung, Ablauf und Erneuerung. Der Status der CA ändert sich in Reaktion auf manuelle Aktionen oder automatisierte Aktualisierungen (wie anhand der durchgezogenen Linien angezeigt). In den meisten Fällen führt ein neuer Status zu einer neuen möglichen Aktion (dargestellt durch eine gepunktete Linie), die der CA-Administrator anwenden kann. Im Kasten unten rechts sehen Sie die möglichen Statuswerte, die Lösch- und Wiederherstellungsaktionen ermöglichen.

Aktualisieren Sie eine private Zertifizierungsstelle in AWS Private Certificate Authority

Sie können den Status einer privaten Zertifizierungsstelle aktualisieren oder ihre Sperrkonfiguration ändern, nachdem Sie sie erstellt haben. Dieses Thema enthält Details zum CA-Status und zum CA-Lebenszyklus sowie Beispiele für Konsolen- und CLI-Updates für CAs.

Aktualisieren Sie eine CA (Konsole)

Die folgenden Verfahren zeigen, wie Sie vorhandene CA-Konfigurationen mithilfe von aktualisieren AWS-Managementkonsole.

CA-Status aktualisieren (Konsole)

In diesem Beispiel wird der Status einer aktivierten Zertifizierungsstelle auf deaktiviert geändert.

Um den Status einer CA zu aktualisieren

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause
2. Wählen Sie auf der Seite Private Zertifizierungsstellen eine private Zertifizierungsstelle aus der Liste aus, die derzeit aktiv ist.
3. Wählen Sie im Menü Aktionen die Option Deaktivieren aus, um die private CA zu deaktivieren.

Aktualisierung der Sperrkonfiguration einer CA (Konsole)

Sie können die Sperrkonfiguration für Ihre private Zertifizierungsstelle aktualisieren, indem Sie beispielsweise entweder OCSP- oder CRL-Unterstützung hinzufügen oder entfernen oder deren Einstellungen ändern.

> **Note**
>
> Änderungen an der Sperrkonfiguration einer Zertifizierungsstelle wirken sich nicht auf Zertifikate aus, die bereits ausgestellt wurden. Damit der verwaltete Widerruf funktioniert, müssen ältere Zertifikate erneut ausgestellt werden.

Für OCSP ändern Sie die folgenden Einstellungen:

- Aktivieren oder deaktivieren Sie OCSP.
- Aktivieren oder deaktivieren Sie einen benutzerdefinierten vollqualifizierten OCSP-Domänennamen (FQDN).
- Ändern Sie den FQDN.

Für eine CRL können Sie jede der folgenden Einstellungen ändern:

- Der CRL-Typ (vollständig oder partitioniert)
- Gibt an, ob die private CA eine Zertifikatsperrliste (CRL) generiert
- Die Anzahl der Tage, bevor eine Zertifikatsperrliste abläuft. Beachten Sie, dass der Versuch, die CRL zu regenerieren, bei der Hälfte der von Ihnen angegebenen Anzahl von Tagen AWS Private CA beginnt.
- Der Name des Amazon S3 S3-Buckets, in dem Ihre CRL gespeichert ist.
- Ein Alias, um den Namen Ihres Amazon S3 S3-Buckets vor der Öffentlichkeit zu verbergen.

> **Important**
>
> Die Änderung eines der oben genannten Parameter kann negative Auswirkungen haben. Beispiele hierfür sind das Deaktivieren der CRL-Generierung, das Ändern des Gültigkeitszeitraums oder das Ändern des S3-Buckets, nachdem Sie Ihre private CA in Betrieb genommen haben. Solche Änderungen können bestehende Zertifikate beschädigen, die von der CRL und der aktuellen CRL-Konfiguration abhängen. Das Ändern des Alias kann sicher durchgeführt werden, solange der alte Alias mit dem richtigen Bucket verknüpft bleibt.

Um die Sperreinstellungen zu aktualisieren

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause.
2. Wählen Sie auf der Seite Private Zertifizierungsstellen eine private Zertifizierungsstelle aus der Liste aus. Dadurch wird das Detailfenster für die CA geöffnet.
3. Wählen Sie die Registerkarte Widerrufskonfiguration und dann Bearbeiten aus.
4. Unter Optionen zum Widerruf von Zertifikaten werden zwei Optionen angezeigt:

   - Aktivieren Sie die CRL-Verteilung
   - Schalten Sie OCSP ein

   Sie können einen, keinen oder beide dieser Sperrmechanismen für Ihre CA konfigurieren. Der verwaltete Widerruf ist zwar optional, wird aber als bewährte Methode empfohlen. Bevor Sie diesen Schritt abschließen, finden Sie unter Planen Sie Ihre Methode zum Widerruf von AWS Private CA Zertifikaten Informationen zu den Vorteilen der einzelnen Methoden, zur eventuell erforderlichen vorläufigen Einrichtung und zu zusätzlichen Sperrfunktionen.

Um eine CRL zu konfigurieren

1. Wählen Sie CRL-Verteilung aktivieren aus.
2. Um einen Amazon S3 S3-Bucket für Ihre CRL-Einträge zu erstellen, wählen Sie Neuen S3-Bucket erstellen aus. Geben Sie einen eindeutigen Bucket-Namen an. (Sie müssen den Pfad zum Bucket nicht einschließen.) Andernfalls lassen Sie diese Option deaktiviert und wählen Sie einen vorhandenen Bucket aus der Liste der S3-Bucket-Namen aus.

   Wenn Sie einen neuen Bucket erstellen, AWS Private CA erstellt und fügt ihm die erforderliche Zugriffsrichtlinie hinzu. Wenn Sie sich dafür entscheiden, einen vorhandenen Bucket zu verwenden, müssen Sie ihm eine Zugriffsrichtlinie anhängen, bevor Sie mit der Generierung CRLs beginnen können. Verwenden Sie eines der unter beschriebenen RichtlinienmusterZugriffsrichtlinien für CRLs in Amazon S3. Informationen zum Anhängen einer Richtlinie finden Sie unter Hinzufügen einer Bucket-Richtlinie mithilfe der Amazon S3 S3-Konsole.

> **Note**
>
> Wenn Sie die AWS Private CA Konsole verwenden, schlägt der Versuch, eine CA zu erstellen, fehl, wenn die beiden folgenden Bedingungen zutreffen:
>
> - Sie setzen die Einstellungen für den Block Public Access in Ihrem Amazon S3 S3-Bucket oder Konto durch.
> - Sie haben darum AWS Private CA gebeten, automatisch einen Amazon S3 S3-Bucket zu erstellen.
>
> In dieser Situation versucht die Konsole standardmäßig, einen öffentlich zugänglichen Bucket zu erstellen, und Amazon S3 lehnt diese Aktion ab. Überprüfen Sie in diesem Fall Ihre Amazon S3 S3-Einstellungen. Weitere Informationen finden Sie unter Sperren des öffentlichen Zugriffs auf Ihren Amazon S3 S3-Speicher.

3. Erweitern Sie Erweitert, um weitere Konfigurationsoptionen zu erhalten.

   - Wählen Sie Partitionierung aktivieren, um die Partitionierung von zu aktivieren. CRLs Wenn Sie die Partitionierung nicht aktivieren, gilt für Ihre CA die maximale Anzahl widerrufener Zertifikate, die in den Kontingenten angegeben ist.AWS Private Certificate Authority Weitere Informationen zu partitionierten Dateien finden Sie unter CRLs CRL-Typen.
   - Fügen Sie einen benutzerdefinierten CRL-Namen hinzu, um einen Alias für Ihren Amazon S3 S3-Bucket zu erstellen. Dieser Name ist in Zertifikaten enthalten, die von der Zertifizierungsstelle in der Erweiterung „CRL Distribution Points“ ausgestellt wurden, die in RFC 5280 definiert ist. Um CRLs over zu verwenden IPv6, setzen Sie dies auf den DualStack-S3-Endpunkt Ihres Buckets, wie unter Using over beschrieben. CRLs IPv6
   - Fügen Sie einen benutzerdefinierten Pfad hinzu, um einen DNS-Alias für den Dateipfad in Ihrem Amazon S3 S3-Bucket zu erstellen.
   - Geben Sie die Gültigkeitsdauer in Tagen ein, in der Ihre CRL gültig bleiben soll. Der Standardwert lautet 7 Tage. Im Online-Bereich CRLs ist eine Gültigkeitsdauer von 2-7 Tagen üblich. AWS Private CA versucht, die CRL in der Mitte des angegebenen Zeitraums zu regenerieren.

4. Wählen Sie Änderungen speichern, wenn Sie fertig sind.

Um OCSP zu konfigurieren

1. Wählen Sie auf der Seite Certificate Revocation die Option Turn on OCSP aus.
2. (Optional) Geben Sie im Feld Benutzerdefinierter OCSP-Endpunkt einen vollqualifizierten Domänennamen (FQDN) für Ihren OCSP-Endpunkt ein. Um OCSP over zu verwenden IPv6, legen Sie dieses Feld auf einen Dual-Stack-Endpunkt fest, wie unter OCSP Over verwenden beschrieben. IPv6

   Wenn Sie in diesem Feld einen FQDN angeben, wird der FQDN anstelle der Standard-URL für den OCSP-Responder in die Authority Information Access-Erweiterung jedes ausgestellten Zertifikats AWS Private CA eingefügt. AWS Wenn ein Endpunkt ein Zertifikat empfängt, das den benutzerdefinierten FQDN enthält, fragt er diese Adresse nach einer OCSP-Antwort ab. Damit dieser Mechanismus funktioniert, müssen Sie zwei zusätzliche Maßnahmen ergreifen:

   - Verwenden Sie einen Proxyserver, um den Datenverkehr, der bei Ihrem benutzerdefinierten FQDN eingeht, an den AWS OCSP-Responder weiterzuleiten.
   - Fügen Sie Ihrer DNS-Datenbank einen entsprechenden CNAME-Eintrag hinzu.

   > **Tip**
   >
   > Weitere Hinweise zur Implementierung einer vollständigen OCSP-Lösung mit einem benutzerdefinierten CNAME finden Sie unter. Passen Sie die OCSP-URL an für AWS Private CA

   Hier ist zum Beispiel ein CNAME-Eintrag für benutzerdefiniertes OCSP, wie er in Amazon Route 53 erscheinen würde.

Datensatzname	Typ	Routing-Richtlinie	Unterscheidungsmerkmal	Bewerten/Weiterleiten des Datenverkehrs an
alternative.example.com	CNAME	Einfach	-	proxy.example.com

> **Note**
>
> Der Wert des CNAME-Werts darf kein Protokollpräfix wie „http://“ oder „https://“ enthalten.

3. Wählen Sie Änderungen speichern, wenn Sie fertig sind.

Aktualisierung einer CA (CLI)

Die folgenden Verfahren zeigen, wie Sie den Status und die Sperrkonfiguration einer vorhandenen Zertifizierungsstelle mithilfe der aktualisieren AWS CLI.

> **Note**
>
> Änderungen an der Sperrkonfiguration einer Zertifizierungsstelle wirken sich nicht auf Zertifikate aus, die bereits ausgestellt wurden. Damit der verwaltete Widerruf funktioniert, müssen ältere Zertifikate erneut ausgestellt werden.

Um den Status Ihrer privaten CA ()AWS CLI zu aktualisieren

Verwenden Sie den Befehl update-certificate-authority.

Dies ist nützlich, wenn Sie bereits über eine Zertifizierungsstelle verfügen`DISABLED`, deren Status Sie ändern möchten`ACTIVE`. Bestätigen Sie zunächst den Anfangsstatus der CA mit dem folgenden Befehl.

```
$ 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
```

Dies führt zu einer Ausgabe, die der folgenden ähnelt.

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

```
        "CreatedAt": "2021-03-05T14:24:12.867000-08:00",
        "LastStateChangeAt": "2021-03-08T13:17:40.221000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "DISABLED",
        "NotBefore": "2021-03-08T07:46:27-08:00",
        "NotAfter": "2022-03-08T08:46:27-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "CustomCname": "alternative.example.com",
                "S3BucketName": "amzn-s3-demo-bucket"
   },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

Der folgende Befehl setzt den Status der privaten CA auf`ACTIVE`. Dies ist nur möglich, wenn ein gültiges Zertifikat auf der CA installiert ist.

```
$ 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"
```

Überprüfen Sie den neuen Status der CA.

```
$ aws acm-pca describe-certificate-authority \
      --certificate-authority-arn "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566" \
      --output json
```

Der Status wird jetzt als angezeigt`ACTIVE`.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-05T14:24:12.867000-08:00",
        "LastStateChangeAt": "2021-03-08T13:23:09.352000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T07:46:27-08:00",
        "NotAfter": "2022-03-08T08:46:27-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "CustomCname": "alternative.example.com",
                "S3BucketName": "amzn-s3-demo-bucket"
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
```

```
}
```

In einigen Fällen haben Sie möglicherweise eine aktive Zertifizierungsstelle, für die kein Sperrmechanismus konfiguriert ist. Wenn Sie mit der Verwendung einer Zertifikatssperrliste (CRL) beginnen möchten, gehen Sie wie folgt vor.

So fügen Sie einer vorhandenen Zertifizierungsstelle eine CRL hinzu ()AWS CLI

1. Verwenden Sie den folgenden Befehl, um den aktuellen Status der CA zu überprüfen.

   ```
   $ 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
   ```

   Die Ausgabe bestätigt, dass die CA zwar den Status hatACTIVE, aber nicht für die Verwendung einer CRL konfiguriert ist.

   ```
   {
       "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. Erstellen und speichern Sie eine Datei mit einem Namen, `revoke_config.txt` um beispielsweise Ihre CRL-Konfigurationsparameter zu definieren.

```
{
   "CrlConfiguration":{
      "Enabled": true,
      "ExpirationInDays": 7,
      "S3BucketName": "amzn-s3-demo-bucket"
   }
}
```

> **Note**
>
> Wenn Sie eine Matter Device Attestation CA aktualisieren, um sie zu aktivieren CRLs, müssen Sie sie so konfigurieren, dass die CDP-Erweiterung in den ausgestellten Zertifikaten weggelassen wird, um dem aktuellen Matter-Standard zu entsprechen. Definieren Sie dazu Ihre CRL-Konfigurationsparameter wie unten dargestellt:
>
> ```
> {
>    "CrlConfiguration":{
>       "Enabled": true,
>       "ExpirationInDays": 7,
>       "S3BucketName": "amzn-s3-demo-bucket"
>       "CrlDistributionPointExtensionConfiguration":{
>          "OmitExtension": true
>       }
>    }
> }
> ```

3. Verwenden Sie den update-certificate-authorityBefehl und die Sperrkonfigurationsdatei, um die CA zu aktualisieren.

```
$ 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. Überprüfen Sie erneut den Status der CA.

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

Die Ausgabe bestätigt, dass CA jetzt für die Verwendung einer CRL konfiguriert ist.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-08T14:36:26.449000-08:00",
        "LastStateChangeAt": "2021-03-08T14:50:52.224000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T13:46:50-08:00",
        "NotAfter": "2022-03-08T14:46:50-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "S3BucketName": "amzn-s3-demo-bucket",
```

```
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

In einigen Fällen möchten Sie möglicherweise OCSP-Sperrunterstützung hinzufügen, anstatt wie im vorherigen Verfahren eine CRL zu aktivieren. Gehen Sie in diesem Fall wie folgt vor.

Um OCSP-Unterstützung zu einer vorhandenen CA hinzuzufügen ()AWS CLI

1. Erstellen und speichern Sie eine Datei mit einem Namen, `revoke_config.txt` um beispielsweise Ihre OCSP-Parameter zu definieren.

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

2. Verwenden Sie den update-certificate-authorityBefehl und die Sperrkonfigurationsdatei, um die CA zu aktualisieren.

```
$ 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. Überprüfen Sie erneut den Status der CA.

```
$ aws acm-pca describe-certificate-authority
 --certificate-authority-arnarn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
 --output json
```

Die Ausgabe bestätigt, dass CA jetzt für die Verwendung von OCSP konfiguriert ist.

```
{
    "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**
>
> Sie können auch sowohl die CRL- als auch die OCSP-Unterstützung auf einer CA konfigurieren.

Löschen Ihrer privaten CA

Sie können eine private Zertifizierungsstelle dauerhaft aus der AWS-Managementkonsole oder AWS CLI löschen. Möglicherweise möchten Sie eine CA löschen, um sie mit einer neuen CA zu ersetzen, die über einen neuen privaten Schlüssel verfügt. Gehen Sie folgendermaßen vor, um eine CA sicher zu löschen:

1. Erstellen Sie die Ersatz-CA.
2. Sobald die neue private CA produktiv ist, deaktivieren Sie die alte, ohne sie jedoch sofort zu löschen.
3. Behalten Sie die alte CA in deaktiviertem Zustand, bis alle Zertifikate abgelaufen sind, die von dieser CA ausgestellt wurden.
4. Löschen Sie die alte CA.

AWS Private CA überprüft nicht, ob alle ausgestellten Zertifikate abgelaufen sind, bevor es eine Löschanforderung verarbeitet. Sie können einen Auditbericht generieren, um zu ermitteln, welche Zertifikate abgelaufen sind. Während die CA deaktiviert ist, können Sie zwar Zertifikate sperren, jedoch keine neuen ausstellen.

Wenn Sie eine private CA löschen müssen, bevor alle von ihr ausgestellten Zertifikate abgelaufen sind, empfehlen wir, dass Sie auch das CA-Zertifikat widerrufen. Das CA-Zertifikat wird in der Zertifikatsperrliste der übergeordneten CA aufgeführt und die private CA wird von den Clients als nicht vertrauenswürdig eingestuft.

Important

Eine private CA kann gelöscht werden, wenn sie sich im Status `PENDING_CERTIFICATE`, `CREATING`, `EXPIRED`, `DISABLED` oder `FAILED` befindet. Vor dem Löschen einer CA im Status `ACTIVE` müssen Sie sie deaktivieren. Andernfalls resultiert die Löschanfrage in einer Ausnahme. Wenn Sie eine private Zertifizierungsstelle im `DISABLED` Bundesstaat `PENDING_CERTIFICATE` oder löschen, können Sie die Dauer der Wiederherstellung auf 7 bis 30 Tage festlegen, wobei 30 die Standardeinstellung ist. Während dieses Zeitraums wird der Status auf `DELETED` gesetzt und die CA kann wiederhergestellt werden. Einer privaten Zertifizierungsstelle, die gelöscht wird, während sie sich im `FAILED` Status `CREATING` oder befindet, ist kein Wiederherstellungszeitraum zugewiesen und sie kann nicht wiederhergestellt werden. Weitere Informationen finden Sie unter Stellen Sie eine private CA wieder her.

Für eine private CA wird nichts mehr berechnet, nachdem sie gelöscht wurde. Wenn jedoch eine gelöschte CA wiederhergestellt wurde, bezahlen Sie für die Zeit zwischen Löschung und Wiederherstellung. Weitere Informationen finden Sie unter Preisgestaltung für AWS Private Certificate Authority.

So löschen Sie eine private CA (Konsole)

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause.
2. Wählen Sie auf der Seite Private Zertifizierungsstellen Ihre private Zertifizierungsstelle aus der Liste aus.
3. Wenn sich Ihre Zertifizierungsstelle in diesem `ACTIVE` Bundesstaat befindet, müssen Sie sie zuerst deaktivieren. Wählen Sie im Menü Aktionen die Option Deaktivieren. Wenn Sie dazu aufgefordert werden, wählen Sie Ich verstehe das Risiko, weiter.
4. Wählen Sie für eine CA, die sich nicht im `ACTIVE` Status befindet, Actions, Delete aus.
5. Wenn sich Ihre CA im `PENDING_CERTIFICATE` Status`DISABLED`,, oder befindet`EXPIRED`, können Sie auf der Seite „CA löschen“ einen Wiederherstellungszeitraum von 7 bis 30 Tagen angeben. Wenn sich Ihre private CA nicht in einem dieser Zustände befindet, kann sie später nicht wiederhergestellt werden und das Löschen ist dauerhaft.
6. Wählen Sie Löschen aus.
7. Wenn Sie sicher sind, dass Sie die private Zertifizierungsstelle löschen möchten, wählen Sie bei Aufforderung Permanently delete (Dauerhaft löschen) aus. Der Status der privaten Zertifizierungsstelle ändert sich in `DELETED`. Sie können die private Zertifizierungsstelle jedoch vor dem Ende des Wiederherstellungszeitraums wiederherstellen. Rufen Sie den ListCertificateAuthoritiesAPI-Vorgang DescribeCerticateAuthorityoder auf, um den Wiederherstellungszeitraum einer privaten Zertifizierungsstelle in diesem `DELETED` Bundesstaat zu überprüfen.

So löschen Sie eine private CA (AWS CLI)

Verwenden Sie den delete-certificate-authorityBefehl, um eine private CA zu löschen.

```
$ 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
```

Stellen Sie eine private CA wieder her

Sie können eine gelöschte private Zertifizierungsstelle wiederherstellen, solange sich die Zertifizierungsstelle innerhalb des Wiederherstellungszeitraums befindet, den Sie beim Löschen angegeben haben. Die Wiederherstellungszeit beträgt 7 bis 30 Tage. Nach Ablauf dieses Zeitraums wird die private Zertifizierungsstelle dauerhaft gelöscht. Weitere Informationen finden Sie unter Löschen Ihrer privaten CA. Eine private Zertifizierungsstelle, die dauerhaft gelöscht wurde, kann nicht mehr wiederhergestellt werden.

> **Note**
>
> Für eine private CA wird nichts mehr berechnet, nachdem sie gelöscht wurde. Wenn jedoch eine gelöschte CA wiederhergestellt wurde, bezahlen Sie für die Zeit zwischen Löschung und Wiederherstellung. Weitere Informationen finden Sie unter Preisgestaltung für AWS Private Certificate Authority.

Wiederherstellung einer privaten CA (Konsole)

Sie können die verwenden AWS-Managementkonsole , um eine private CA wiederherzustellen.

So stellen Sie eine private Zertifizierungsstelle wieder her (Konsole)

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause.
2. Wählen Sie auf der Seite Private Zertifizierungsstellen Ihre gelöschte private Zertifizierungsstelle aus der Liste aus.
3. Wählen Sie im Menü Aktionen die Option Restore (Wiederherstellen).
4. Wählen Sie auf der Seite Restore CA erneut Restore aus.
5. Ist der Vorgang erfolgreich, wird der Status der privaten Zertifizierungsstelle auf ihren Status vor der Löschung gesetzt. Wählen Sie „Aktionen“, „Aktivieren“ und erneut „Aktivieren“, um den Status zu ändern`ACTIVE`. Wenn sich die private CA zum Löschzeitpunkt im Status `PENDING_CERTIFICATE` befand, müssen Sie ein CA-Zertifikat in die private CA importieren, bevor Sie sie aktivieren können.

Stellen Sie eine private Zertifizierungsstelle wieder her (AWS CLI)

Verwenden Sie den restore-certificate-authorityBefehl, um eine gelöschte private Zertifizierungsstelle wiederherzustellen, die sich im `DELETED` Status befindet. Die folgenden Schritte beschreiben den gesamten Prozess, der erforderlich ist, um eine private Zertifizierungsstelle zu löschen, wiederherzustellen und erneut zu aktivieren.

Löschen, Wiederherstellen und Reaktivieren einer privaten CA (AWS CLI)

1. Löschen Sie die private Zertifizierungsstelle.

   Führen Sie den delete-certificate-authorityBefehl aus, um die private CA zu löschen. Wenn der Status der privaten CA `DISABLED` oder ist`PENDING_CERTIFICATE`, können Sie den `--permanent-deletion-time-in-days` Parameter so einstellen, dass der Wiederherstellungszeitraum der privaten CA zwischen 7 und 30 Tagen liegt. Wenn Sie keinen Wiederherstellungszeitraum angeben, lautet der Standardwert 30 Tage. Ist der Vorgang erfolgreich, setzt dieser Befehl den Status der privaten Zertifizierungsstelle auf `DELETED`.

   > **Note**
   >
   > Um wiederhergestellt werden zu können, muss der Status der privaten Zertifizierungsstelle zum Löschzeitpunkt `DISABLED` oder `PENDING_CERTIFICATE` lauten.

   ```
   $ 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. Stellen Sie die private Zertifizierungsstelle wieder her.

   Führen Sie den restore-certificate-authorityBefehl aus, um die private CA wiederherzustellen. Sie müssen den Befehl ausführen, bevor der Wiederherstellungszeitraum, den Sie mit dem Befehl delete-certificate-authority festgelegt haben, abläuft. Ist der Vorgang erfolgreich, setzt der Befehl den Status der privaten Zertifizierungsstelle auf ihren Status vor der Löschung.

   ```
   $ aws acm-pca restore-certificate-authority \
   ```

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

3. Setzen Sie die private Zertifizierungsstelle auf `ACTIVE`.

   Führen Sie den update-certificate-authorityBefehl aus, um den Status der privaten CA in zu ändern`ACTIVE`.

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

Verwenden Sie extern signierte private CA-Zertifikate

Wenn es sich bei der Vertrauensbasis Ihrer privaten Zertifizierungsstellenhierarchie um eine Zertifizierungsstelle außerhalb von handeln muss AWS Private CA, können Sie Ihre eigene Stammzertifizierungsstelle erstellen und selbst signieren. Alternativ können Sie ein privates CA-Zertifikat erhalten, das von einer externen privaten CA signiert wird, die von Ihrer Organisation betrieben wird. Unabhängig von der Quelle können Sie diese extern abgerufene Zertifizierungsstelle verwenden, um ein privates Zertifikat einer untergeordneten Zertifizierungsstelle zu signieren, das AWS Private CA verwaltet wird.

> **Note**
>
> Verfahren zur Einrichtung oder Beschaffung eines externen Vertrauensdienstanbieters fallen nicht in den Anwendungsbereich dieses Handbuchs.

Wenn Sie eine externe übergeordnete Zertifizierungsstelle mit verwenden AWS Private CA , können Sie die Einschränkungen der Zertifizierungsstellennamen durchsetzen, wie sie im Abschnitt „Namensbeschränkungen“ von RFC 5280 definiert sind. Namenseinschränkungen bieten CA-Administratoren die Möglichkeit, die Namen von Antragstellern in Zertifikaten einzuschränken.

Wenn Sie beabsichtigen, ein privates Zertifikat einer untergeordneten Zertifizierungsstelle mit einer externen Zertifizierungsstelle zu signieren, müssen Sie drei Aufgaben erledigen, bevor Sie über eine funktionierende Zertifizierungsstelle verfügen: AWS Private CA

1. Generieren Sie eine Zertifikatsignieranforderung (CSR).

2. Senden Sie die CSR an Ihre externe Signaturbehörde und senden Sie sie mit einem signierten Zertifikat und einer Zertifikatskette zurück.
3. Installieren Sie ein signiertes Zertifikat in AWS Private CA.

In den folgenden Verfahren wird beschrieben, wie Sie diese Aufgaben entweder mit dem AWS-Managementkonsole oder dem ausführen AWS CLI.

So erhalten und installieren Sie ein extern signiertes CA-Zertifikat (Konsole)

1. (Optional) Wenn Sie sich noch nicht auf der Detailseite der Zertifizierungsstelle befinden, öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause. Wählen Sie auf der Seite Private Zertifizierungsstellen eine untergeordnete Zertifizierungsstelle mit dem Status Ausstehendes Zertifikat, Aktiv, Deaktiviert oder Abgelaufen aus.
2. Wählen Sie „Aktionen“, „CA-Zertifikat installieren“, um die Seite „Zertifikat der untergeordneten Zertifizierungsstelle installieren“ zu öffnen.
3. Wählen Sie auf der Seite Zertifikat der untergeordneten Zertifizierungsstelle installieren unter Typ der Zertifizierungsstelle auswählen die Option Externe private Zertifizierungsstelle aus.
4. Unter CSR für diese Zertifizierungsstelle zeigt die Konsole den Base64-codierten ASCII-Text der CSR an. Sie können den Text mit der Schaltfläche Kopieren kopieren oder CSR in eine Datei exportieren und lokal speichern wählen.

   > **Note**
   >
   > Das genaue Format des CSR-Textes muss beim Kopieren und Einfügen beibehalten werden.

5. Wenn Sie die Offline-Schritte zum Abrufen eines signierten Zertifikats von Ihrer externen Signaturstelle nicht sofort ausführen können, können Sie die Seite schließen und zu ihr zurückkehren, sobald Sie über ein signiertes Zertifikat und eine Zertifikatskette verfügen.

   Andernfalls, wenn Sie bereit sind, führen Sie einen der folgenden Schritte aus:

   - Fügen Sie den Base64-codierten ASCII-Text Ihrer Zertifizierungsstelle und Ihrer Zertifikatskette in die entsprechenden Textfelder ein.
   - Wählen Sie Hochladen, um den Zertifikatshauptteil und die Zertifikatskette aus lokalen Dateien in die entsprechenden Textfelder zu laden.

6. Wählen Sie Bestätigen und installieren.

Um ein extern signiertes CA-Zertifikat (CLI) zu erhalten und zu installieren

1. Verwenden Sie den get-certificate-authority-csrBefehl, um die Certificate Signing Request (CSR) für Ihre private CA abzurufen. Wenn Sie die CSR an Ihr Display senden möchten, verwenden Sie die `--output text` Option, um CR/LF Zeichen am Ende jeder Zeile zu entfernen. Um die Zertifikatsignierungsanforderung an eine Datei zu senden, verwenden Sie die Umleitungsoption (**>**) gefolgt von einem Dateinamen.

   ```
   $ 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
   ```

   Nachdem Sie eine CSR als lokale Datei gespeichert haben, können Sie sie mit dem folgenden OpenSSL-Befehl überprüfen:

   ```
   openssl req -in path_to_CSR_file -text -noout
   ```

   Dieser Befehl generiert eine Ausgabe ähnlich der Folgenden. Beachten Sie, dass die CA-Erweiterung TRUE ist, was anzeigt, dass die CSR für ein CA-Zertifikat gilt.

   ```
   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. Senden Sie die CSR an Ihre externe Signaturstelle und rufen Sie Dateien ab, die das Base64-PEM-codierte signierte Zertifikat und die Zertifikatskette enthalten.
3. Verwenden Sie den import-certificate-authority-certificateBefehl, um die private CA-Zertifikatsdatei und die Kettendatei in zu importieren. AWS Private CA

```
$ 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
```

Zertifikate ausstellen und verwalten in AWS Private CA

Nachdem Sie eine private Zertifizierungsstelle (CA) erstellt und aktiviert und den Zugriff darauf konfiguriert haben, können Sie oder Ihre autorisierten Benutzer Zertifikate ausstellen und verwalten. Wenn Sie noch keine AWS Identity and Access Management (IAM-) Richtlinien für die CA eingerichtet haben, erfahren Sie im Abschnitt Identity and Access Management dieses Handbuchs mehr über deren Konfiguration. Informationen zur Konfiguration des CA-Zugriffs in Szenarien mit einem oder mehreren Konten finden Sie unter. Steuern Sie den Zugriff auf die private CA

Themen

Stellen Sie private Endentitätszertifikate aus

Wenn eine private Zertifizierungsstelle eingerichtet ist, können Sie private Endzertifikate entweder von AWS Certificate Manager (ACM) oder anfordern. AWS Private CA Die Funktionen der beiden Dienste werden in der folgenden Tabelle verglichen.

Funktion	ACM	AWS Private CA
Stellen Sie Zertifikate für Endunternehmen aus	✓ (mit RequestCertificateoder der Konsole)	✓ (verwenden IssueCert ificate)
Verbindung mit Load Balancern und mit dem Internet verbundenen Diensten AWS	✓	Nicht unterstützt

Funktion	ACM	AWS Private CA
Verwaltete Zertifikatserneuerung	✓	Indirekt durch ACM unterstützt
Konsolenunterstützung	✓	Nicht unterstützt
API-Unterstützung	✓	✓
CLI-Unterstützung	✓	✓

Bei der AWS Private CA Erstellung eines Zertifikats folgt es einer Vorlage, die den Zertifikatstyp und die Pfadlänge angibt. Wenn der API- oder CLI-Anweisung, die das Zertifikat erstellt, kein Vorlagen-ARN zur Verfügung gestellt wird, wird standardmäßig die Vorlage EndEntityCertificate/V1 angewendet. Weitere Informationen zu verfügbaren Zertifikatvorlagen finden Sie unter Verwenden Sie Zertifikatsvorlagen AWS Private CA.

ACM-Zertifikate sind zwar auf öffentliches Vertrauen ausgelegt, erfüllen aber AWS Private CA die Anforderungen Ihrer privaten PKI. Folglich können Sie Zertifikate mithilfe der AWS Private CA API und CLI auf eine Weise konfigurieren, die von ACM nicht zugelassen wird. Diese umfassen u. a. folgende:

- Erstellen eines Zertifikats mit einem beliebigen Betreffnamen.
- Verwenden Sie einen der unterstützten Algorithmen und Schlüssellängen für private Schlüssel.
- Verwendung eines der unterstützten Signaturalgorithmen.
- Geben Sie einen beliebigen Gültigkeitszeitraum für Ihre private CA und Ihre privaten Zertifikate an.

Nachdem Sie ein privates TLS-Zertifikat mit erstellt haben AWS Private CA, können Sie es in ACM importieren und mit einem unterstützten AWS Dienst verwenden.

> **Note**
>
> Zertifikate, die mit dem unten stehenden Verfahren, mit dem issue-certificate Befehl oder mit der IssueCertificateAPI-Aktion erstellt wurden, können nicht direkt zur Verwendung nach außen AWS exportiert werden. Sie können jedoch Ihre private Zertifizierungsstelle verwenden, um über ACM ausgestellte Zertifikate zu signieren, und diese Zertifikate können zusammen mit ihren geheimen Schlüsseln exportiert werden. Weitere Informationen finden

Sie unter Anfordern eines privaten Zertifikats und Exportieren eines privaten Zertifikats im ACM-Benutzerhandbuch.

Stellen Sie ein Standardzertifikat aus ()AWS CLI

Sie können den AWS Private CA CLI-Befehl issue-certificate oder die API-Aktion verwenden, um ein Entity-Zertifikat IssueCertificateanzufordern. Dieser Befehl erfordert den Amazon-Ressourcennamen (ARN) der privaten CA, die Sie zum Ausstellen des Zertifikats verwenden möchten. Sie müssen auch eine Certificate Signing Request (CSR) mit einem Programm wie OpenSSL generieren.

Wenn Sie die AWS Private CA API verwenden oder AWS CLI ein privates Zertifikat ausstellen, wird das Zertifikat nicht verwaltet, was bedeutet, dass Sie es nicht mit der ACM-Konsole, der ACM-CLI oder der ACM-API anzeigen oder exportieren können, und das Zertifikat wird nicht automatisch erneuert. Sie können jedoch den PCA-Befehl get-certificate verwenden, um die Zertifikatsdetails abzurufen, und wenn Sie Eigentümer der Zertifizierungsstelle sind, können Sie einen Prüfbericht erstellen.

Überlegungen beim Erstellen von Zertifikaten

- Gemäß RFC 5280 darf die Länge des von Ihnen angegebenen Domainnamens (technisch gesehen der allgemeine Name) 64 Byte (Zeichen), einschließlich Punkte, nicht überschreiten. Wenn Sie einen längeren Domainnamen hinzufügen möchten, geben Sie ihn im Feld Alternativer Betreffname an, das Namen mit einer Länge von bis zu 253 Pfund unterstützt.
- Wenn Sie AWS CLI Version 1.6.3 oder höher verwenden, verwenden Sie das Präfix, `fileb://` wenn Sie Base64-kodierte Eingabedateien angeben, z. B. CSRs Dadurch wird sichergestellt, dass die Daten korrekt AWS Private CA analysiert werden.

Der folgende OpenSSL-Befehl generiert eine CSR und einen privaten Schlüssel für ein Zertifikat:

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

Sie können den Inhalt der CSR wie folgt überprüfen:

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

Die resultierende Ausgabe sollte dem folgenden abgekürzten Beispiel ähneln:

```
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
```

Mit dem folgenden Befehl wird ein Zertifikat erstellt. Da keine Vorlage angegeben ist, wird standardmäßig ein Basiszertifikat für die Endeinheit ausgestellt.

```
$ 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"
```

Der ARN des ausgestellten Zertifikats wird zurückgegeben:

```
{
```

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

> **Note**
>
> AWS Private CA gibt sofort einen ARN mit einer Seriennummer zurück, wenn es den issue-certificate Befehl empfängt. Die Zertifikatsverarbeitung erfolgt jedoch asynchron und kann immer noch fehlschlagen. In diesem Fall schlägt auch ein get-certificate Befehl fehl, der den neuen ARN verwendet.

Stellen Sie mithilfe einer APIPassthrough Vorlage ein Zertifikat mit einem benutzerdefinierten Betreffnamen aus

In diesem Beispiel wird ein Zertifikat ausgestellt, das benutzerdefinierte Elemente des Antragstellernamens enthält. Zusätzlich zur Angabe einer CSR wie der in Stellen Sie ein Standardzertifikat aus ()AWS CLI übergeben Sie dem issue-certificate Befehl zwei zusätzliche Argumente: den ARN einer APIPassthrough Vorlage und eine JSON-Konfigurationsdatei, die die benutzerdefinierten Attribute und ihre Objektkennungen () OIDs spezifiziert. Sie können es nicht zusammen mit verwenden `StandardAttributesCustomAttributes`. Sie können jedoch Standard OIDs als Teil von übergeben. `CustomAttributes` Die standardmäßigen Betreffnamen OIDs sind in der folgenden Tabelle aufgeführt (Informationen aus RFC 4519 und der globalen OID-Referenzdatenbank):

Name des Betreffs	Abkürzung	Objekt-ID
countryName	c	2.5.4.6
commonName	cn	2.5.4.3
DNQualifier [definierter Namenskennzeichner]		2.5.4.46
Qualifikator für die Generierung		2.5.4.44
givenName		2.5.4.42

Name des Betreffs	Abkürzung	Objekt-ID
Initialen		2.5.4.43
Lokalität	l	2.5.4.7
Name der Organisation	o	2.5.4.10
organizationalUnitName	ou	2.5.4.11
Pseudonym		2.5.4.65
Seriennummer		2.5.4.5
st [Bundesstaat]		2.5.4.8
Nachname	sn	2.5.4.4
Titel		2.5.4.12
Domänenkomponente	dc	0.9.2342.19200300.100.1.25
userid		0,9,2342,19200300.100.1.1

Die Beispielkonfigurationsdatei enthält den folgenden Code: `api_passthrough_config.txt`

```
{
  "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"
      }
    ]
```

```
  }
}
```

Verwenden Sie den folgenden Befehl, um das Zertifikat auszustellen:

```
$ 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
```

Der ARN des ausgestellten Zertifikats wird zurückgegeben:

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

Rufen Sie das Zertifikat lokal wie folgt ab:

```
$ 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
```

Sie können den Inhalt des Zertifikats mit OpenSSL überprüfen:

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

> **Note**
>
> Es ist auch möglich, eine private Zertifizierungsstelle zu erstellen, die benutzerdefinierte Attribute an jedes ausgestellte Zertifikat weitergibt.

Stellen Sie mithilfe einer APIPassthrough Vorlage ein Zertifikat mit benutzerdefinierten Erweiterungen aus

In diesem Beispiel wird ein Zertifikat ausgestellt, das benutzerdefinierte Erweiterungen enthält. Dazu müssen Sie dem issue-certificate Befehl drei Argumente übergeben: den ARN einer APIPassthrough Vorlage und eine JSON-Konfigurationsdatei, die die benutzerdefinierten Erweiterungen spezifiziert, und eine CSR wie die in Stellen Sie ein Standardzertifikat aus ()AWS CLI gezeigte.

Die Beispielkonfigurationsdatei `api_passthrough_config.txt` enthält den folgenden Code:

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

Das benutzerdefinierte Zertifikat wird wie folgt ausgestellt:

```
$ 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
```

Der ARN des ausgestellten Zertifikats wird zurückgegeben:

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

Rufen Sie das Zertifikat lokal wie folgt ab:

```
$ 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
```

Sie können den Inhalt des Zertifikats mit OpenSSL überprüfen:

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

Rufen Sie ein privates Zertifikat ab

Sie können die AWS Private CA API verwenden und AWS CLI ein privates Zertifikat ausstellen. Wenn Sie dies tun, können Sie die AWS Private CA API AWS CLI oder verwenden, um dieses Zertifikat abzurufen. Wenn Sie ACM verwendet haben, um Ihre private CA zu erstellen und Zertifikate anzufordern, müssen Sie ACM verwenden, um das Zertifikat und den verschlüsselten privaten Schlüssel zu exportieren. Weitere Informationen finden Sie unter Exportieren eines privaten Zertifikats.

Um ein Endzertifizierungszertifikat abzurufen

Verwenden Sie den AWS CLI Befehl get-certificate, um ein privates Endentitätszertifikat abzurufen. Sie können auch den GetCertificateAPI-Vorgang verwenden. Wir empfehlen, die Ausgabe mit jq, einem SED-ähnlichen Parser, zu formatieren.

> **Note**
>
> Wenn Sie ein Zertifikat widerrufen möchten, können Sie den Befehl get-certificate zum Abrufen der Seriennummer im Hexadezimalformat verwenden. Sie können auch einen Auditbericht zum Abrufen der Hex-Seriennummer erstellen. Weitere Informationen finden Sie unter Verwenden Sie Prüfberichte mit Ihrer privaten Zertifizierungsstelle.

```
$ 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'
```

Dieser Befehl gibt das Zertifikat und die Zertifikatskette im folgenden Standardformat aus.

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

Um ein CA-Zertifikat abzurufen

Sie können die AWS Private CA API verwenden und AWS CLI das Zertifikat der Zertifizierungsstelle (CA) für Ihre private CA abrufen. Führen Sie den Befehl get-certificate-authority-certificate aus. Sie können auch die Operation GetCertificateAuthorityCertificate aufrufen. Wir empfehlen, die Ausgabe mit jq, einem sed-ähnlichen Parser, zu formatieren.

```
$ 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'
```

Dieser Befehl gibt das CA-Zertifikat im folgenden Standardformat aus.

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

Private Zertifikate auflisten

Um Ihre privaten Zertifikate aufzulisten, erstellen Sie einen Prüfbericht, rufen Sie ihn aus dem zugehörigen S3-Bucket ab und analysieren Sie den Berichtsinhalt nach Bedarf. Informationen zum Erstellen von AWS Private CA Auditberichten finden Sie unterVerwenden Sie Prüfberichte mit Ihrer

privaten Zertifizierungsstelle. Informationen zum Abrufen eines Objekts aus einem S3-Bucket finden Sie unter Objekt herunterladen im Amazon Simple Storage Service-Benutzerhandbuch.

Die folgenden Beispiele veranschaulichen Ansätze zur Erstellung von Prüfberichten und deren Analyse nach nützlichen Daten. Die Ergebnisse werden in JSON formatiert, und die Daten werden mit jq, einem SED-ähnlichen Parser, gefiltert.

1. Erstellen Sie einen Prüfbericht.

Der folgende Befehl generiert einen Prüfbericht für eine angegebene Zertifizierungsstelle.

```
$ 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
```

Bei Erfolg gibt der Befehl die ID und den Speicherort des neuen Prüfberichts zurück.

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

2. Rufen Sie einen Prüfbericht ab und formatieren Sie ihn.

Mit diesem Befehl wird ein Prüfbericht abgerufen, sein Inhalt in der Standardausgabe angezeigt und die Ergebnisse so gefiltert, dass nur Zertifikate angezeigt werden, die am oder nach dem 01.12.2020 ausgestellt wurden.

```
$ 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")'
```

Die zurückgegebenen Artikel ähneln den folgenden:

```
{
   "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. Speichern Sie einen Prüfbericht lokal.

Wenn Sie mehrere Abfragen durchführen möchten, ist es praktisch, einen Prüfbericht in einer lokalen Datei zu speichern.

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

Derselbe Filter wie zuvor liefert dieselbe Ausgabe:

```
$ 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. Abfrage innerhalb eines Datumsbereichs

Sie können wie folgt nach Zertifikaten abfragen, die innerhalb eines bestimmten Zeitraums ausgestellt wurden:

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

Der gefilterte Inhalt wird in der Standardausgabe angezeigt:

```
{
   "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. Suchen Sie nach Zertifikaten, die einer bestimmten Vorlage folgen.

Der folgende Befehl filtert den Berichtsinhalt mithilfe eines Vorlagen-ARN:

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

In der Ausgabe werden übereinstimmende Zertifikatsdatensätze angezeigt:

```
{
   "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. Filter nach widerrufenen Zertifikaten

Verwenden Sie den folgenden Befehl, um alle widerrufenen Zertifikate zu finden:

```
$ cat my_local_audit_report.json | jq '.[] | select(.revokedAt != null)'
```

Ein gesperrtes Zertifikat wird wie folgt angezeigt:

```
{
   "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. Filtern Sie mithilfe eines regulären Ausdrucks.

Der folgende Befehl sucht nach Betreffnamen, die die Zeichenfolge „leaf“ enthalten:

```
$ cat my_local_audit_report.json | jq '.[] | select(.subject|test("leaf"))'
```

Passende Zertifikatsdatensätze werden wie folgt zurückgegeben:

```
{
   "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"
}
```

Exportieren Sie ein privates Zertifikat und seinen geheimen Schlüssel

AWS Private CA kann ein privates Zertifikat, das es signiert und ausgestellt hat, nicht direkt exportieren. Sie können es jedoch verwenden, AWS Certificate Manager um ein solches Zertifikat

zusammen mit seinem verschlüsselten geheimen Schlüssel zu exportieren. Das Zertifikat ist dann vollständig portabel und kann überall in Ihrer privaten PKI eingesetzt werden. Weitere Informationen finden Sie im AWS Certificate Manager Benutzerhandbuch unter Exportieren eines privaten Zertifikats.

Als zusätzlichen Vorteil AWS Certificate Manager bietet es eine verwaltete Verlängerung für private Zertifikate, die mithilfe der ACM-Konsole, der `RequestCertificate` ACM-API oder des request-certificate Befehls im ACM-Abschnitt von ausgestellt wurden. AWS CLI Weitere Informationen zu Verlängerungen finden Sie unter Erneuern von Zertifikaten in einer privaten PKI.

Widerrufen Sie ein privates Zertifikat

Sie können ein AWS Private CA Zertifikat mit dem AWS CLI Befehl revoke-certificate oder der RevokeCertificateAPI-Aktion widerrufen. Ein Zertifikat muss möglicherweise vor seinem geplanten Ablauf gesperrt werden, wenn beispielsweise sein geheimer Schlüssel kompromittiert wurde oder die zugehörige Domain ungültig wird. Damit der Widerruf wirksam ist, muss der Client, der das Zertifikat verwendet, bei jedem Versuch, eine sichere Netzwerkverbindung aufzubauen, den Sperrstatus überprüfen können.

AWS Private CA bietet zwei vollständig verwaltete Mechanismen zur Unterstützung der Überprüfung des Sperrstatus: Online Certificate Status Protocol (OCSP) und Zertifikatssperrlisten (CRLs). Mit OCSP fragt der Client eine autoritative Sperrdatenbank ab, die in Echtzeit einen Status zurückgibt. Bei einer CRL überprüft der Client das Zertifikat anhand einer Liste widerrufener Zertifikate, die er regelmäßig herunterlädt und speichert. Clients lehnen es ab, gesperrte Zertifikate zu akzeptieren.

Sowohl OCSP als auch sind von den in Zertifikaten eingebetteten Validierungsinformationen CRLs abhängig. Aus diesem Grund muss eine ausstellende Zertifizierungsstelle vor der Ausstellung so konfiguriert werden, dass sie einen oder beide dieser Mechanismen unterstützt. Informationen zur Auswahl und Implementierung der verwalteten Sperrung durch finden Sie AWS Private CA unterPlanen Sie Ihre Methode zum Widerruf von AWS Private CA Zertifikaten.

Widerrufene Zertifikate werden immer in AWS Private CA Prüfberichten aufgezeichnet.

> **Note**
>
> Für kontoübergreifende Anrufer ist ein Share mit entsprechender `AWSRAMRevokeCertificateCertificateAuthority` Genehmigung erforderlich. Sperrberechtigungen sind nicht in enthalten.

`AWSRAMDefaultPermissionCertificateAuthority` Um den Widerruf durch kontoübergreifende Emittenten zu ermöglichen, muss der CA-Administrator zwei RAM-Shares erstellen, die beide auf dieselbe CA verweisen:

1. Eine Aktie mit der `AWSRAMRevokeCertificateCertificateAuthority` entsprechenden Genehmigung.
2. Eine Aktie mit der `AWSRAMDefaultPermissionCertificateAuthority` Genehmigung.

Um ein Zertifikat zu widerrufen

Verwenden Sie die RevokeCertificateAPI-Aktion oder den Befehl revoke-certificate, um ein privates PKI-Zertifikat zu widerrufen. Die Seriennummer muss im Hexadezimalformat vorliegen. Sie können die Seriennummer abrufen, indem Sie den Befehl get-certificate aufrufen. Der Befehl `revoke-certificate` gibt keine Antwort zurück.

```
$ 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"
```

Widerrufene Zertifikate und OCSP

Bei OCSP-Antworten kann es bis zu 60 Minuten dauern, bis der neue Status angezeigt wird, wenn Sie ein Zertifikat widerrufen. Im Allgemeinen unterstützt OCSP tendenziell eine schnellere Verteilung von Sperrinformationen, da OCSP-Antworten im Gegensatz zu solchen, CRLs die von Clients tagelang zwischengespeichert werden können, in der Regel nicht von Clients zwischengespeichert werden.

Widerrufene Zertifikate in einer Zertifikatssperrliste

Eine Zertifikatssperrliste wird in der Regel etwa 30 Minuten nach Widerrufen eines Zertifikats aktualisiert. Falls eine CRL-Aktualisierung aus irgendeinem Grund fehlschlägt, werden alle 15 Minuten weitere AWS Private CA Versuche unternommen.

Mit Amazon CloudWatch können Sie Alarme für die Metriken `CRLGenerated` und erstellen`MisconfiguredCRLBucket`. Weitere Informationen finden Sie unter Unterstützte

CloudWatch Metriken. Weitere Informationen zum Erstellen und Konfigurieren finden CRLs Sie unterRichten Sie eine CRL ein für AWS Private CA.

Das folgende Beispiel zeigt ein widerrufenes Zertifikat in einer Zertifikatssperrliste (Certificate Revocation List, (CRL)).

```
Certificate Revocation List (CRL):
        Version 2 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: /C=US/ST=WA/L=Seattle/O=Examples LLC/OU=Corporate Office/
CN=www.example.com
        Last Update: Jan 10 19:28:47 2018 GMT
        Next Update: Jan  8 20:28:47 2028 GMT
        CRL extensions:
            X509v3 Authority key identifier:
                keyid:3B:F0:04:6B:51:54:1F:C9:AE:4A:C0:2F:11:E6:13:85:D8:84:74:67

            X509v3 CRL Number:
                1515616127629
Revoked Certificates:
    Serial Number: B17B6F9AE9309C51D5573BCA78764C23
        Revocation Date: Jan  9 17:19:17 2018 GMT
        CRL entry extensions:
            X509v3 CRL Reason Code:
                Key Compromise
    Signature Algorithm: sha256WithRSAEncryption
         21:2f:86:46:6e:0a:9c:0d:85:f6:b6:b6:db:50:ce:32:d4:76:
         99:3e:df:ec:6f:c7:3b:7e:a3:6b:66:a7:b2:83:e8:3b:53:42:
         f0:7a:bc:ba:0f:81:4d:9b:71:ee:14:c3:db:ad:a0:91:c4:9f:
         98:f1:4a:69:9a:3f:e3:61:36:cf:93:0a:1b:7d:f7:8d:53:1f:
         2e:f8:bd:3c:7d:72:91:4c:36:38:06:bf:f9:c7:d1:47:6e:8e:
         54:eb:87:02:33:14:10:7f:b2:81:65:a1:62:f5:fb:e1:79:d5:
         1d:4c:0e:95:0d:84:31:f8:5d:59:5d:f9:2b:6f:e4:e6:60:8b:
         58:7d:b2:a9:70:fd:72:4f:e7:5b:e4:06:fc:e7:23:e7:08:28:
         f7:06:09:2a:a1:73:31:ec:1c:32:f8:dc:03:ea:33:a8:8e:d9:
         d4:78:c1:90:4c:08:ca:ba:ec:55:c3:00:f4:2e:03:b2:dd:8a:
         43:13:fd:c8:31:c9:cd:8d:b3:5e:06:c6:cc:15:41:12:5d:51:
         a2:84:61:16:a0:cf:f5:38:10:da:a5:3b:69:7f:9c:b0:aa:29:
         5f:fc:42:68:b8:fb:88:19:af:d9:ef:76:19:db:24:1f:eb:87:
         65:b2:05:44:86:21:e0:b4:11:5c:db:f6:a2:f9:7c:a6:16:85:
         0e:81:b2:76
```

Widerrufene Zertifikate in einem Auditbericht

Alle Zertifikate, einschließlich der widerrufenen Zertifikate, werden in den Auditbericht für eine private Zertifizierungsstelle aufgenommen. Das folgende Beispiel zeigt einen Auditbericht mit einem ausgestellten und einem widerrufenen Zertifikat. Weitere Informationen finden Sie unter Verwenden Sie Prüfberichte mit Ihrer privaten Zertifizierungsstelle.

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

Automatisieren Sie den Export eines erneuerten Zertifikats

Wenn Sie eine Zertifizierungsstelle erstellen, können Sie diese Zertifizierungsstelle in die Zertifikatsausstellung und -erneuerung importieren AWS Certificate Manager und ACM die Verwaltung überlassen. AWS Private CA Wenn ein Zertifikat, das erneuert wird, einem integrierten

Dienst zugeordnet ist, wendet der Dienst das neue Zertifikat nahtlos an. Wenn das Zertifikat jedoch ursprünglich für die Verwendung an anderer Stelle in Ihrer PKI-Umgebung exportiert wurde (z. B. auf einem lokalen Server oder einer Appliance), müssen Sie es nach der Verlängerung erneut exportieren.

Eine Beispiellösung, die den ACM-Exportprozess mithilfe von Amazon EventBridge und AWS Lambda automatisiert, finden Sie unter Automatisieren des Exports erneuerter Zertifikate.

Verwenden Sie Zertifikatsvorlagen AWS Private CA

AWS Private CA verwendet Konfigurationsvorlagen, um sowohl CA-Zertifikate als auch Endentitätszertifikate auszustellen. Wenn Sie ein CA-Zertifikat von der PCA-Konsole aus ausstellen, wird die entsprechende Stamm- oder untergeordnete CA-Zertifikatsvorlage automatisch angewendet.

Wenn Sie die CLI oder API verwenden, um ein Zertifikat auszustellen, können Sie einen Vorlagen-ARN als Parameter für die `IssueCertificate` Aktion angeben. Wenn Sie keinen ARN angeben, wird die `EndEntityCertificate/V1` Vorlage standardmäßig angewendet. Weitere Informationen finden Sie in der Dokumentation zu den IssueCertificateBefehlen API und issue-certificate.

> **Note**
>
> AWS Certificate Manager (ACM) Benutzer mit kontenübergreifendem gemeinsamen Zugriff auf eine private Zertifizierungsstelle können verwaltete Zertifikate ausstellen, die von der Zertifizierungsstelle signiert sind. Kontoübergreifende Aussteller sind durch eine ressourcenbasierte Richtlinie eingeschränkt und haben nur Zugriff auf die folgenden Vorlagen für Endzertifikate:
>
> - EndEntityCertificate/V1
> - EndEntityClientAuthCertificate/V1
> - EndEntityServerAuthCertificate/V1
> - BlankEndEntityCertificate_ /V1 APIPassthrough
> - BlankEndEntityCertificate_ /V1 APICSRPassthrough
> - Untergeordnet _ 0/V1 CACertificate PathLen
>
> Weitere Informationen finden Sie unter Ressourcenbasierte Richtlinien.

Themen

AWS Private CA Vorlagensorten

AWS Private CA unterstützt vier Arten von Vorlagen.

- Basisvorlagen

  Vordefinierte Vorlagen, in denen keine Passthrough-Parameter zulässig sind.

- CSRPassthrough Vorlagen

  Vorlagen, die ihre entsprechenden Basisvorlagenversionen erweitern, indem sie CSR-Passthrough zulassen. Erweiterungen in der CSR, die für die Ausstellung des Zertifikats verwendet werden, werden auf das ausgestellte Zertifikat kopiert. In Fällen, in denen die CSR Erweiterungswerte enthält, die mit der Vorlagendefinition in Konflikt stehen, hat die Vorlagendefinition immer die höhere Priorität. Weitere Informationen zur Priorität finden Sie unterAWS Private CA Vorlagenreihenfolge der Operationen.

- APIPassthrough Vorlagen

  Vorlagen, die ihre entsprechenden Basisvorlagenversionen erweitern, indem sie API-Passthrough zulassen. Dynamische Werte, die dem Administrator oder anderen Zwischensystemen bekannt sind, sind der Entität, die das Zertifikat anfordert, möglicherweise nicht bekannt, können möglicherweise nicht in einer Vorlage definiert werden und sind möglicherweise nicht in der CSR verfügbar. Der CA-Administrator kann jedoch zusätzliche Informationen aus einer anderen Datenquelle, z. B. einem Active Directory, abrufen, um die Anfrage abzuschließen. Wenn ein Computer beispielsweise nicht weiß, zu welcher Organisationseinheit er gehört, kann der Administrator die Informationen in Active Directory nachschlagen und sie der Zertifikatsanforderung hinzufügen, indem er die Informationen in eine JSON-Struktur einfügt.

  Die Werte im `ApiPassthrough` Parameter der `IssueCertificate` Aktion werden in das ausgestellte Zertifikat kopiert. In Fällen, in denen der `ApiPassthrough` Parameter Informationen enthält, die mit der Vorlagendefinition in Konflikt stehen, hat die Vorlagendefinition immer die höhere Priorität. Weitere Informationen zur Priorität finden Sie unterAWS Private CA Vorlagenreihenfolge der Operationen.

- APICSRPassthrough Vorlagen

  Vorlagen, die ihre entsprechenden Basisvorlagenversionen erweitern, indem sie sowohl API- als auch CSR-Passthrough zulassen. Erweiterungen in der CSR, die zur Ausstellung des Zertifikats verwendet wurden, werden auf das ausgestellte Zertifikat kopiert, und Werte im `ApiPassthrough` `IssueCertificate` Aktionsparameter werden ebenfalls kopiert. In Fällen, in denen ein Konflikt zwischen der Vorlagendefinition, den API-Passthrough-Werten und den CSR-Passthrough-Erweiterungen besteht, hat die Vorlagendefinition die höchste Priorität, gefolgt von den API-Passthrough-Werten, gefolgt von den CSR-Passthrough-Erweiterungen. Weitere Informationen zur Priorität finden Sie unter. AWS Private CA Vorlagenreihenfolge der Operationen

In den folgenden Tabellen sind alle von unterstützten Vorlagentypen AWS Private CA mit Links zu ihren Definitionen aufgeführt.

> **Note**
>
> Informationen zu Vorlagen ARNs in GovCloud Regionen finden Sie AWS Private Certificate Authorityim AWS GovCloud (US) Benutzerhandbuch.

Basisvorlagen

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
CodeSigningCertificate/V1	`arn:aws:acm-pca:::template/CodeSigningCertificate/V1`	Codesignatur
EndEntityCertificate/V1	`arn:aws:acm-pca:::template/EndEntityCertificate/V1`	Endentität
EndEntityClientAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate/V1`	Endentität
EndEntityServerAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntity`	Endentität

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
	`ServerAuthCertificate/V1`	
OCSPSigningZertifikat/V1	`arn:aws:acm-pca:::template/OCSPSigningCertificate/V1`	OCSP Signing
Stamm /V1 CACertificate	`arn:aws:acm-pca:::template/RootCACertificate/V1`	CA
Untergeordnet _ 0/V1 CACertificate PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1`	CA
Untergeordnet _ 1/V1 CACertificate PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1/V1`	CA
Untergeordnet _ 2/V1 CACertificate PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2/V1`	CA
Untergeordnet _ 3/V1 CACertificate PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3/V1`	CA

CSRPassthrough Vorlagen

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
BlankEndEntityCertificate_ CSRPassthrough /V1	`arn:aws:acm-pca::: template/BlankEndE ntityCertificate_C SRPassthrough/V1`	Endentität
BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1 CSRPassthrough	`arn:aws:acm-pca::: template/BlankEndE ntityCertificate_C riticalBasicConstr aints_CSRPassthrough/ V1`	Endentität
BlankSubordinateCACertifica te_PathLen0_CSRPassthrough/V1	`arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen0_CSRPas sthrough/V1`	CA
BlankSubordinateCACertifica te_PathLen1_CSRPassthrough/V1	`arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen1_CSRPas sthrough/V1`	CA
BlankSubordinateCACertifica te_PathLen2_CSRPassthrough/V1	`arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen2_CSRPas sthrough/V1`	CA
BlankSubordinateCACertifica te_PathLen3_CSRPassthrough/V1	`arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica`	CA

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
	te_PathLen3_CSRPas sthrough/V1	
CodeSigningCertificate_ /V1 CSRPassthrough	arn:aws:acm-pca::: template/CodeSigni ngCertificate_CSRP assthrough/V1	Codesignatur
EndEntityCertificate_ /V1 CSRPassthrough	arn:aws:acm-pca::: template/EndEntity Certificate_CSRPas sthrough/V1	Endentität
EndEntityClientAuthCertificate_ / V1 CSRPassthrough	arn:aws:acm-pca::: template/EndEntity ClientAuthCertific ate_CSRPassthrough/V1	Endentität
EndEntityServerAuthCertificate_ / V1 CSRPassthrough	arn:aws:acm-pca::: template/EndEntity ServerAuthCertific ate_CSRPassthrough/V1	Endentität
OCSPSigningZertifikat_ /V1 CSRPassthrough	arn:aws:acm-pca::: template/OCSPSigni ngCertificate_CSRP assthrough/V1	OCSP Signing
Untergeordnet _ 0_ /V1 CACertifi cate PathLen CSRPassthrough	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen0_CSRPassthrough/ V1	CA

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
Untergeordnet _ 1_ /V1 CACertificate PathLen CSRPassthrough	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_CSRPassthrough/V1`	CA
Untergeordnet _ 2_ /V1 CACertificate PathLen CSRPassthrough	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_CSRPassthrough/V1`	CA
Untergeordnet _ 3_ /V1 CACertificate PathLen CSRPassthrough	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_CSRPassthrough/V1`	CA

APIPassthrough Vorlagen

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
BlankEndEntityCertificate_APIPassthrough /V1	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_APIPassthrough/V1`	Endentität
BlankEndEntityCertificate_CriticalBasicConstraints _ /V1 APIPassthrough	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough/V1`	Endentität

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
CodeSigningCertificate_ /V1 APIPassthrough	arn:aws:acm-pca:::template/CodeSigningCertificate_APIPassthrough/V1	Codesignatur
EndEntityCertificate_ /V1 APIPassthrough	arn:aws:acm-pca:::template/EndEntityCertificate_APIPassthrough/V1	Endentität
EndEntityClientAuthCertificate_ /V1 APIPassthrough	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_APIPassthrough/V1	Endentität
EndEntityServerAuthCertificate_ /V1 APIPassthrough	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APIPassthrough/V1	Endentität
OCSPSigningZertifikat_ /V1 APIPassthrough	arn:aws:acm-pca:::template/OCSPSigningCertificate_APIPassthrough/V1	OCSP Signing
Wurzel _ /V1 CACertificate APIPassthrough	arn:aws:acm-pca:::template/RootCACertificate_APIPassthrough/V1	CA
BlankRootCACertificate_ /V1 APIPassthrough	arn:aws:acm-pca:::template/BlankRootCACertificate_APIPassthrough/V1	CA

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
BlankRootCACertificate_ PathLen 0_ /V1 APIPassthrough	`arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen0_APIPassthrough/V1`	CA
BlankRootCACertificate_ 1_ /V1 PathLen APIPassthrough	`arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen1_APIPassthrough/V1`	CA
BlankRootCACertificate_ 2_ /V1 PathLen APIPassthrough	`arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen2_APIPassthrough/V1`	CA
BlankRootCACertificate_ 3_ /V1 PathLen APIPassthrough	`arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen3_APIPassthrough/V1`	CA
Untergeordnet _ 0_ /V1 CACertificate PathLen APIPassthrough	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APIPassthrough/V1`	CA
BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1`	CA

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
Untergeordnet _ 1_ /V1 CACertificate PathLen APIPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
Untergeordnet _ 2_ /V1 CACertificate PathLen APIPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
Untergeordnet _ 3_ /V1 CACertificate PathLen APIPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APIPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1	CA

APICSRPassthrough Vorlagen

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
BlankEndEntityCertificate_ APICSRPassthrough /V1	`arn:aws:acm-pca::: template/BlankEndE ntityCertificate_A PICSRPassthrough/V1`	Endentität
BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1 APICSRPassthrough	`arn:aws:acm-pca::: template/BlankEndE ntityCertificate_C riticalBasicConstr aints_APICSRPassth rough/V1`	Endentität
CodeSigningCertificate_ /V1 APICSRPassthrough	`arn:aws:acm-pca::: template/CodeSigni ngCertificate_APIC SRPassthrough/V1`	Codesignatur
EndEntityCertificate_ /V1 APICSRPassthrough	`arn:aws:acm-pca::: template/EndEntity Certificate_APICSR Passthrough/V1`	Endentität
EndEntityClientAuthCertificate_ / V1 APICSRPassthrough	`arn:aws:acm-pca::: template/EndEntity ClientAuthCertific ate_APICSRPassthrough/ V1`	Endentität
EndEntityServerAuthCertificate_ / V1 APICSRPassthrough	`arn:aws:acm-pca::: template/EndEntity ServerAuthCertific`	Endentität

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
	ate_APICSRPassthrough/V1	
OCSPSigningZertifikat_ /V1 APICSRPassthrough	arn:aws:acm-pca:::template/OCSPSigningCertificate_APICSRPassthrough/V1	OCSP Signing
Untergeordnet _ 0_ /V1 CACertificate PathLen APICSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
Untergeordnet _ 1_ /V1 CACertificate PathLen APICSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
Untergeordnet CACertificate _ 2_/3_ V1 PathLen APICSRPassthrough PathLen APIPassthrough	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APICSRPassthrough/V1`	CA
BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1`	CA
Untergeordnet _ 3_ /V1 CACertificate PathLen APICSRPassthrough	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA
BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA

AWS Private CA Vorlagenreihenfolge der Operationen

Die in einem ausgestellten Zertifikat enthaltenen Informationen können aus vier Quellen stammen: der Vorlagendefinition, dem API-Passthrough, dem CSR-Passthrough und der CA-Konfiguration.

API-Passthrough-Werte werden nur berücksichtigt, wenn Sie eine API-Passthrough- oder APICSR-Passthrough-Vorlage verwenden. CSR-Passthrough wird nur berücksichtigt, wenn Sie eine oder eine APICSR-Passthrough-Vorlage verwenden. CSRPassthrough Wenn diese Informationsquellen miteinander in Konflikt stehen, gilt in der Regel eine allgemeine Regel: Für jeden Erweiterungswert hat die Vorlagendefinition die höchste Priorität, gefolgt von API-Passthrough-Werten, gefolgt von CSR-Passthrough-Erweiterungen.

Beispiele

1. Die Vorlagendefinition für EndEntityClientAuthCertificate_ APIPassthrough definiert die ExtendedKeyUsage Erweiterung mit dem Wert „TLS-Webserver-Authentifizierung, TLS-Webclient-Authentifizierung“. Wenn in der CSR oder im `IssueCertificate` `ApiPassthrough` Parameter definiert ExtendedKeyUsage ist, ExtendedKeyUsage wird der `ApiPassthrough` Wert für ignoriert, da die Vorlagendefinition Priorität hat, und der CSR-Wert für den ExtendedKeyUsage Wert wird ignoriert, da es sich bei der Vorlage nicht um eine CSR-Passthrough-Variante handelt.

   > **Note**
   >
   > Die Vorlagendefinition kopiert dennoch andere Werte aus der CSR, wie z. B. Betreff und Alternativer Name des Betreffs. Diese Werte werden immer noch aus der CSR übernommen, obwohl es sich bei der Vorlage nicht um eine CSR-Passthrough-Variante handelt, da die Vorlagendefinition immer die höchste Priorität hat.

2. Die Vorlagendefinition für EndEntityClientAuthCertificate_ APICSRPassthrough definiert, dass die Erweiterung Subject Alternative Name (SAN) aus der API oder CSR kopiert wurde. Wenn die SAN-Erweiterung in der CSR definiert und im `IssueCertificate ApiPassthrough` Parameter angegeben ist, hat der API-Passthrough-Wert Vorrang, da API-Passthrough-Werte Vorrang vor CSR-Passthrough-Werten haben.

AWS Private CA Vorlagendefinitionen

Die folgenden Abschnitte enthalten Konfigurationsdetails zu unterstützten AWS Private CA Zertifikatsvorlagen.

BlankEndEntityCertificate_ APIPassthrough /V1-Definition

Mit leeren Vorlagen für Endentitätszertifikate können Sie Endentitätszertifikate ausstellen, für die nur X.509 Basic-Einschränkungen gelten. Dies ist das einfachste Endentitätszertifikat, das ausgestellt werden AWS Private CA kann, es kann jedoch mithilfe der API-Struktur angepasst werden. Die Erweiterung Basic Constraints definiert, ob es sich bei dem Zertifikat um ein CA-Zertifikat handelt oder nicht. Eine leere Vorlage für ein Endentitätszertifikat erzwingt für Basic-Einschränkungen den Wert FALSE, um sicherzustellen, dass ein Endentitätszertifikat und kein CA-Zertifikat ausgestellt wird.

Sie können leere Passthrough-Vorlagen verwenden, um Smartcard-Zertifikate auszustellen, für die bestimmte Werte für Key Usage (KU) und Extended Key Usage (EKU) erforderlich sind.

Beispielsweise können für die erweiterte Schlüsselverwendung eine Clientauthentifizierung und eine Smartcard-Anmeldung erforderlich sein, und für die Verwendung von Schlüsseln sind möglicherweise digitale Signatur, Nichtabweisung und Schlüsselverschlüsselung erforderlich. Im Gegensatz zu anderen Passthrough-Vorlagen ermöglichen leere Vorlagen für Endentitätszertifikate die Konfiguration von KU- und EKU-Erweiterungen, wobei KU einer der neun unterstützten Werte sein kann (DigitalSignature, NonRepudiation, KeyEncipherment, DataEncipherment keyCertSign, KeyAgreementRLSign,, c, EncipherOnly und DecipherOnly) und EKU jeder der unterstützten Werte sein kann (ServerAuth, ClientAuth, Codesigning, EmailProtection, Timestamping, und OCSPSigning) plus benutzerdefinierte Erweiterungen.

BlankEndEntityCertificateAPIPassthrough_/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	CA: FALSCH
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankEndEntityCertificate_ /V1-Definition APICSRPassthrough

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_ APIPassthrough /V1-Definition.

BlankEndEntityCertificate_ APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]

X509v3-Parameter	Wert
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	CA: FALSCH
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1-Definition APICSRPassthrough

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_ APIPassthrough /V1-Definition.

BlankEndEntityCertificate_ CriticalBasicConstraints _ APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	Kritisch, CA: FALSE
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration, API oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1-Definition APIPassthrough

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_ APIPassthrough /V1-Definition.

BlankEndEntityCertificate_ CriticalBasicConstraints _ APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	Kritisch, CA: FALSE
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder API]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankEndEntityCertificate_ CriticalBasicConstraints _ /V1-Definition CSRPassthrough

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_ APIPassthrough /V1-Definition.

BlankEndEntityCertificate_ CriticalBasicConstraints _ CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	Kritisch, CA: FALSE

X509v3-Parameter	Wert
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankEndEntityCertificate_ /V1-Definition CSRPassthrough

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_ APIPassthrough /V1-Definition.

BlankEndEntityCertificate_ CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	CA: FALSCH
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 0`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 0`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]

X509v3-Parameter	Wert
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 0`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 1`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 1`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 1`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 2`

X509v3-Parameter	Wert
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 2`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 2`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 3`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 3`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unterBlankEndEntityCertificate_APIPassthrough /V1-Definition.

BlankSubordinateCACertificate_PathLen3_APICSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 3`

X509v3-Parameter	Wert
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

CodeSigningCertificate/V1-Definition

Diese Vorlage wird verwendet, um Zertifikate für die Codesignatur zu erstellen. Sie können Codesignaturzertifikate AWS Private CA mit jeder Codesignaturlösung verwenden, die auf einer privaten CA-Infrastruktur basiert. Beispielsweise AWS IoT können Kunden, die Code Signing for verwenden, ein Codesignaturzertifikat mit generieren und es in importieren. AWS Private CA AWS Certificate Manager Weitere Informationen finden Sie unter Wozu dient Code Signing? AWS IoTund Besorgen Sie sich ein Codesignaturzertifikat und importieren Sie es.

CodeSigningCertificate/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	`CA:FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Verwendung von Schlüsseln	Kritisch, Codesignatur
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

CodeSigningCertificate_ APICSRPassthrough /V1-Definition

Diese Vorlage erweitert CodeSigningCertificate /V1 um die Unterstützung von API- und CSR-Passthrough-Werten.

CodeSigningCertificateAPICSRPassthrough_ /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	`CA:FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Verwendung von Schlüsseln	Kritisch, Codesignatur
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

CodeSigningCertificate_ /V1-Definition APIPassthrough

Diese Vorlage ist mit der `CodeSigningCertificate` Vorlage identisch, AWS Private CA mit einem Unterschied: In dieser Vorlage werden zusätzliche Erweiterungen über die API an das Zertifikat übergeben, wenn die Erweiterungen nicht in der Vorlage angegeben sind. In der Vorlage angegebene Erweiterungen haben immer Vorrang vor Erweiterungen in der API.

CodeSigningCertificate_ APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	`CA:FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Verwendung von Schlüsseln	Kritisch, Codesignatur
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

CodeSigningCertificate_ /V1-Definition CSRPassthrough

Diese Vorlage ist identisch mit der `CodeSigningCertificate` Vorlage, AWS Private CA mit einem Unterschied: In dieser Vorlage werden zusätzliche Erweiterungen aus der Certificate Signing Request (CSR) an das Zertifikat übergeben, wenn die Erweiterungen nicht in der Vorlage angegeben sind. Erweiterungen, die in der Vorlage angegeben sind, haben stets Vorrang vor Erweiterungen in der CSR.

CodeSigningCertificate_ /V1 CSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	`CA:FALSE`

X509v3-Parameter	Wert
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Verwendung von Schlüsseln	Kritisch, Codesignatur
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

EndEntityCertificate/V1-Definition

Diese Vorlage wird verwendet, um Zertifikate für Endentitäten wie Betriebssysteme oder Webserver zu erstellen.

EndEntityCertificate/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	CA:FALSE
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webserver-Authentifizierung, TLS-Webclient-Authentifizierung

X509v3-Parameter	Wert
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

EndEntityCertificate_ APICSRPassthrough /V1-Definition

Diese Vorlage erweitert EndEntityCertificate /V1 um die Unterstützung von API- und CSR-Passthrough-Werten.

EndEntityCertificateAPICSRPassthrough_ /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webserver-Authentifizierung, TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityCertificate_ /V1-Definition APIPassthrough

Diese Vorlage ist mit der `EndEntityCertificate` Vorlage identisch, AWS Private CA mit einem Unterschied: In dieser Vorlage werden zusätzliche Erweiterungen über die API an das Zertifikat übergeben, wenn die Erweiterungen nicht in der Vorlage angegeben sind. In der Vorlage angegebene Erweiterungen haben immer Vorrang vor Erweiterungen in der API.

EndEntityCertificate_ APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	CA:`FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webserver-Authentifizierung, TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityCertificate_ /V1-Definition CSRPassthrough

Diese Vorlage ist identisch mit der `EndEntityCertificate` Vorlage, AWS Private CA mit einem Unterschied: In dieser Vorlage werden zusätzliche Erweiterungen aus der Certificate Signing Request (CSR) an das Zertifikat übergeben, wenn die Erweiterungen nicht in der Vorlage angegeben sind. Erweiterungen, die in der Vorlage angegeben sind, haben stets Vorrang vor Erweiterungen in der CSR.

EndEntityCertificate_ /V1 CSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	CA:FALSE
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webserver-Authentifizierung, TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

EndEntityClientAuthCertificate/V1-Definition

Diese Vorlage unterscheidet sich von der Vorlage `EndEntityCertificate` nur durch den Wert für die erweiterte Schlüsselverwendung, der sie auf die TLS-Webclient-Authentifizierung beschränkt.

EndEntityClientAuthCertificate/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	CA:FALSE

X509v3-Parameter	Wert
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

EndEntityClientAuthCertificate_ /V1-Definition APICSRPassthrough

Diese Vorlage erweitert EndEntityClientAuthCertificate /V1 um die Unterstützung von API- und CSR-Passthrough-Werten.

EndEntityClientAuthCertificateAPICSRPassthrough_ /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webclient-Authentifizierung

X509v3-Parameter	Wert
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityClientAuthCertificate_ /V1-Definition APIPassthrough

Diese Vorlage ist identisch mit der Vorlage `EndEntityClientAuthCertificate`, mit einem Unterschied: Übergibt in dieser AWS Private CA Vorlage zusätzliche Erweiterungen über die API an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. In der Vorlage angegebene Erweiterungen haben immer Vorrang vor Erweiterungen in der API.

EndEntityClientAuthCertificate_ APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	CA:`FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityClientAuthCertificate_ /V1-Definition CSRPassthrough

Diese Vorlage ist identisch mit der Vorlage `EndEntityClientAuthCertificate`, mit einem Unterschied: Übergibt in dieser Vorlage AWS Private CA zusätzliche Erweiterungen aus der Certificate Signing Request (CSR) an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. Erweiterungen, die in der Vorlage angegeben sind, haben stets Vorrang vor Erweiterungen in der CSR.

EndEntityClientAuthCertificate_ /V1 CSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	CA:`FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

EndEntityServerAuthCertificate/V1-Definition

Diese Vorlage unterscheidet sich von der Vorlage `EndEntityCertificate` nur durch den Wert für die erweiterte Schlüsselverwendung, der sie auf die TLS-Webserver-Authentifizierung beschränkt.

EndEntityServerAuthCertificate/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	CA:FALSE
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webserver-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

EndEntityServerAuthCertificate_ APICSRPassthrough /V1-Definition

Diese Vorlage erweitert EndEntityServerAuthCertificate /V1 um die Unterstützung von API- und CSR-Passthrough-Werten.

EndEntityServerAuthCertificateAPICSRPassthrough_ /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]

X509v3-Parameter	Wert
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webserver-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityServerAuthCertificate_ /V1-Definition APIPassthrough

Diese Vorlage ist identisch mit der Vorlage `EndEntityServerAuthCertificate`, mit einem Unterschied: Übergibt in dieser AWS Private CA Vorlage zusätzliche Erweiterungen über die API an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. In der Vorlage angegebene Erweiterungen haben immer Vorrang vor Erweiterungen in der API.

EndEntityServerAuthCertificate_ APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	CA:`FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webserver-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityServerAuthCertificate_ /V1-Definition CSRPassthrough

Diese Vorlage ist identisch mit der Vorlage `EndEntityServerAuthCertificate`, mit einem Unterschied: Übergibt in dieser Vorlage AWS Private CA zusätzliche Erweiterungen aus der Certificate Signing Request (CSR) an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. Erweiterungen, die in der Vorlage angegeben sind, haben stets Vorrang vor Erweiterungen in der CSR.

EndEntityServerAuthCertificate_ /V1 CSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	CA:`FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Verwendung von Schlüsseln	TLS-Webserver-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

OCSPSigningZertifikat/V1-Definition

Diese Vorlage wird verwendet, um Zertifikate zum Signieren von OCSP-Antworten zu erstellen. Die Vorlage ist identisch mit der `CodeSigningCertificate` Vorlage, mit der Ausnahme, dass der Wert für die erweiterte Schlüsselverwendung die OCSP-Signierung anstelle der Codesignatur angibt.

OCSPSigningZertifikat/V1

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	`CA:FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Verwendung von Schlüsseln	Kritisch, OCSP-Signatur
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

OCSPSigningDefinition von Certificate_ /V1 APICSRPassthrough

Diese Vorlage erweitert das OCSPSigning Zertifikat/V1-Format um die Unterstützung von API- und CSR-Passthrough-Werten.

OCSPSigningCertificate_ /V1 APICSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	`CA:FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]

X509v3-Parameter	Wert
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Verwendung von Schlüsseln	Kritisch, OCSP-Signatur
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

OCSPSigningDefinition von Certificate_ /V1 APIPassthrough

Diese Vorlage ist identisch mit der Vorlage `OCSPSigningCertificate`, mit einem Unterschied: Übergibt in dieser Vorlage AWS Private CA zusätzliche Erweiterungen über die API an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. In der Vorlage angegebene Erweiterungen haben immer Vorrang vor Erweiterungen in der API.

OCSPSigningZertifikat_ /V1 APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	`CA:FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Verwendung von Schlüsseln	Kritisch, OCSP-Signatur
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

OCSPSigningDefinition von Certificate_ /V1 CSRPassthrough

Diese Vorlage ist identisch mit der Vorlage `0CSPSigningCertificate`, mit einem Unterschied: Übergibt in dieser Vorlage zusätzliche AWS Private CA Erweiterungen aus der Zertifikatsignieranforderung (CSR) an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. Erweiterungen, die in der Vorlage angegeben sind, haben stets Vorrang vor Erweiterungen in der CSR.

OCSPSigningCertificate_ /V1 CSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	`CA:FALSE`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Verwendung von Schlüsseln	Kritisch, OCSP-Signatur
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

Root-/V1-Definition CACertificate

Diese Vorlage wird verwendet, um selbstsignierte CA-Zertifikate auszustellen. CA-Zertifikate enthalten eine Erweiterung für wichtige grundlegende Einschränkungen, wobei das CA-Feld so festgelegt ist, dass `TRUE` zum Ausstellen von CA-Zertifikaten verwendet werden kann. Die

Vorlage gibt keine Pfadlänge (pathLenConstraint) an, da dies eine future Erweiterung der Hierarchie verhindern könnte. Eine erweiterte Schlüsselverwendung ist ausgeschlossen, um die Verwendung des CA-Zertifikats als TLS-Client- oder Serverzertifikat zu verhindern. Es werden keine Zertifikatsperrlisteninformationen angegeben, da ein selbstsigniertes Zertifikat nicht widerrufen werden kann.

Stamm /V1 CACertificate

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur keyCertSign, CRL-Zeichen
CRL-Verteilungspunkte	N/A

Definition von Root CACertificate _ APIPassthrough /V1

Diese Vorlage erweitert Root CACertificate /V1 um die Unterstützung von API-Passthrough-Werten.

Root _ /V1 CACertificate APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`
Schlüssel-ID der Behörde	[Passthrough von der API]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]

X509v3-Parameter	Wert
Schlüsselnutzung	Kritisch, digitale Signatur keyCertSign, CRL-Zeichen
CRL-Verteilungspunkte*	N/A

BlankRootCACertificate_ /V1-Definition APIPassthrough

Mit leeren Stammzertifikatvorlagen können Sie Stammzertifikate ausstellen, für die nur grundlegende X.509-Einschränkungen gelten. Dies ist das einfachste Stammzertifikat, das ausgestellt werden AWS Private CA kann, es kann jedoch mithilfe der API-Struktur angepasst werden. Die Erweiterung Basic Constraints definiert, ob es sich bei dem Zertifikat um ein CA-Zertifikat handelt oder nicht. Eine leere Stammzertifikatvorlage erzwingt den Wert vier Basiseinschränkungen, um sicherzustellen, dass ein Stammzertifizierungsstellenzertifikat ausgestellt wird. `TRUE`

Sie können leere Passthrough-Root-Vorlagen verwenden, um Stammzertifikate auszustellen, für die bestimmte Werte für die Schlüsselverwendung (KU) erforderlich sind. Beispielsweise kann für die Verwendung von Schlüsseln `keyCertSign` und erforderlich sein, dies ist `cRLSign` jedoch nicht `digitalSignature` der Fall. Im Gegensatz zu anderen Root-Passthrough-Zertifikatsvorlagen, die nicht leer sind, ermöglichen leere Stammzertifikatsvorlagen die Konfiguration der KU-Erweiterung, wobei KU für jeden der neun unterstützten Werte (`digitalSignature`,,`nonRepudiation`,,`keyEncipherment`,`dataEncipherment`, `keyAgreement` `keyCertSign` `cRLSign``encipherOnly`, und`decipherOnly`) stehen kann.

BlankRootCACertificate_ /V1 APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]

BlankRootCACertificate_ PathLen 0_ /V1-Definition APIPassthrough

Allgemeine Informationen zu leeren Root-CA-Vorlagen finden Sie unter. ???

BlankRootCACertificate_ PathLen 0_ /V1 APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 0`
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]

BlankRootCACertificate_ PathLen 1_ /V1-Definition APIPassthrough

Allgemeine Informationen zu leeren Root-CA-Vorlagen finden Sie unter. ???

BlankRootCACertificate_ PathLen 1_ /V1 APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 1`
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]

BlankRootCACertificate_ PathLen 2_ /V1-Definition APIPassthrough

Allgemeine Informationen zu leeren Root-CA-Vorlagen finden Sie unter. ???

BlankRootCACertificate_ PathLen 2_ /V1 APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]

X509v3-Parameter	Wert
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 2`
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]

BlankRootCACertificate_ PathLen 3_ /V1-Definition APIPassthrough

Allgemeine Informationen zu leeren Root-CA-Vorlagen finden Sie unter. ???

BlankRootCACertificate_ PathLen 3_ /V1 APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 3`
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]

Untergeordnete CACertificate _ 0/V1-Definition PathLen

Diese Vorlage wird verwendet, um untergeordnete CA-Zertifikate mit einer Pfadlänge von auszustellen. `0` CA-Zertifikate enthalten eine Erweiterung für wichtige grundlegende Einschränkungen, wobei das CA-Feld so festgelegt ist, dass `TRUE` zum Ausstellen von CA-Zertifikaten verwendet werden kann. Die erweiterte Schlüsselverwendung ist nicht enthalten, wodurch verhindert wird, dass das CA-Zertifikat als TLS-Client- oder Serverzertifikat verwendet wird.

Mehr über Zertifizierungspfade erfahren Sie auf der Seite mit Informationen über das Festlegen von Längenbeschränkungen für den Zertifizierungspfad.

Untergeordnetes _ 0/V1 CACertificate PathLen

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]

X509v3-Parameter	Wert
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 0`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

*CRL-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt werden, wenn die Zertifizierungsstelle mit aktivierter CRL-Generierung konfiguriert ist.

Untergeordnete _ 0_ /V1-Definition CACertificate PathLen APICSRPassthrough

Diese Vorlage erweitert Subordinate CACertificate _ PathLen 0/V1 um die Unterstützung von API- und CSR-Passthrough-Werten.

Subordinate _ 0_ /V1 CACertificate PathLen APICSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 0`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen

X509v3-Parameter	Wert
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

Untergeordnete CACertificate _ 0_ /V1-Definition PathLen APIPassthrough

Diese Vorlage erweitert Subordinate CACertificate _ PathLen 0/V1 um die Unterstützung von API-Passthrough-Werten.

Subordinate _ 0_ /V1 CACertificate PathLen APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 0`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

Untergeordnete CACertificate _ 0_ /V1-Definition PathLen CSRPassthrough

Diese Vorlage ist identisch mit der `SubordinateCACertificate_PathLen0` Vorlage, mit einem Unterschied: In dieser Vorlage werden zusätzliche Erweiterungen aus der Certificate Signing Request

(CSR) an das Zertifikat AWS Private CA übergeben, wenn die Erweiterungen nicht in der Vorlage angegeben sind. Erweiterungen, die in der Vorlage angegeben sind, haben stets Vorrang vor Erweiterungen in der CSR.

> **Note**
>
> Eine CSR, die benutzerdefinierte zusätzliche Erweiterungen enthält, muss außerhalb von erstellt werden. AWS Private CA

Untergeordnet CACertificate _ 0_ /V1 PathLen CSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 0`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlistengenerierung konfiguriert ist.

Untergeordnete _ 1/V1-Definition CACertificate PathLen

Diese Vorlage wird verwendet, um untergeordnete CA-Zertifikate mit einer Pfadlänge von auszustellen. 1 CA-Zertifikate enthalten eine wichtige Erweiterung Basic Constraints, bei der das CA-Feld auf gesetzt ist, `TRUE` um anzugeben, dass das Zertifikat zur Ausstellung von CA-Zertifikaten

verwendet werden kann. Die erweiterte Schlüsselverwendung ist nicht enthalten, wodurch verhindert wird, dass das CA-Zertifikat als TLS-Client- oder Serverzertifikat verwendet wird.

Mehr über Zertifizierungspfade erfahren Sie auf der Seite mit Informationen über das Festlegen von Längenbeschränkungen für den Zertifizierungspfad.

Untergeordnet _ 1/V1 CACertificate PathLen

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 1`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlistengenerierung konfiguriert ist.

Untergeordnete CACertificate _ PathLen 1_ /V1-Definition APICSRPassthrough

Diese Vorlage erweitert Subordinate CACertificate _ PathLen 1/V1 um die Unterstützung von API- und CSR-Passthrough-Werten.

Subordinate _ 1_ /V1 CACertificate PathLen APICSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]

X509v3-Parameter	Wert
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 1`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

Untergeordnete CACertificate _ 1_ /V1-Definition PathLen APIPassthrough

Diese Vorlage erweitert Subordinate CACertificate _ PathLen 0/V1 um die Unterstützung von API-Passthrough-Werten.

Subordinate _ 1_ /V1 CACertificate PathLen APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 1`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

Untergeordnete CACertificate _ 1_ /V1-Definition PathLen CSRPassthrough

Diese Vorlage ist mit der `SubordinateCACertificate_PathLen1` Vorlage identisch, mit einem Unterschied: In dieser Vorlage werden zusätzliche Erweiterungen aus der Certificate Signing Request (CSR) an das Zertifikat AWS Private CA übergeben, wenn die Erweiterungen nicht in der Vorlage angegeben sind. Erweiterungen, die in der Vorlage angegeben sind, haben stets Vorrang vor Erweiterungen in der CSR.

> **Note**
>
> Eine CSR, die benutzerdefinierte zusätzliche Erweiterungen enthält, muss außerhalb von erstellt werden. AWS Private CA

Untergeordnet CACertificate _ 1_ /V1 PathLen CSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 1`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlistengenerierung konfiguriert ist.

Untergeordnete _ 2/V1-Definition CACertificate PathLen

Diese Vorlage wird verwendet, um untergeordnete CA-Zertifikate mit einer Pfadlänge von 2 auszustellen. CA-Zertifikate enthalten eine wichtige Erweiterung Basic Constraints, bei der das CA-Feld auf gesetzt ist, `TRUE` um anzugeben, dass das Zertifikat zur Ausstellung von CA-Zertifikaten verwendet werden kann. Die erweiterte Schlüsselverwendung ist nicht enthalten, wodurch verhindert wird, dass das CA-Zertifikat als TLS-Client- oder Serverzertifikat verwendet wird.

Mehr über Zertifizierungspfade erfahren Sie auf der Seite mit Informationen über das Festlegen von Längenbeschränkungen für den Zertifizierungspfad.

Untergeordnet _ 2/V1 CACertificate PathLen

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 2`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlistengenerierung konfiguriert ist.

Untergeordnete CACertificate _ PathLen 2_ /V1-Definition APICSRPassthrough

Diese Vorlage erweitert Subordinate CACertificate _ PathLen 2/V1 um die Unterstützung von API- und CSR-Passthrough-Werten.

Subordinate _ 2_ /V1 CACertificate PathLen APICSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 2`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

Untergeordnete CACertificate _ 2_ /V1-Definition PathLen APIPassthrough

Diese Vorlage erweitert Subordinate CACertificate _ PathLen 2/V1 um die Unterstützung von API-Passthrough-Werten.

Subordinate _ 2_ /V1 CACertificate PathLen APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 2`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]

X509v3-Parameter	Wert
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

Untergeordnete CACertificate _ 2_ /V1-Definition PathLen CSRPassthrough

Diese Vorlage ist identisch mit der `SubordinateCACertificate_PathLen2` Vorlage, mit einem Unterschied: In dieser Vorlage werden zusätzliche Erweiterungen aus der Certificate Signing Request (CSR) an das Zertifikat AWS Private CA übergeben, wenn die Erweiterungen nicht in der Vorlage angegeben sind. Erweiterungen, die in der Vorlage angegeben sind, haben stets Vorrang vor Erweiterungen in der CSR.

> **Note**
>
> Eine CSR, die benutzerdefinierte zusätzliche Erweiterungen enthält, muss außerhalb von erstellt werden. AWS Private CA

Untergeordnetes CACertificate _ 2_ /V1 PathLen CSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 2`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]

X509v3-Parameter	Wert
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlistengenerierung konfiguriert ist.

Untergeordnete _ 3/V1-Definition CACertificate PathLen

Diese Vorlage wird verwendet, um untergeordnete CA-Zertifikate mit einer Pfadlänge von 3 auszustellen. CA-Zertifikate enthalten eine wichtige Erweiterung Basic Constraints, bei der das CA-Feld auf gesetzt ist, `TRUE` um anzugeben, dass das Zertifikat zur Ausstellung von CA-Zertifikaten verwendet werden kann. Die erweiterte Schlüsselverwendung ist nicht enthalten, wodurch verhindert wird, dass das CA-Zertifikat als TLS-Client- oder Serverzertifikat verwendet wird.

Mehr über Zertifizierungspfade erfahren Sie auf der Seite mit Informationen über das Festlegen von Längenbeschränkungen für den Zertifizierungspfad.

Untergeordnet _ 3/V1 CACertificate PathLen

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 3`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen

X509v3-Parameter	Wert
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlistengenerierung konfiguriert ist.

Untergeordnete CACertificate _ PathLen 3_ /V1-Definition APICSRPassthrough

Diese Vorlage erweitert Subordinate CACertificate _ PathLen 3/V1 um die Unterstützung von API- und CSR-Passthrough-Werten.

Subordinate _ 3_ /V1 CACertificate PathLen APICSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 3`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

Untergeordnete CACertificate _ 3_ /V1-Definition PathLen APIPassthrough

Diese Vorlage erweitert Subordinate CACertificate _ PathLen 3/V1 um die Unterstützung von API-Passthrough-Werten.

Subordinate _ 3_ /V1 CACertificate PathLen APIPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Passthrough von API oder CSR]
Betreff	[Passthrough von API oder CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 3`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

Untergeordnete CACertificate _ 3_ /V1-Definition PathLen CSRPassthrough

Diese Vorlage ist identisch mit der `SubordinateCACertificate_PathLen3` Vorlage, mit einem Unterschied: In dieser Vorlage werden zusätzliche Erweiterungen aus der Certificate Signing Request (CSR) an das Zertifikat AWS Private CA übergeben, wenn die Erweiterungen nicht in der Vorlage angegeben sind. Erweiterungen, die in der Vorlage angegeben sind, haben stets Vorrang vor Erweiterungen in der CSR.

> **Note**
>
> Eine CSR, die benutzerdefinierte zusätzliche Erweiterungen enthält, muss außerhalb von erstellt werden. AWS Private CA

Untergeordnet CACertificate _ 3_ /V1 PathLen CSRPassthrough

X509v3-Parameter	Wert
Alternativer Name des Betreffs	[Weiterleitung von CSR]
Betreff	[Passthrough von CSR]
Grundlegende Einschränkungen	kritisch, `CA:TRUE`, `pathlen: 3`
Schlüssel-ID der Behörde	[SKI aus dem CA-Zertifikat]
Schlüssel-ID für den Betreff	[Abgeleitet von CSR]
Schlüsselnutzung	Kritisch, digitale Signatur`keyCertSign` , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough aus der CA-Konfiguration oder CSR]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlistengenerierung konfiguriert ist.

Sicherheit in AWS Private Certificate Authority

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für gelten AWS Private Certificate Authority, finden Sie AWS-Services unter Umfang nach Compliance-ProgrammenAWS-Services und unter .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Nutzung anwenden können AWS Private CA. In den folgenden Themen erfahren Sie, wie Sie die Konfiguration vornehmen AWS Private CA , um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere verwenden können AWS-Services , die Ihnen bei der Überwachung und Sicherung Ihrer AWS Private CA Ressourcen helfen.

Topics

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

Für den Zugriff auf AWS Private CA sind Anmeldeinformationen erforderlich, mit denen Sie Ihre Anfragen authentifizieren AWS können. In den folgenden Themen finden Sie Einzelheiten dazu, wie Sie AWS Identity and Access Management (IAM) verwenden können, um Ihre privaten Zertifizierungsstellen (CAs) zu schützen, indem Sie kontrollieren, wer darauf zugreifen kann.

In AWS Private CA ist die primäre Ressource, mit der Sie arbeiten, eine Zertifizierungsstelle (CA). Jede private CA, die Sie besitzen oder kontrollieren, wird durch einen Amazon-Ressourcennamen (ARN) identifiziert, der das folgende Format hat.

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

Ein Ressourcenbesitzer ist die Hauptentität des AWS Kontos, in dem eine AWS Ressource erstellt wird. Die Funktionsweise wird anhand der folgenden Beispiele deutlich.

- Wenn Sie Ihre Anmeldeinformationen verwenden, Root-Benutzer des AWS-Kontos um eine private Zertifizierungsstelle zu erstellen, ist Ihr AWS Konto Eigentümer der Zertifizierungsstelle.

  > **Important**
  >
  > - Wir raten davon ab, eine Root-Benutzer des AWS-Kontos zur Erstellung zu verwenden CAs.
  > - Wir empfehlen dringend, bei jedem Zugriff die Multi-Faktor-Authentifizierung (MFA) zu verwenden. AWS Private CA

- Wenn Sie in Ihrem AWS Konto einen IAM-Benutzer erstellen, können Sie diesem Benutzer die Erlaubnis erteilen, eine private CA zu erstellen. Jedoch besitzt das Konto, zu dem dieser Benutzer gehört, die Zertifizierungsstelle.
- Wenn Sie in Ihrem AWS Konto eine IAM-Rolle erstellen und ihr die Berechtigung zum Erstellen einer privaten CA erteilen, kann jeder, der diese Rolle übernehmen kann, die CA erstellen. Jedoch besitzt das Konto, zu dem diese Rolle gehört, die private Zertifizierungsstelle.

Eine Berechtigungsrichtlinie beschreibt, wer Zugriff auf welche Objekte hat. Im folgenden Abschnitt werden die verfügbaren Optionen zum Erstellen von Berechtigungsrichtlinien erläutert.

> **Note**
>
> In dieser Dokumentation wird die Verwendung von IAM im Kontext von beschrieben. AWS Private CA Er enthält keine detaillierten Informationen über den IAM-Service. Eine umfassende IAM-Dokumentation finden Sie im IAM User Guide. Informationen über die IAM-Richtliniensyntax und Beschreibungen sind in der AWS IAM Policy Reference enthalten.

AWS Private CA API-Operationen und -Berechtigungen

Wenn Sie Zugriffskontroll- und Berechtigungsrichtlinien einrichten, die Sie an eine IAM-Identität anhängen möchten (identitätsbasierte Richtlinien), verwenden Sie die folgende Tabelle als Referenz. In der ersten Spalte der Tabelle sind die einzelnen AWS Private CA API-Operationen aufgeführt. Sie geben Aktionen in einem `Action`-Element der Richtlinie an. Die restlichen Spalten enthalten die zusätzlichen Informationen.

AWS Private CA API-Operationen	Erforderliche Berechtigungen	Ressourcen
CreateCertificateAuthority	`acm-pca:CreateCertificateAuthority` `acm-pca:TagCertificateAuthority` (Nur erforderlich, wenn eine CA mit Tags erstellt wird.)	`arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566`
CreateCertificateAuthorityAuditReport	`acm-pca:CreateCertificateAuthorityAuditReport`	`arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566`
CreatePermission	`acm-pca:CreatePermission`	`arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-`

AWS Private CA API-Operationen	Erforderliche Berechtigungen	Ressourcen
		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*

AWS Private CA API-Operationen	Erforderliche Berechtigungen	Ressourcen
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*

AWS Private CA API-Operationen	Erforderliche Berechtigungen	Ressourcen
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	–
ListPermissions	acm-pca:ListPermissions	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListTags	acm-pca:ListTags	–
PutPolicy	acm-pca:PutPolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

AWS Private CA API-Operationen	Erforderliche Berechtigungen	Ressourcen
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*

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter Eine Rolle für einen externen Identitätsanbieter (Verbund) erstellen im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter Eine Rolle für einen IAM-Benutzer erstellen im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

AWS verwaltete Richtlinien

AWS Private CA umfasst eine Reihe vordefinierter AWS verwalteter Richtlinien für AWS Private CA Administratoren, Benutzer und Prüfer. Diese Richtlinien zu verstehen, hilft Ihnen bei der Implementierung von Kundenverwaltete Richtlinien.

Wählen Sie eine der unten aufgeführten Richtlinien aus, um Details und einen Beispielrichtliniencode zu sehen.

AWSPrivateCAFullZugriff

Gewährt uneingeschränkte administrative Kontrolle.

Eine JSON-Liste der Richtliniendetails finden Sie unter AWSPrivateCAFullAccess.

AWSPrivateCAReadNur

Gewährt Zugriff, der auf schreibgeschützte API-Operationen beschränkt ist.

Eine JSON-Liste der Richtliniendetails finden Sie unter AWSPrivate CARead Only.

AWSPrivateCAPrivilegedBenutzer

Gewährt die Möglichkeit, CA-Zertifikate auszustellen und zu widerrufen. Diese Richtlinie verfügt über keine weiteren administrativen Funktionen und bietet nicht die Möglichkeit, Endentitätszertifikate auszustellen. Berechtigungen und die Benutzer-Richtlinie schließen sich gegenseitig aus.

Eine JSON-Liste der Richtliniendetails finden Sie unter AWSPrivateCAPrivilegedBenutzer.

AWSPrivateCAUser

Gewähren Sie die Möglichkeit, Endentitätszertifikate auszustellen und zu widerrufen. Diese Richtlinie verfügt über keine administrativen Funktionen und bietet nicht die Möglichkeit, CA-Zertifikate auszustellen. Berechtigungen schließen sich mit der PrivilegedUserRichtlinie gegenseitig aus.

Eine JSON-Liste der Richtliniendetails finden Sie unter AWSPrivateCAUser.

AWSPrivateCAAuditor

Gewähren Sie Zugriff auf schreibgeschützte API-Operationen und die Erlaubnis, einen CA-Prüfbericht zu erstellen.

Eine JSON-Liste der Richtliniendetails finden Sie unter. AWSPrivateCAAuditor

AWSPrivateCAConnectorForKubernetesPolicy

Gewährt grundlegende Berechtigungen für den AWS Private CA Connector für Kubernetes.

Eine JSON-Liste der Richtliniendetails finden Sie unter. AWSPrivateCAConnectorForKubernetesPolicy

Aktualisierungen der AWS verwalteten Richtlinien für AWS Private CA

In der folgenden Tabelle finden Sie Details zu Aktualisierungen der AWS verwalteten Richtlinien, die AWS Private CA seit Beginn der Nachverfolgung dieser Änderungen durch den Dienst vorgenommen wurden. Abonnieren Sie den RSS-Feed auf der Dokumentverlauf Seite AWS Private CA, um automatische Benachrichtigungen über alle Änderungen an zu erhalten.

Verwaltete Richtlinienänderungen

Änderungen	Beschreibung	Date
Neue Richtlinie: AWS PrivatCAConnector ForKubernetesPolicy	Neue verwaltete Richtlinie für die Verwendung mit AWS Private CA Connector for Kubernetes eingeführt.	19. Mai 2025
AWS CAPrivilegedPrivatnutzer und AWS Privatnutzer CAUser — Aktualisierte Richtlinie	Ersetzt `StringLike` durch `ArnLike`, und `StringNotLike` durch `ArnNotLike` .	22. Januar 2025

Änderungen	Beschreibung	Date
	Die Vorlage arn wurde aktualisiert und enthält nun auch `arn:aws:acm-pca:::template` Platzhalt erarn:aws:acm-pca:*:*:template .	
Neue Richtliniennamen: • `AWS PrivateCA FullAccess` • `AWS PrivateCA ReadOnly` • `AWS PrivateCA PrivilegedUser` • `AWS PrivateCAAuditor` • `AWS PrivateCAUser`	Die Präfixe für Richtlini ennamen wurden von `AWS CertificateManager PrivateCA` zu `AWS PrivateCA` geändert. Die Funktionalität bleibt unverändert.	13. Februar 2023

Kundenverwaltete Richtlinien

Es hat sich bewährt, Sie nicht zu verwenden AWS, Root-Benutzer des AWS-Kontos um mit anderen zu interagieren. AWS Private CA Verwenden Sie stattdessen AWS Identity and Access Management (IAM), um einen IAM-Benutzer, eine IAM-Rolle oder einen Verbundbenutzer zu erstellen. Erstellen Sie eine Administratorgruppe und fügen Sie sich dieser hinzu. Melden Sie sich dann als Administrator an. Fügen Sie der Gruppe bei Bedarf weitere Benutzer hinzu.

Eine weitere bewährte Methode besteht darin, eine vom Kunden verwaltete IAM-Richtlinie zu erstellen, die Sie Benutzern zuweisen können. Vom Kunden verwaltete Richtlinien sind eigenständige Richtlinien auf Identitätsbasis, die Sie erstellen und an mehrere Benutzer, Gruppen oder Rollen in Ihrem AWS -Konto anfügen können. Eine solche Richtlinie sorgt dafür, dass Benutzer nur die von Ihnen angegebenen AWS Private CA -Aktionen ausführen dürfen.

Das folgende Beispiel für eine vom Kunden verwaltete Richtlinie ermöglicht Benutzern das Erstellen eines CA-Auditberichts. Dies ist lediglich ein Beispiel. Sie können alle gewünschten AWS Private CA Operationen auswählen. Weitere Beispiele finden Sie unter Eingebundene Richtlinien.

So erstellen Sie eine vom Kunden verwaltete Richtlinie

1. Melden Sie sich mit den Anmeldeinformationen eines AWS Administrators bei der IAM-Konsole an.
2. Wählen Sie im Navigationsbereich der Konsole Policies (Richtlinien) aus.
3. Wählen Sie Richtlinie erstellen aus.
4. Wählen Sie den Tab JSON.
5. Kopieren Sie die folgende Richtlinie, und fügen Sie sie in den Editor ein.

   JSON

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

6. Wählen Sie Richtlinie prüfen.
7. Geben Sie bei Name `PcaListPolicy` ein.
8. (Optional) Geben Sie eine Beschreibung ein.
9. Wählen Sie Richtlinie erstellen aus.

Ein Administrator kann die Richtlinie an jeden IAM-Benutzer anhängen, um die AWS Private CA Aktionen einzuschränken, die der Benutzer ausführen kann. Informationen zur Anwendung einer Berechtigungsrichtlinie finden Sie unter Ändern von Berechtigungen für einen IAM-Benutzer im IAM-Benutzerhandbuch.

Eingebundene Richtlinien

Eingebundene Richtlinien sind Richtlinien, die Sie erstellen und verwalten und die Sie direkt in einen Benutzer, eine Gruppe oder Rolle einbetten. Die folgenden Richtlinienbeispiele zeigen, wie Sie Berechtigungen zur Durchführung von AWS Private CA Aktionen zuweisen. Allgemeine Informationen

zu Inline-Richtlinien finden Sie unter Arbeiten mit Inline-Richtlinien im IAM-Benutzerhandbuch. Sie können die AWS-Managementkonsole, AWS Command Line Interface (AWS CLI) oder die IAM-API verwenden, um Inline-Richtlinien zu erstellen und einzubetten.

> **Important**
>
> Wir empfehlen dringend, bei jedem Zugriff die Multi-Faktor-Authentifizierung (MFA) zu verwenden. AWS Private CA

Themen

Das Angebot ist privat CAs

Die folgende Richtlinie ermöglicht es einem Benutzer, alle privaten Daten CAs in einem Konto aufzulisten.

JSON

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

```
   ]
}
```

Ein privates CA-Zertifikat wird abgerufen

Mit der folgenden Richtlinie können Benutzer ein bestimmtes privates Zertifizierungsstellenzertifikat abrufen.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:GetCertificateAuthorityCertificate",
      "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID/certificate/certificate_ID"
   }
}
```

Ein privates CA-Zertifikat importieren

Mit der folgenden Richtlinie kann ein Benutzer ein privates Zertifizierungsstellenzertifikat importieren.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:ImportCertificateAuthorityCertificate",
      "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID/certificate/certificate_ID"
   }
}
```

Löschen einer privaten CA

Mit der folgenden Richtlinie können Benutzer eine bestimmte private Zertifizierungsstelle löschen.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:DeleteCertificateAuthority",
      "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID/certificate/certificate_ID"   }
}
```

Tag-on-create: Hinzufügen von Tags an eine CA zum Zeitpunkt der Erstellung

Die folgende Richtlinie ermöglicht es einem Benutzer, während der Erstellung der CA Tags anzuwenden.

JSON

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

Tag-on-create: Eingeschränktes Tagging

Die folgende tag-on-create Richtlinie verhindert die Verwendung des Schlüssel-Wert-Paars Environment=Prod bei der Erstellung einer Zertifizierungsstelle. Das Markieren mit anderen Schlüssel-Wert-Paaren ist zulässig.

JSON

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

Steuern des Zugriffs auf Private CA mithilfe von Tags

Die folgende Richtlinie erlaubt den Zugriff nur CAs mit dem Schlüssel-Wert-Paar Environment=. PreProd Sie erfordert außerdem, dass new dieses Tag CAs einschließt.

JSON

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

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

Nur-Lese-Zugriff auf AWS Private CA

Mit der folgenden Richtlinie können Benutzer private Zertifizierungsstellen beschreiben und auflisten sowie das private CA-Zertifikat und die Zertifikatkette abrufen.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":{
       "Effect":"Allow",
       "Action":[
          "acm-pca:DescribeCertificateAuthority",
          "acm-pca:DescribeCertificateAuthorityAuditReport",
          "acm-pca:ListCertificateAuthorities",
          "acm-pca:ListTags",
          "acm-pca:GetCertificateAuthorityCertificate",
          "acm-pca:GetCertificateAuthorityCsr",
          "acm-pca:GetCertificate"
       ],
       "Resource":"*"
    }
}
```

Voller Zugriff auf AWS Private CA

Die folgende Richtlinie ermöglicht es einem Benutzer, jede AWS Private CA Aktion auszuführen.

JSON

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

Bewährte Sicherheitsmethoden für den kontoübergreifenden Zugriff auf private CAs

Ein AWS Private CA Administrator kann eine CA mit Principals (Benutzern, Rollen usw.) in einer anderen AWS-Konto teilen. Wenn eine Freigabe empfangen und akzeptiert wurde, kann der Principal die CA verwenden, um mithilfe AWS Private CA unserer Ressourcen Endentitätszertifikate auszustellen. AWS Certificate Manager Der Prinzipal kann die CA verwenden, um untergeordnete CA-Zertifikate auszustellen mit. AWS Private CA

> **Important**
>
> Gebühren für ein Zertifikat, das in einem kontoübergreifenden Szenario ausgestellt wurde, werden dem Konto in Rechnung gestellt, das das AWS Zertifikat ausstellt.

Um den Zugriff auf eine Zertifizierungsstelle gemeinsam zu nutzen, können AWS Private CA Administratoren eine der folgenden Methoden wählen:

- Verwenden Sie AWS Resource Access Manager (RAM), um die CA als Ressource mit einem Principal in einem anderen Konto oder mit zu teilen AWS Organizations. RAM ist eine Standardmethode für die gemeinsame Nutzung von AWS Ressourcen zwischen Konten. Weitere Informationen zu RAM finden Sie im AWS RAM Benutzerhandbuch. Weitere Informationen zu AWS Organizations finden Sie im AWS Organizations -Benutzerhandbuch.
- Verwenden Sie die AWS Private CA API oder CLI, um eine ressourcenbasierte Richtlinie an eine CA anzuhängen und so einem Principal in einem anderen Konto Zugriff zu gewähren. Weitere Informationen finden Sie unter Ressourcenbasierte Richtlinien.

Der Steuern Sie den Zugriff auf die private CA Abschnitt dieses Handbuchs enthält Workflows für die Gewährung des Zugriffs sowohl für Einzelkonten als auch für kontoübergreifende Szenarien. CAs

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind Berechtigungsrichtlinien, die Sie erstellen und manuell einer Ressource (in diesem Fall einer privaten Zertifizierungsstelle) und nicht einer Benutzeridentität oder Rolle zuordnen. Anstatt Ihre eigenen Richtlinien zu erstellen, können Sie auch AWS verwaltete Richtlinien für verwenden. AWS Private CA Durch AWS RAM die Anwendung einer ressourcenbasierten Richtlinie kann ein AWS Private CA Administrator den Zugriff auf eine Zertifizierungsstelle direkt oder über einen Benutzer mit einem anderen AWS Konto teilen. AWS Organizations Alternativ kann ein AWS Private CA Administrator die PCA APIs PutPolicy, GetPolicyund oder die entsprechenden AWS CLI Befehle put-policy DeletePolicy, get-policy und delete-policy verwenden, um ressourcenbasierte Richtlinien anzuwenden und zu verwalten.

Allgemeine Informationen zu ressourcenbasierten Richtlinien finden Sie unter Identitätsbasierte Richtlinien und Ressourcenbasierte Richtlinien und Steuern des Zugriffs mithilfe von Richtlinien.

Um die Liste der AWS verwalteten ressourcenbasierten Richtlinien für anzuzeigen AWS Private CA, navigieren Sie in der Konsole zur Bibliothek für verwaltete Berechtigungen und suchen Sie nach. AWS Resource Access Manager CertificateAuthority Wie bei jeder Richtlinie empfehlen wir, die Richtlinie vor ihrer Anwendung in einer Testumgebung anzuwenden, um sicherzustellen, dass sie Ihren Anforderungen entspricht.

AWS Certificate Manager (ACM) Benutzer mit kontenübergreifendem gemeinsamen Zugriff auf eine private Zertifizierungsstelle können verwaltete Zertifikate ausstellen, die von der Zertifizierungsstelle signiert sind. Kontoübergreifende Aussteller sind durch eine ressourcenbasierte Richtlinie eingeschränkt und haben nur Zugriff auf die folgenden Vorlagen für Endzertifikate:

- EndEntityCertificate/V1
- EndEntityClientAuthCertificate/V1
- EndEntityServerAuthCertificate/V1
- BlankEndEntityCertificate_ /V1 APIPassthrough
- BlankEndEntityCertificate_ /V1 APICSRPassthrough
- Untergeordnet _ 0/V1 CACertificate PathLen

Beispiele für Richtlinien

Dieser Abschnitt enthält Beispiele für kontoübergreifende Richtlinien für verschiedene Anforderungen. In allen Fällen wird das folgende Befehlsmuster verwendet, um eine Richtlinie anzuwenden:

```
$ 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
```

Zusätzlich zur Angabe des ARN einer CA gibt der Administrator eine AWS Konto-ID oder eine AWS Organizations ID an, der Zugriff auf die CA gewährt wird. Die JSON-Datei jeder der folgenden Richtlinien ist aus Gründen der Lesbarkeit als Datei formatiert, kann aber auch als Inline-CLI-Argumente bereitgestellt werden.

> **Note**
>
> Die unten aufgeführte Struktur der ressourcenbasierten JSON-Richtlinien muss genau eingehalten werden. Nur die ID-Felder für die Principals (die AWS Kontonummer oder die AWS Organisations-ID) und die CA ARNs können von Kunden konfiguriert werden.

1. Datei: policy1.json — Teilen des Zugriffs auf eine CA mit einem Benutzer in einem anderen Konto

   *555555555555*Ersetzen Sie es durch die AWS Konto-ID, die die CA gemeinsam nutzt.

   Ersetzen Sie für den Ressourcen-ARN Folgendes durch Ihre eigenen Werte:

   - *aws*- Die AWS Partition. Zum Beispiel`aws`, `aws-us-gov`aws-cn,, usw.
   - *us-east-1*- Die AWS Region, in der die Ressource verfügbar ist, z. `us-west-1` B.

- *111122223333*- Die AWS Konto-ID des Ressourcenbesitzers.
- *11223344-1234-1122-2233-112233445566*- Die Ressourcen-ID der Zertifizierungsstelle.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Sid": "ExampleStatementID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "555555555555"
            },
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListPermissions",
                "acm-pca:ListTags"
            ],
            "Resource": "arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID"
        },
        {
            "Sid": "ExampleStatementID2",
            "Effect": "Allow",
            "Principal": {
                "AWS": "555555555555"
            },
            "Action": [
                "acm-pca:IssueCertificate"
            ],
            "Resource": "arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID",
            "Condition": {
                "StringEquals": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
EndEntityCertificate/V1"
                }
            }
        }
    ]
}
```

2. Datei: policy2.json — Gemeinsamer Zugriff auf eine CA über AWS Organizations

   Durch die *o-a1b2c3d4z5* ID ersetzen. AWS Organizations

   Ersetzen Sie für den Ressourcen-ARN Folgendes durch Ihre eigenen Werte:

   - *aws*- Die AWS Partition. Zum Beispiel`aws`, `aws-us-gov`aws-cn,, usw.
   - *us-east-1*- Die AWS Region, in der die Ressource verfügbar ist, z. `us-west-1` B.
   - *111122223333*- Die AWS Konto-ID des Ressourcenbesitzers.
   - *11223344-1234-1122-2233-112233445566*- Die Ressourcen-ID der Zertifizierungsstelle.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleStatementID3",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "acm-pca:IssueCertificate",
            "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID",
            "Condition": {
                "StringEquals": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
EndEntityCertificate/V1",
                    "aws:PrincipalOrgID": "o-a1b2c3d4z5"
                },
                "StringNotEquals": {
                    "aws:PrincipalAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "ExampleStatementID4",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
```

```
                "acm-pca:ListPermissions",
                "acm-pca:ListTags"
            ],
            "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "o-a1b2c3d4z5"
                },
                "StringNotEquals": {
                    "aws:PrincipalAccount": "111122223333"
                }
            }
        }
    ]
}
```

Datenschutz in AWS Private Certificate Authority

Das Modell der AWS gemeinsamen Verantwortung und geteilter Verantwortung gilt für den Datenschutz in AWS Private Certificate Authority. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.

- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit der Konsole, der AWS Private CA API oder auf andere AWS-Services Weise arbeiten oder diese verwenden. AWS CLI AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Aufbewahrung und Einhaltung der Sicherheitsbestimmungen von AWS Private CA privaten Schlüsseln

Die privaten Schlüssel für private Daten CAs werden in AWS verwalteten Hardware-Sicherheitsmodulen (HSMs) gespeichert. Sie HSMs entsprechen den FIPS PUB 140-2 Level 3-Sicherheitsanforderungen für kryptografische Module.

Datenverschlüsselung im AWS Private CA Connector für Active Directory

AWS Private CA Connector for AD speichert Kundenkonfigurationsdaten zu Connectoren, Vorlagen, Verzeichnisregistrierungen, Dienstprinzipalnamen und Zugriffskontrolleinträgen für Vorlagengruppen. Diese Daten werden bei der Übertragung und im Ruhezustand verschlüsselt. Informationen zu Zertifikaten, die über Connector for AD ausgestellt wurden, können mithilfe der GetCertificateAktion in der AWS Private CA API abgerufen werden. Es werden keine Informationen zu den ausgestellten Zertifikaten oder zu dem Client oder Computer, der ein Zertifikat anfordert, gespeichert AWS.

Compliance-Validierung für AWS Private Certificate Authority

Externe Prüfer bewerten die Sicherheit und Einhaltung von Vorschriften im AWS Private Certificate Authority Rahmen mehrerer AWS Compliance-Programme. Hierzu zählen unter anderem SOC, PCI, FedRAMP und HIPAA.

Eine Liste der AWS Dienstleistungen im Rahmen bestimmter Compliance-Programme finden Sie unter AWS Dienstleistungen im Umfang nach Compliance-Programmen AWS-Services unter . Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS Private CA hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Für Organisationen, die ihre Amazon S3 S3-Buckets verschlüsseln müssen, wird in den folgenden Themen beschrieben, wie die Verschlüsselung für AWS Private CA Ressourcen konfiguriert wird:
  - Verschlüsseln von Auditberichten
  - Verschlüsseln Sie Ihre CRLs
- Schnellstartanleitungen zu Sicherheit und Compliance Schnellstartanleitungen zu — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte zur Implementierung von sicherheits- und Compliance-orientierten Basisumgebungen beschrieben. AWS
- Whitepaper „Architecting for HIPAA Security and Compliance“ — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-konforme Anwendungen entwickeln können. AWS
- AWS Compliance-Ressourcen — Diese Sammlung von Arbeitsmappen und Leitfäden kann auf Ihre Branche und Ihren Standort zutreffen.
- Bewertung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub CSPM— Dieser AWS Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus, sodass Sie überprüfen können AWS , ob Sie die Sicherheitsstandards und Best Practices der Branche einhalten.

Verwenden Sie Prüfberichte mit Ihrer privaten Zertifizierungsstelle

Sie können einen Auditbericht erstellen, der die Zertifikate auflistet, die ihre private CA ausgestellt oder widerrufen hat. Der Bericht wird in einem neuen oder bestehenden S3-Bucket gespeichert, den Sie bei der Eingabe angeben.

Weitere Informationen zum Hinzufügen von Verschlüsselungsschutz zu Ihren Audit-Berichten finden Sie unter Verschlüsseln Sie Ihre Auditberichte.

Die Prüfberichtsdatei hat den folgenden Pfad und Dateinamen. Der ARN für einen Amazon S3 S3-Bucket ist der Wert für`amzn-s3-demo-bucket`. `CA_ID`Dist die eindeutige Kennung einer ausstellenden Zertifizierungsstelle. `UUID`Dist die eindeutige Kennung eines Prüfberichts.

```
amzn-s3-demo-bucket/audit-report/CA_ID/UUID.[json|csv]
```

Sie können alle 30 Minuten einen neuen Bericht erstellen und diesen aus Ihrem Bucket herunterladen. Das folgende Beispiel zeigt einen CSV-getrennten Bericht.

```
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
```

Das folgende Beispiel zeigt einen Bericht im JSON-Format.

```
[
   {
      "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**
>
> Wenn ein Zertifikat AWS Certificate Manager erneuert wird, füllt der private CA-Prüfbericht das `requestedByServicePrincipal` Feld mit. `acm.amazonaws.com` Dies weist darauf hin, dass der AWS Certificate Manager Dienst die `IssueCertificate` Aktion der AWS Private CA API im Namen eines Kunden aufgerufen hat, um das Zertifikat zu verlängern.

Bereiten Sie einen Amazon S3 S3-Bucket für Auditberichte vor

> **Important**
>
> AWS Private CA unterstützt die Verwendung von Amazon S3 Object Lock nicht. Wenn Sie Object Lock für Ihren Bucket aktivieren, AWS Private CA ist es nicht möglich, Auditberichte in den Bucket zu schreiben.

Um Ihre Auditberichte zu speichern, müssen Sie einen Amazon S3 S3-Bucket vorbereiten. Weitere Informationen finden Sie unter Wie erstelle ich einen S3-Bucket?

Ihr S3-Bucket muss durch eine Berechtigungsrichtlinie geschützt sein, die AWS Private CA den Zugriff auf den von Ihnen angegebenen S3-Bucket und das Schreiben in diesen ermöglicht. Autorisierte Benutzer und Dienstprinzipale benötigen die `Put` Erlaubnis, Objekte im Bucket platzieren AWS Private CA zu dürfen, und die `Get` Erlaubnis, sie abzurufen. Wir empfehlen Ihnen, die unten aufgeführte Richtlinie anzuwenden, die den Zugriff sowohl auf ein AWS Konto als auch auf den ARN einer privaten CA einschränkt. Alternativ können Sie den Bedingungsschlüssel aws: SourceOrg ID verwenden, um den Zugriff auf eine bestimmte Organisation in einzuschränken. AWS Organizations Weitere Informationen zu Bucket-Richtlinien finden Sie unter Bucket-Richtlinien für Amazon Simple Storage Service.

Erstellen Sie einen Prüfbericht

Sie können einen Prüfbericht entweder über die Konsole oder über erstellen AWS CLI.

So erstellen Sie einen Audit-Bericht (Konsole)

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Konsole zu https://console.aws.amazon.com/acm-pca/Hause.
2. Wählen Sie auf der Seite Private Zertifizierungsstellen Ihre private Zertifizierungsstelle aus der Liste aus.
3. Klicken Sie im Menü Aktionen auf Audit-Bericht generieren.
4. Finden Sie unter Ziel des Prüfberichts die Option Neuen S3-Bucket erstellen? , wählen Sie Ja und geben Sie einen eindeutigen Bucket-Namen ein, oder wählen Sie Nein und wählen Sie einen vorhandenen Bucket aus der Liste aus.

   Wenn Sie Ja wählen, AWS Private CA wird die Standardrichtlinie erstellt und an Ihren Bucket angehängt. Die Standardrichtlinie beinhaltet einen `aws:SourceAccount` Bedingungsschlüssel,

der den Zugriff auf ein bestimmtes AWS Konto beschränkt. Wenn Sie den Zugriff weiter einschränken möchten, können Sie der Richtlinie weitere Bedingungsschlüssel hinzufügen, wie im vorherigen Beispiel.

Wenn Sie Nein wählen, müssen Sie Ihrem Bucket eine Richtlinie hinzufügen, bevor Sie einen Prüfbericht erstellen können. Verwenden Sie das unter beschriebene RichtlinienmusterBereiten Sie einen Amazon S3 S3-Bucket für Auditberichte vor. Informationen zum Anhängen einer Richtlinie finden Sie unter Hinzufügen einer Bucket-Richtlinie mithilfe der Amazon S3 S3-Konsole.

5. Wählen Sie unter Ausgabeformat JSON für JavaScript Objektnotation oder CSV für kommagetrennte Werte.
6. Wählen Sie Generate audit report (Auditbericht generieren).

So erstellen Sie einen Audit-Bericht (AWS CLI)

1. Wenn Sie noch keinen S3-Bucket haben, den Sie verwenden können, erstellen Sie einen.
2. Hängen Sie eine Richtlinie an Ihren Bucket an. Verwenden Sie das unter beschriebene RichtlinienmusterBereiten Sie einen Amazon S3 S3-Bucket für Auditberichte vor. Informationen zum Anhängen einer Richtlinie finden Sie unter Hinzufügen einer Bucket-Richtlinie mithilfe der Amazon S3 S3-Konsole
3. Verwenden Sie den Befehl create-certificate-authority-audit-report, um den Auditbericht zu erstellen und ihn im vorbereiteten S3-Bucket zu platzieren.

```
$ aws acm-pca create-certificate-authority-audit-report \
--certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
--s3-bucket-name amzn-s3-demo-bucket \
--audit-report-response-format JSON
```

Rufen Sie einen Auditbericht ab

Verwenden Sie die Amazon S3 S3-Konsole, API, CLI oder SDK, um einen Prüfbericht zur Inspektion abzurufen. Weitere Informationen finden Sie unter Objekt herunterladen im Amazon Simple Storage Service-Benutzerhandbuch.

Verschlüsseln Sie Ihre Auditberichte

Sie können optional die Verschlüsselung für den Amazon S3 S3-Bucket konfigurieren, der Ihre Auditberichte enthält. AWS Private CA unterstützt zwei Verschlüsselungsmodi für Assets in S3:

- Automatische serverseitige Verschlüsselung mit von Amazon S3 verwalteten AES-256-Schlüsseln.
- Vom Kunden verwaltete Verschlüsselung unter Verwendung AWS Key Management Service und Konfiguration nach Ihren Spezifikationen AWS KMS key .

> **Note**
>
> AWS Private CA unterstützt nicht die Verwendung von Standard-KMS-Schlüsseln, die automatisch von S3 generiert werden.

In den folgenden Verfahren wird die Einrichtung der einzelnen Verschlüsselungsoptionen beschrieben.

So konfigurieren Sie die automatische Verschlüsselung

Führen Sie die folgenden Schritte aus, um die serverseitige S3-Verschlüsselung zu aktivieren.

1. Öffnen Sie die Amazon S3 S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie in der Buckets-Tabelle den Bucket aus, der Ihre AWS Private CA Ressourcen aufnehmen soll.
3. Wählen Sie auf der Seite für Ihren Bucket die Registerkarte Properties (Eigenschaften) aus.
4. Wählen Sie die Karte Default encryption (Standardverschlüsselung aus.
5. Wählen Sie Enable (Aktivieren) aus.
6. Wählen Sie den Amazon S3 S3-Schlüssel (SSE-S3).
7. Wählen Sie Save Changes.

So konfigurieren Sie die benutzerdefinierte Verschlüsselung

Führen Sie die folgenden Schritte aus, um die Verschlüsselung mit einem benutzerdefinierten Schlüssel zu aktivieren.

1. Öffnen Sie die Amazon S3 S3-Konsole unter https://console.aws.amazon.com/s3/.

2. Wählen Sie in der Buckets-Tabelle den Bucket aus, der Ihre AWS Private CA Ressourcen aufnehmen soll.
3. Wählen Sie auf der Seite für Ihren Bucket die Registerkarte Properties (Eigenschaften) aus.
4. Wählen Sie die Karte Default encryption (Standardverschlüsselung aus.
5. Wählen Sie Enable (Aktivieren) aus.
6. Wählen Sie AWS Key Management Service den Schlüssel (SSE-KMS).
7. Wählen Sie entweder Aus Ihren AWS KMS Schlüsseln auswählen oder AWS KMS key ARN eingeben.
8. Wählen Sie Save Changes.
9. (Optional) Wenn Sie noch keinen KMS-Schlüssel haben, erstellen Sie einen mit dem folgenden Befehl AWS CLI create-key:

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

   Die Ausgabe enthält die Schlüssel-ID und den Amazon-Ressourcennamen (ARN) des KMS-Schlüssels. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

   ```
   {
       "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. Mit den folgenden Schritten erteilen Sie dem AWS Private CA Dienstprinzipal die Erlaubnis, den KMS-Schlüssel zu verwenden. Standardmäßig sind alle KMS-Schlüssel privat. Nur der Besitzer der Ressource kann einen KMS-Schlüssel zum Verschlüsseln und Entschlüsseln von Daten verwenden. Der Ressourceninhaber kann jedoch anderen Benutzern und Ressourcen Zugriffsberechtigungen für den KMS-Schlüssel erteilen. Der Dienstprinzipal muss sich in derselben Region befinden, in der der KMS-Schlüssel gespeichert ist.

a. Speichern Sie zunächst die Standardrichtlinie für Ihren KMS-Schlüssel `policy.json` mit dem folgenden get-key-policyBefehl:

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

b. Öffnen Sie die Datei `policy.json` in einem Text-Editor. Wählen Sie eine der folgenden Richtlinienerklärungen aus und fügen Sie sie der vorhandenen Richtlinie hinzu.

Wenn Ihr Amazon S3 S3-Bucket-Key aktiviert ist, verwenden Sie die folgende Anweisung:

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

Wenn Ihr Amazon S3 S3-Bucket-Key deaktiviert ist, verwenden Sie die folgende Anweisung:

```
{
   "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. Wenden Sie abschließend die aktualisierte Richtlinie mit dem folgenden put-key-policyBefehl an:

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

Sicherheit der Infrastruktur in AWS Private Certificate Authority

Wird als verwalteter Service durch AWS globale Netzwerksicherheit geschützt. AWS Private Certificate Authority Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe für den Zugriff AWS Private CA über das Netzwerk. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

AWS Private CA VPC-Endpunkte ()AWS PrivateLink

Sie können eine private Verbindung zwischen Ihrer VPC und AWS Private CA durch die Konfiguration eines VPC-Schnittstellen-Endpunkts herstellen. Schnittstellen-Endpunkte werden von AWS

PrivateLinkeiner Technologie für den privaten Zugriff auf AWS Private CA API-Operationen unterstützt. AWS PrivateLink leitet den gesamten Netzwerkverkehr zwischen Ihrer VPC und AWS Private CA über das Amazon-Netzwerk weiter und verhindert so, dass er im offenen Internet offengelegt wird. Jeder VPC-Endpunkt wird durch eine oder mehrere Elastic Network-Schnittstellen mit privaten IP-Adressen in Ihren VPC-Subnetzen repräsentiert.

Der VPC-Endpunkt der Schnittstelle verbindet Ihre VPC direkt, AWS Private CA ohne dass ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder Direct Connect eine Verbindung erforderlich ist. Die Instances in Ihrer VPC benötigen für die Kommunikation mit der AWS Private CA -API keine öffentlichen IP-Adressen.

Für die Nutzung AWS Private CA über Ihre VPC müssen Sie von einer Instance aus eine Verbindung herstellen, die sich innerhalb der VPC befindet. Alternativ können Sie Ihr privates Netzwerk mit Ihrer VPC verbinden, indem Sie ein AWS Virtual Private Network (Site-to-Site VPN) oder Direct Connect verwenden. Weitere Informationen dazu Site-to-Site VPN finden Sie unter VPN-Verbindungen im Amazon VPC-Benutzerhandbuch. Informationen zu Direct Connect finden Sie unter Erstellen einer Verbindung im Direct Connect -Benutzerhandbuch.

AWS Private CA erfordert nicht die Verwendung von AWS PrivateLink, wir empfehlen es jedoch als zusätzliche Sicherheitsebene. Weitere Informationen zu AWS PrivateLink VPC-Endpunkten finden Sie unter Zugreifen auf Dienste über. AWS PrivateLink

Überlegungen zu AWS Private CA -VPC-Endpunkten

Bevor Sie Schnittstellen-VPC-Endpoints für einrichten AWS Private CA, sollten Sie die folgenden Überlegungen beachten:

- AWS Private CA unterstützt in einigen Availability Zones möglicherweise keine VPC-Endpunkte. Wenn Sie einen VPC-Endpunkt erstellen, überprüfen Sie zunächst die Unterstützung in der Managementkonsole. Nicht unterstützte Availability Zones sind mit „Service not supported in this Availability Zone“ gekennzeichnet.
- VPC-Endpunkte unterstützen keine regionsübergreifenden Anforderungen. Stellen Sie sicher, dass Sie Ihren Endpunkt innerhalb derselben Region erstellen, in der Sie Ihre API-Aufrufe an AWS Private CA ausgeben möchten.
- VPC-Endpunkte unterstützen nur von Amazon bereitgestelltes DNS über Amazon Route 53. Wenn Sie Ihre eigene DNS verwenden möchten, können Sie die bedingte DNS-Weiterleitung nutzen. Weitere Informationen finden Sie unter DHCP Options Sets im Amazon VPC-Benutzerhandbuch.

- Die Sicherheitsgruppe für den VPC-Endpunkt müssen eingehende Verbindungen auf Port 443 aus dem privaten Subnetz der VPC zulassen.

AWS Private CA Die API unterstützt derzeit VPC-Endpunkte in den folgenden Bereichen: AWS-Regionen

- US East (Ohio)
- USA Ost (Nord-Virginia)
- USA West (Nordkalifornien)
- USA West (Oregon)
- Africa (Cape Town)
- Asien-Pazifik (Hongkong)
- Asien-Pazifik (Hyderabad)
- Asien-Pazifik (Jakarta)
- Asien-Pazifik (Melbourne)
- Asien-Pazifik (Mumbai)
- Asien-Pazifik (Osaka)
- Asien-Pazifik (Seoul)
- Asien-Pazifik (Singapur)
- Asien-Pazifik (Sydney)
- Asien-Pazifik (Tokio)
- Canada (Central)
- Kanada West (Calgary)
- Europe (Frankfurt)
- Europa (Irland)
- Europa (London)
- Europa (Milan)
- Europa (Paris)
- Europa (Spain)
- Europa (Stockholm)

- Europa (Zürich)
- Israel (Tel Aviv)
- Middle East (Bahrain)
- Naher Osten (VAE)
- Südamerika (São Paulo)

Erstellung der VPC-Endpoints für AWS Private CA

Sie können einen VPC-Endpunkt für den AWS Private CA Service entweder mit der VPC-Konsole unter https://console.aws.amazon.com/vpc/oder mit dem erstellen. AWS Command Line Interface Weitere Informationen finden Sie im Verfahren Creating an Interface Endpoint im Amazon VPC-Benutzerhandbuch. AWS Private CA unterstützt Aufrufe aller API-Operationen in Ihrer VPC.

Wenn Sie private DNS-Hostnamen für den Endpunkt aktiviert haben, wird der AWS Private CA Standardendpunkt jetzt in Ihren VPC-Endpunkt aufgelöst. Eine umfassende Liste der Standard-Serviceendpunkte finden Sie unter Service-Endpunkte und Kontingente.

Wenn Sie keine privaten DNS-Hostnamen aktiviert haben, stellt Amazon VPC einen DNS-Endpunktnamen bereit, den Sie im folgenden Format verwenden können:

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

> **Note**
>
> Der Wert *region* steht für die Regionskennung für eine AWS Region, die von unterstützt wird AWS Private CA, z. B. `us-east-2` für die Region USA Ost (Ohio). Eine Liste von AWS Private CA finden Sie unter AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.

Weitere Informationen finden Sie unter AWS Private CA VPC-Endpunkte (AWS PrivateLink) im Amazon VPC-Benutzerhandbuch.

Erstellen einer VPC-Endpunktrichtlinie für AWS Private CA

Sie können eine Richtlinie für Amazon VPC-Endpunkte erstellen AWS Private CA , um Folgendes anzugeben:

- Der Prinzipal, der die Aktionen ausführen kann
- Aktionen, die ausgeführt werden können
- Ressourcen, für die Aktionen ausgeführt werden können

Weitere Informationen finden Sie unter Controlling Access to Services with VPC Endpoints im Amazon VPC-Leitfaden.

Beispiel — VPC-Endpunktrichtlinie für Aktionen AWS Private CA

Wenn sie an einen Endpunkt angehängt ist, gewährt die folgende Richtlinie allen Prinzipalen Zugriff auf die AWS Private CA Aktionen`IssueCertificate`,`DescribeCertificateAuthority`,`GetCertificate`, `GetCertificateAuthorityCertificateListPermissions`, und. `ListTags` Die Ressource in jedem Abschnitt ist eine private Zertifizierungsstelle. Im ersten Abschnitt wird die Erstellung von Endentitätszertifikaten unter Verwendung der angegebenen privaten Zertifizierungsstelle und Zertifikatvorlage autorisiert. Wenn Sie die verwendete Vorlage nicht kontrollieren möchten, wird der Abschnitt `Condition` nicht benötigt. Wenn Sie diesen jedoch entfernen, können alle Prinzipale CA-Zertifikate sowie Endentitätszertifikate erstellen.

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

Unterstützung für Dual-Stack-Endpunkte

AWS Private Certificate Authority bietet einen öffentlichen Dual-Stack-Endpunkt, der IPv4 sowohl IPv6 Clients als auch unterstützt. Ein Dual-Stack-Endpunkt ermöglicht es Clients, mit ihnen zu kommunizieren, indem sie entweder IPv4 oder IPv6 Adressen AWS Private CA verwenden. AWS Private CA für Active Directory und AWS Private CA Connector für SCEP unterstützen auch Dual-Stack-Endpunkte.

Der öffentliche AWS Private CA Dual-Stack-Endpunkt `https://acm-pca.your-region.api.aws` unterstützt sowohl Clients als auch. IPv4 IPv6 AWS Private CA ist auch privat über IPv4 und IPv6 von Ihrer Virtual Private Cloud (VPC) aus zugänglich. AWS PrivateLink Weitere Informationen zum Erstellen von VPC-Endpunkten mit privater Schnittstelle für finden Sie AWS Private CA unter. AWS Private CA VPC-Endpunkte ()AWS PrivateLink

Weitere Informationen finden Sie in den folgenden Ressourcen:

- IP-Adressierung für Ihre VPCs und Subnetze
- IPv6 Unterstützung für Ihre VPC

IPv6 Adressen in IAM verwenden und AWS Private CA

Stellen Sie vor dem Zugriff sicher IPv6, AWS Private Certificate Authority dass alle IAM-Richtlinien, die IP-Adressbeschränkungen enthalten, so aktualisiert wurden, dass sie auch IPv6 Adressbereiche enthalten. IP-basierte Richtlinien, die nicht für den Umgang mit IPv6 Adressen aktualisiert wurden,

können dazu führen, dass Clients fälschlicherweise Zugriff verlieren oder erhalten, wenn sie sie verwenden IPv6. Weitere Informationen zur AWS Private CA Dual-Stack-Unterstützung finden Sie unterUnterstützung für Dual-Stack-Endpunkte.

> **Important**
>
> Diese Anweisungen lassen keine Aktionen zu. Verwenden Sie diese Anweisungen in Kombination mit anderen Anweisungen, die bestimmte Aktionen zulassen.

Die folgende Anweisung verweigert ausdrücklich den Zugriff auf alle AWS Private CA Berechtigungen für Anfragen, die aus dem `192.0.2.*` IPv4 Adressbereich stammen. IP-Adressen, die sich außerhalb dieses Bereichs befinden, werden nicht ausdrücklich verweigert AWS Private CA . Da sich alle IPv6 Adressen außerhalb des verweigerten Bereichs befinden, verweigert diese Anweisung nicht ausdrücklich AWS Private CA Berechtigungen für IPv6 Adressen.

```
{
    "Sid": "DenyPrivateCAPermissions",
    "Effect": "Deny",
    "Action": [
        "acm-pca:*"
    ],
    "Resource": "*",
    "Condition": {
        "NotIpAddress": {
            "aws:SourceIp": [
                "192.0.2.0/24"
            ]
        }
    }
}
```

Sie können das `Condition` Element so ändern, dass sowohl IPv4 (`192.0.2.0/24`) als auch IPv6 (`2001:db8::/32`) Adressbereiche verweigert werden, wie im folgenden Beispiel gezeigt:

```
{
    "Sid": "DenyPrivateCAPermissions",
    "Effect": "Deny",
    "Action": [
```

```
        "acm-pca:*"
    ],
    "Resource": "*",
    "Condition": {
        "NotIpAddress": {
            "aws:SourceIp": [
                "192.0.2.0/24",
                "2001:db8::/32"
            ]
        }
    }
}
```

AWS Private Certificate Authority CP/CPS Framework für Kunden

AWS Private Certificate Authority bietet Infrastrukturdienste, die es Ihnen ermöglichen, Zertifizierungsstellenhierarchien (CA), einschließlich Stamm- und untergeordneter Zertifizierungsstellen, zu erstellen CAs, ohne die Investitions- und Wartungskosten für den Betrieb einer lokalen Zertifizierungsstelle anfallen zu müssen. Bei der Erstellung AWS Private CA Ihrer CA-Hierarchien tragen Sie und eine gemeinsame Verantwortung. AWS Private CA Das Modell der geteilten Verantwortung kann Ihnen helfen, Ihre betriebliche Belastung zu verringern, da AWS die physische Sicherheit der Einrichtungen, in denen der Service betrieben wird, verwaltet und kontrolliert wird. Sie übernehmen die Verantwortung und Verwaltung der Zertifizierungsstelle (einschließlich Erstellung und Löschung von CA-Ressourcen, Verteilung von Vertrauensankern, Erstellung von PKI-Hierarchien, Zertifizierungsrichtlinien und -praktiken, Konfiguration für das Zulassen oder Verweigern der gemeinsamen Nutzung von CA AWS-Konten, Richtlinien für die Verwendung von Vorlagen, Auditing, Zugriffskontrollen, einschließlich Aufgabentrennung, und andere CA-Konfigurationen und Richtlinien). Sie sollten die Dienste, die Sie wählen, sorgfältig abwägen, da Ihre Verantwortlichkeiten je nach den verwendeten Diensten, der Integration dieser Dienste in Ihre IT-Umgebung und den geltenden Gesetzen und Vorschriften variieren. Weitere Informationen finden Sie im Modell der gemeinsamen Verantwortung für AWS Cloud Sicherheit.

Die Erstellung einer Zertifizierungsrichtlinie (CP) oder eines Certification Practice Statements (CPS) für Ihre private Zertifizierungsstelle ist ein wichtiger Bestandteil der Verwaltung Ihrer Public-Key-Infrastruktur (PKI). Ein CP definiert alle requirements/rules für Ihre PKI und das CPS erklärt, wie Sie die CP-Anforderungen erfüllen. Sie sind dafür verantwortlich, einen CP und ein CPS als Zertifizierungsstelle für Ihre PKI einzurichten. AWS Private CA stellt Ihnen AWS Kontroll- und Compliance-Unterlagen zur Verfügung, z. B. den AWS System and Organization Controls (SOC) 2-Bericht, den Sie bei der Erstellung Ihres CP und CPS sowie bei der Durchführung

Ihrer Kontrollbewertungs- und Überprüfungsverfahren nach Bedarf verwenden können. AWS SOC-Berichte sind unabhängige Prüfungsberichte von Drittanbietern, die belegen, wie wichtige Compliance-Kontrollen und -Ziele AWS erreicht werden. Der Zweck der Berichte besteht darin, Ihnen und Ihren Prüfern zu helfen, die zur Unterstützung der Betriebsabläufe und der Einhaltung von Vorschriften eingerichteten AWS Kontrollen zu verstehen.

Dieses Dokument stellt ein Framework vor, das sich an RFC 3647 orientiert, um Sie bei der Erstellung Ihres CP und CPS zu unterstützen und die gemeinsame Verantwortung zwischen Ihnen und AWS Private CA Abschnitte der CP/CPS Anforderungen, für die die Einhaltung der Vorschriften zuständig ist, sind mit „gemeinsam genutzt“ oder „AWS Private Certificate Authority" gekennzeichnet. Die entsprechenden „ergänzenden Informationen“ sollen Ihnen helfen zu verstehen, wie AWS Private CA die damit verbundene Anforderung erfüllt wird. AWS Private CA CP/CPS Bei Anforderung 5 (4.5.1) handelt es sich beispielsweise um eine AWS Private CA Verantwortung, und die entsprechende Kontrollsprache finden Sie in Abschnitt D.6 des AWS SOC 2-Berichts, um Ihr CP/CPS auszufüllen. Weitere Informationen zu AWS SOC-Berichten und dazu, wie Sie Zugriff auf SOC-Berichte beantragen können, finden Sie auf unserer SOC-Seite. FAQs

CP/CPS-Anforderungen und Zuständigkeiten

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
1. Einführung (Alle)	Sie	Sie sind dafür verantwortlich, den Überblick, den Namen und die Identifikation des Dokuments, die PKI-Teilnehmer, die Verwendung von Zertifikaten, die Richtlinienverwaltung sowie Definitionen und Akronyme im Zusammenhang mit Ihrer PKI zu dokumentieren.
2. Aufgaben im Zusammenhang mit der Veröffentlichung und dem Repositorium (Alle)	Sie	Sie sind dafür verantwortlich, die Definitionen zu Ihrer PKI zu dokumentieren.
3. Identifizierung und Authentifizierung (Alle)	Sie	Sie sind dafür verantwortlich, die Verfahren zur Authentif

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
		izierung der Identität und and/ or anderer Attribute eines Endbenutzer-Zertifikatsantr agstellers bei einer CA oder Registration Authority (RA) vor der Zertifikatsausstellung zu dokumentieren.
4. Betriebliche Anforderungen für den gesamten Lebenszyk lus des Zertifikats (4.4.1 — 4.4.6, 4.4.9 — 4.4.11)	Freigegeben	Sie sind dafür verantwortlich, die Anforderungen festzuleg en, die an die ausstelle nde Zertifizierungsstelle, den Antragsteller CAs RAs, Abonnenten oder andere Teilnehmer in Bezug auf den Lebenszyklus eines Zertifikats gestellt werden. AWS Private CA bietet Ihnen zwei vollständig verwaltete Mechanismen zur Unterstüt zung der Überprüfung des Sperrstatus: das Online Certificate Status Protocol (OCSP) und die Zertifika tssperrlisten (CRLs), mit denen Sie 4.4.9 und 4.4.10 erfüllen können.
4. Betriebliche Anforderungen für den gesamten Lebenszyk lus des Zertifikats (4.4.7, 4.4.8, 4.4.12)	–	AWS Private CA unterstützt weder Certificate Re-Key noch Certificate Modification noch Key Escrow and Recovery.

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
5. Einrichtung, Verwaltung und Betriebskontrollen (4.5.1)	AWS Private CA	Sie übernehmen Zugriffsk ontrollen, die Ihnen helfen, die Anforderungen in diesem Abschnitt zu erfüllen, die im AWS Private CA SOC 2 Type 2-Bericht enthalten sind (siehe Abschnitt D.6 Physische Sicherheit und Umweltschutz). **Note** Sie sind verantwor tlich für die physische Sicherheit und Datenklassifizierung von CA-Daten, die aus der AWS Umgebung exportiert oder übertragen werden, aber nicht für die physische Sicherheit der CA-Daten, die dort gespeichert sind. AWS

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
5. Einrichtung, Verwaltung und Betriebskontrollen (4.5.2)	Freigegeben	Sie sind dafür verantwortlich, die Anforderungen in diesem Abschnitt zu erfüllen, die sich speziell auf die Definition vertrauenswürdiger Rollen für den Betrieb Ihrer PKI-Umgebung beziehen. AWS Private CA verwaltet vertrauenswürdige Rollen, die für den physischen Zugriff auf kryptografische Module spezifisch sind.

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
5. Einrichtung, Verwaltung und Betriebskontrollen (4.5.3)	Freigegeben	Sie sind dafür verantwortlich, die Anforderungen in diesem Abschnitt zu erfüllen, die sich auf die Verfahren zur Zuverlässigkeitsüberprüfung, Schulung und Disziplinarmaßnahmen für Ihre Vertrauenspersonen beziehen. Sie übernehmen die Kontrollen in Bezug auf Zuverlässigkeitsüberprüfungen, Schulungen und Disziplinarmaßnahmen für AWS Mitarbeiter, die in den Anwendungsbereich des AWS Private CA SOC 2 Type 2-Berichts fallen (siehe Abschnitt A. Richtlinien, A.1 Kontrollumgebung, B. Kommunikation und D.1 Sicherheitsorganisation und D.2 Benutzerzugriff für Mitarbeiter).

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
5. Einrichtung, Verwaltung und Betriebskontrollen (4.5.4)	Freigegeben	Sie sind verantwortlich für die Aktivierung und Konfiguration der Aufbewahrung sowie für den Schutz CloudTrail und die Prüfung von Berichtsprotokollen und CloudWatch Warnmeldungen. Darüber hinaus sind Sie dafür verantwortlich, Verfahren zur Protokollverarbeitung zu erstellen und Sicherheitslücken bei Ihrer Nutzung des AWS Private CA Dienstes zu bewerten, die den Anforderungen in diesem Abschnitt entsprechen. Sie übernehmen Kontrollen in Bezug auf die Verfügbarkeit Ihrer Protokolle, die physische access/site Sicherheit, das CA/RA Konfigurationsmanagement, die Sicherheit der AWS Infrastrukturprotokolle und die Schwachstellenbeurteilung der AWS Infrastruktur, die in den Anwendungsbereich des AWS Private CA SOC 2 Type 2-Berichts fallen (siehe Abschnitt A.1 Kontrollumgebung, Abschnitt C.1 Service Commitments, D.2 Employee User Access, D.3 Logical Security, D.6 Physical Security and Environmental Protection, D.7

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
		Change Management, D.8 Datenintegrität, Verfügbarkeit und Redundanz sowie E.1 Überwachungsaktivitäten).
5. Einrichtung, Management und Betriebskontrollen (4.5.5)	Freigegeben	Sie sind dafür verantwortlich, Sicherungs- und Aufbewahrungszeiträume so zu konfigurieren, dass sie den Anforderungen in diesem Abschnitt entsprechen. Sie übernehmen die Kontrollen in Bezug auf die Verfügbarkeit Ihrer Protokolle (bei der Konfiguration), die im Rahmen des AWS Private CA SOC 2 Type 2-Berichts fallen (siehe D.8 Datenintegrität, Verfügbarkeit und Redundanz).
5. Einrichtung, Verwaltung und Betriebskontrollen (4.5.6)	–	AWS Private CA unterstützt Key Changeover nicht.

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
5. Einrichtung, Verwaltung und Betriebskontrollen (4.5.7)	Freigegeben	Sie sind für die Implement ierung von Verfahren zur Behandlung von Zwischenf ällen und Kompromissen verantwortlich, AWS Private CA die speziell auf Sie zugeschnitten sind und die Anforderungen in diesem Abschnitt erfüllen. Sie übernehmen Verfahren zur Behandlung von Zwischenf ällen, zur Gefahrenabwehr, zur Geschäftskontinuität und zur Notfallwiederherstellung, die speziell für den Betrieb physischer Standorte und für den Infrastrukturbetrieb vorgesehen sind und Ihnen dabei helfen, die Anforderu ngen in diesem Abschnitt zu erfüllen, die im AWS Private CA SOC 2 Type 2-Datensc hutzbericht enthalten sind (siehe D.8 Datenintegrität, Verfügbarkeit und Redundanz und Abschnitt D.10 Datenschu tz).

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
5. Einrichtung, Verwaltung und Betriebskontrollen (4.5.8)	Sie	Sie müssen die Anforderungen im Zusammenhang mit den Verfahren zur Kündigung und Kündigung einer CA oder RA dokumentieren, einschließlich der Identität des Verwahrers der Archivunterlagen von CA und RA.
6. Technische Kontrollen (4.6.1)	Freigegeben	Sie sind dafür verantwor tlich, die Anforderungen an die Schlüsselgenerierung und Installation Ihrer PKI zu dokumentieren. AWS Private CA stellt Ihnen kryptografische Module zur Verfügung, die nach FIPS 140-3 Level 3 für die CA-Schlüsselgenerierung zertifizi ert sind.

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
6. Technische Kontrollen (4.6.2)	Freigegeben	Sie sind verantwortlich für die Dokumentation des Schutzes privater Schlüssel und der Kontrollen bei der Entwicklung kryptografischer Module, wie z. B. kryptografische Standardanforderungen und Kontrollen für mehrere Personen. AWS Private CA stellt Ihnen kryptografische Module zur Verfügung, die nach FIPS 140-3 Level 3 für die Generierung von CA-Schlüsseln und physische Zugriffskontrollen durch zwei Parteien zertifiziert sind. HSMs
6. Technische Kontrollen (4.6.3)	Sie	Sie sind dafür verantwortlich, andere Aspekte der Schlüsselpaarverwaltung zu dokumentieren, z. B. die Archivierung Ihres öffentlichen Schlüssels und die Betriebsdauer von Zertifikaten.

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
6. Technische Kontrollen (4.6.4)	–	AWS AWS Private CA HSMs sind immer online und haben keine Ahnung von „Aktivierungsdaten“. **Note** Sie sind dafür verantwortlich, Benutzerzugriffskontrollen für Ihre private Zertifizierungsstelle zu implementieren, um die Möglichkeit, Zertifizierungsstellen zu erstellen und Zertifikate auszustellen, angemessen einzuschränken.

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
6. Technische Kontrollen (4.6.5)	Freigegeben	Sie sind dafür verantwortlich, die Computersicherheitskontrollen für Ihre Nutzung Ihrer privaten CA zu dokumentieren. Sie übernehmen die Kontrollen in Bezug auf den logischen Zugriff von AWS Mitarbeitern, die Netzwerk- und Computersicherheitskontrollen der AWS Infrastruktur und die Steuerung der Kennwortparameter von AWS Mitarbeiterkonten, die in den Anwendungsbereich des AWS Private CA SOC 2 Type 2-Berichts fallen (siehe Abschnitt D.2 Mitarbeiterzugriff, D.3 Logische Sicherheit und D.6 Physische Sicherheit und Umweltschutz).

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
6. Technische Kontrollen (4.6.6)	Freigegeben	Sie sind dafür verantwortlich, die Sicherheitsmanagem entkontrollen im Zusammenh ang mit Ihrer Nutzung Ihrer privaten CA zu dokumentieren. Sie übernehmen die Kontrolle n im Zusammenhang mit den Kontrollen zur Systement wicklung des AWS Private CA Services, die im Rahmen des AWS Private CA SOC 2 Type 2-Berichts behandelt werden (siehe Abschnitt D.7 Change Management).
6. Technische Kontrollen (4.6.7)	Freigegeben	Sie sind dafür verantwor tlich, die Netzwerksicherheit skontrollen für Ihre Nutzung von Private CA zu dokumenti eren, sofern dies für Ihre PKI-Umgebung zutrifft. Sie übernehmen die Kontrolle n im Zusammenhang mit den Netzwerksicherheitskontroll en der AWS Infrastruktur, die im Rahmen des AWS Private CA SOC 2 Type 2-Berichts behandelt werden (siehe Abschnitt C.1 Service Commitments, D.3 Logical Security und E.1 Monitoring Activities).

CP/CPS-Anforderung	Verantwortung	Zusätzliche Informationen
6. Technische Kontrollen (4.6.8)	AWS Private CA	AWS Private CA verwendet vertrauenswürdige Zeitquellen, um CA-Daten mit einem Zeitstempel zu versehen.
7. Zertifikat-, CRL- und OCSP-Profile (Alle)	Freigegeben	Sie sind dafür verantwortlich, die Profilanforderungen und die Eingabe von Zertifikaten zu dokumentieren, die den Anforderungen Ihrer PKI-Umgebung entsprechen. AWS Private CA stellt Ihnen Profilvorlagen zur Verfügung, mit denen Sie Ihre Profilanforderungen erfüllen können.
8. Compliance-Audits und andere Bewertungen (Alle)	Freigegeben	Sie sind für die Dokumentation von Compliance-Audits und anderen Bewertungen verantwortlich. AWS Private CA stellt Ihnen einen SOC 2-Bericht zur Verfügung, der Ihnen und Ihren Prüfern hilft, die zur Unterstützung des Betriebs und der AWS Einhaltung von Vorschriften eingerichteten Kontrollen zu verstehen.
9. Andere geschäftliche und rechtliche Fragen	Sie	Sie sind dafür verantwortlich, allgemeine geschäftliche und rechtliche Angelegenheiten zu dokumentieren, die Ihre private CA betreffen.

AWS Private CA Ressourcen überwachen

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer AWS Private CA anderen AWS Lösungen. AWS bietet die folgenden Überwachungstools, mit denen Sie beobachten AWS Private CA, melden können, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen ergreifen können:

- Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können Kennzahlen erfassen und verfolgen, benutzerdefinierte Dashboards erstellen und Alarme festlegen, die Sie benachrichtigen oder Maßnahmen ergreifen, wenn eine bestimmte Metrik einen von Ihnen festgelegten Schwellenwert erreicht. Sie können beispielsweise die CPU-Auslastung oder andere Kennzahlen Ihrer EC2 Amazon-Instances CloudWatch verfolgen und bei Bedarf automatisch neue Instances starten. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.
- Mit Amazon CloudWatch Logs können Sie Ihre Protokolldateien von EC2 Amazon-Instances und anderen Quellen überwachen CloudTrail, speichern und darauf zugreifen. CloudWatch Logs kann Informationen in den Protokolldateien überwachen und Sie benachrichtigen, wenn bestimmte Schwellenwerte erreicht werden. Sie können Ihre Protokolldaten auch in einem sehr robusten Speicher archivieren. Weitere Informationen finden Sie im Amazon CloudWatch Logs-Benutzerhandbuch.
- AWS CloudTrail erfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS-Konto -Kontos erfolgten, und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon-S3-Bucket. Sie können die Benutzer und Konten, die AWS aufgerufen haben, identifizieren, sowie die Quell-IP-Adresse, von der diese Aufrufe stammen, und den Zeitpunkt der Aufrufe ermitteln. Weitere Informationen finden Sie im AWS CloudTrail - Benutzerhandbuch.
- Amazon EventBridge ist ein serverloser Event-Bus-Service, der es einfach macht, Ihre Anwendungen mit Daten aus einer Vielzahl von Quellen zu verbinden. EventBridge liefert einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen, Software-as-a-Service (SaaS-) Anwendungen und AWS Diensten und leitet diese Daten an Ziele wie Lambda weiter. Auf diese Weise können Sie Ereignisse überwachen, die in Services auftreten, und ereignisgesteuerte Architekturen erstellen. Weitere Informationen finden Sie im EventBridge Amazon-Benutzerhandbuch.

In den folgenden Themen werden Tools zur AWS Cloud-Überwachung beschrieben, die zur Verwendung mit AWS Private CA verfügbar sind.

AWS Private CA CloudWatch Metriken

Amazon CloudWatch ist ein Monitoring-Service für AWS Ressourcen. Sie können CloudWatch damit Kennzahlen sammeln und verfolgen, Alarme einrichten und automatisch auf Änderungen Ihrer AWS Ressourcen reagieren. CloudWatch Metriken werden mindestens einmal veröffentlicht.

AWS Private CA unterstützt die folgenden CloudWatch Metriken.

Metrik	Description
`CRLGenerated`	Eine Zertifikatsperrliste (CRL) wurde generiert. Die Metrik gilt nur für eine private CA.
`MisconfiguredCRLBucket`	Der S3-Bucket, der für die Zertifikatsperrliste angegeben wurde, ist nicht korrekt konfiguriert. Überprüfen Sie die Bucket-Richtlinie. Die Metrik gilt nur für eine private CA.
`Time`	Die Zeit in Millisekunden zwischen einer Ausstellungsanfrage und dem Abschluss (oder Misserfolg) der Ausgabe. Diese Metrik gilt nur für den Vorgang. IssueCertificate
`Success`	Ein Zertifikat wurde erfolgreich ausgestel lt. Diese Metrik gilt nur für den IssueCert ificateVorgang.
`Failure`	Eine Operation ist fehlgeschlagen. Diese Metrik gilt nur für den IssueCertificateVorgang.

Weitere Informationen zu CloudWatch Metriken finden Sie in den folgenden Themen:

- Amazon CloudWatch Metrics verwenden
- CloudWatchAmazon-Alarme erstellen

Überwachung AWS Private CA mit CloudWatch Ereignissen

Sie können Amazon CloudWatch Events verwenden, um Ihre AWS Services zu automatisieren und automatisch auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen zu reagieren. Ereignisse aus AWS Services werden nahezu in Echtzeit an CloudWatch Events übermittelt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie von Interesse sind und welche automatisierten Aktionen ergriffen werden sollen, wenn ein Ereignis einer Regel entspricht. CloudWatch Ereignisse werden mindestens einmal veröffentlicht. Weitere Informationen finden Sie unter Eine CloudWatch Ereignisregel erstellen, die bei einem Ereignis ausgelöst wird.

CloudWatch Ereignisse werden mithilfe von Amazon in Aktionen umgewandelt EventBridge. Mit können Sie Ereignisse verwenden EventBridge, um Ziele wie AWS Lambda Funktionen, AWS Batch Jobs, Amazon SNS SNS-Themen und viele andere auszulösen. Weitere Informationen finden Sie unter Was ist Amazon EventBridge?

Erfolg oder Misserfolg beim Erstellen einer privaten CA

Diese Ereignisse werden durch den CreateCertificateAuthorityVorgang ausgelöst.

Herzlichen Glückwunsch

Bei Erfolg gibt die Operation den ARN der neuen Zertifizierungsstelle zurück.

```
{
   "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"
   }
}
```

Fehler

Bei einem Fehler gibt die Operation einen ARN für die Zertifizierungsstelle zurück. Mithilfe des ARN können Sie anrufen, DescribeCertificateAuthorityum den Status der CA zu ermitteln.

```
{
   "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"
   }
}
```

Erfolg oder Misserfolg bei der Ausstellung eines Zertifikats

Diese Ereignisse werden durch den IssueCertificateVorgang ausgelöst.

Herzlichen Glückwunsch

Bei Erfolg gibt der Vorgang die Werte ARNs der Zertifizierungsstelle und des neuen Zertifikats zurück.

```
{
   "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"
   }
}
```

Fehler

Bei einem Fehler gibt die Operation einen Zertifikats-ARN und den ARN der Zertifizierungsstelle zurück. Mit dem Zertifikat ARN können Sie anrufen, GetCertificateum den Grund für den Fehler zu erfahren.

```
{
   "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"
   }
}
```

Erfolg beim Widerrufen eines Zertifikats

Dieses Ereignis wird durch den RevokeCertificateVorgang ausgelöst.

Wenn das Widerrufen fehlschlägt oder wenn das Zertifikat bereits widerrufen wurde, wird kein Ereignis gesendet.

Herzlichen Glückwunsch

Bei Erfolg gibt der Vorgang den Wert der ARNs Zertifizierungsstelle und des gesperrten Zertifikats zurück.

```
{
   "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"
   }
}
```

Erfolg oder Misserfolg beim Generieren einer CRL

Diese Ereignisse werden durch den RevokeCertificateVorgang ausgelöst, der zur Erstellung einer Zertifikatssperrliste (CRL) führen sollte.

Herzlichen Glückwunsch

Bei Erfolg gibt die Operation den ARN der Zertifizierungsstelle zurück, die der CRL zugeordnet ist.

```
{
   "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"
   }
}
```

Fehler 1 — CRL konnte aufgrund eines Berechtigungsfehlers nicht in Amazon S3 gespeichert werden

Überprüfen Sie Ihre Amazon S3 S3-Bucket-Berechtigungen, falls dieser Fehler auftritt.

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

Fehler 2 — CRL konnte aufgrund eines internen Fehlers nicht in Amazon S3 gespeichert werden

Wenn dieser Fehler auftritt, versuchen Sie, die Operation erneut durchzuführen.

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

Fehler 3 — AWS Private CA Es konnte keine CRL erstellt werden

Überprüfen Sie Ihre CloudWatch -Metriken, um diesen Fehler zu beheben.

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

Erfolg oder Misserfolg bei der Erstellung eines CA-Auditberichts

Diese Ereignisse werden durch den CreateCertificateAuthorityAuditReportVorgang ausgelöst.

Herzlichen Glückwunsch

Bei Erfolg gibt die Operation den ARN der Zertifizierungsstelle und die ID des Audit-Berichts zurück.

```
{
   "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"
   }
}
```

Fehler

Ein Prüfbericht kann fehlschlagen, wenn die PUT Berechtigungen für Ihren Amazon S3 S3-Bucket AWS Private CA fehlen, wenn die Verschlüsselung für den Bucket aktiviert ist oder aus anderen Gründen.

```
{
   "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"
   }
}
```

Protokollieren von AWS Private Certificate Authority API-Aufrufen mit AWS CloudTrail

AWS Private Certificate Authority ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in ausgeführt

wurden AWS Private CA. CloudTrail erfasst API-Aufrufe und Signiervorgänge AWS Private CA als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der AWS Private CA Konsole und Codeaufrufen für die AWS Private CA API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für AWS Private CA. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage ermitteln CloudTrail, an die die Anfrage gestellt wurde AWS Private CA, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Informationen.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

AWS Private CA Informationen in CloudTrail

CloudTrail ist auf Ihrem aktiviert AWS-Konto , wenn Sie das Konto erstellen. Wenn eine Aktivität in stattfindet AWS Private CA, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem anzeigen, suchen und herunterladen AWS-Konto. Weitere Informationen finden Sie unter Ereignisse mit dem CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, einschließlich der Ereignisse für AWS Private CA, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle AWS Private CA Aktionen werden von der AWS Private CA API-Referenz protokolliert CloudTrail und sind in dieser dokumentiert. Beispielsweise generieren Aufrufe von

`IssueCertificate` und `CreateAuditReport` Aktionen Einträge in den CloudTrail Protokolldateien. `ImportCACertificate`

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM-) Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter CloudTrail -Element userIdentity.

AWS Private CA Management-Ereignisse

AWS Private CA integriert sich in CloudTrail , um API-Aktionen aufzuzeichnen, die von einem Benutzer, einer Rolle oder einem AWS Dienst ausgeführt wurden AWS Private CA. Sie können CloudTrail AWS Private CA API-Anfragen in Echtzeit überwachen und Protokolle in Amazon Simple Storage Service, Amazon CloudWatch Logs und Amazon CloudWatch Events speichern. AWS Private CA unterstützt die Protokollierung der folgenden Aktionen und Operationen als Ereignisse in CloudTrail Protokolldateien:

- CreateCertificateAuthority
- CreateCertificateAuthorityAuditReport
- CreatePermission
- DeleteCertificateAuthority
- DeletePermission
- DeletePolicy
- DescribeCertificateAuthority
- DescribeCertificateAuthorityReport
- GetCertificate
- GetCertificateAuthorityCertificate
- GetCertificateAuthorityCsr
- GetPolicy

- ImportCertificateAuthorityCertificate
- IssueCertificate
- ListCertificateAuthorities
- ListPermissions
- ListTags
- PutPolicy
- RestoreCertificateAuthority
- RevokeCertificate
- TagCertificateAuthority
- UntagCertificateAuthority
- UpdateCertificateAuthority
- `GenerateOCSPResponse`- Wird ausgelöst, wenn eine OCSP-Antwort AWS Private CA generiert wird.
- `SignCertificate`- Wird generiert, wenn Ihr Kunde anruft IssueCertificate.
- `SignOCSPResponse`- Wird generiert, wenn eine OCSP-Antwort AWS Private CA signiert wird.
- `GenerateCRL`- Wird AWS Private CA generiert, wenn eine Zertifikatssperrliste (CRL) generiert wird.
- `SignCACSR`- Wird generiert, wenn AWS Private CA eine Certificate Authority (CA) - Zertifikatsignieranforderung (Certificate Authority, CA) signiert wird.
- `SignCRL`- Wird generiert, wenn eine AWS Private CA CRL signiert wird.

Beispiele für Ereignisse AWS Private CA

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Im Folgenden finden Sie Beispiele für AWS Private CA CloudTrail Ereignisse.

Beispiel 1: Management-Ereignis, **`IssueCertificate`**

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `IssueCertificate` Aktion demonstriert.

```
{
   "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"
   }
}
```

Beispiel 2: Management-Ereignis, **`ImportCertificateAuthorityCertificate`**

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `ImportCertificateAuthorityCertificate` Aktion demonstriert.

```
{
   "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"
}
```

Probleme beheben mit AWS Private Certificate Authority

Wenn Sie Probleme mit der Verwendung von AWS Private Certificate Authority haben, lesen Sie in den folgenden Themen nach.

Themen

Probleme mit dem Widerruf von AWS Private CA Zertifikaten beheben

Latenz bei der OCSP-Antwort

Die OCSP-Reaktionszeit kann langsamer sein, wenn der Anrufer geografisch weit von einem regionalen Edge-Cache oder von der Region der ausstellenden Zertifizierungsstelle entfernt ist. Weitere Informationen zur regionalen Edge-Cache-Verfügbarkeit finden Sie unter Globales Edge-Netzwerk. Wir empfehlen, Zertifikate in einer Region auszustellen, in der sie verwendet werden.

Widerruf von selbstsignierten Zertifikaten

Sie können ein selbstsigniertes CA-Zertifikat nicht widerrufen. Um das Zertifikat funktionell zu widerrufen, löschen Sie die CA.

Beheben Sie AWS Private Certificate Authority Ausnahmemeldungen

Ein AWS Private CA Befehl kann aus verschiedenen Gründen fehlschlagen. Informationen zu den einzelnen Ausnahmen und Lösungsempfehlungen finden Sie in der folgenden Tabelle.

AWS Private CA Ausnahmen

Ausnahme Zurückgegeben von AWS Private CA	Description	Abhilfe
`AccessDeniedException`	Die für die Verwendung des angegebenen Befehls erforderlichen Berechtigungen wurden nicht von einer privaten Zertifizierungsstelle an das den Aufruf durchführende Konto delegiert.	Informationen zum Delegieren von Berechtigungen in finden Sie AWS Private CA unterWeisen Sie ACM Berechtigungen zur Zertifikatsverlängerung zu.
`InvalidArgsException`	Eine Zertifikaterstellungs- oder -erneuerungsanforderung mit ungültigen Parametern wurde gestellt.	Überprüfen Sie in der jeweiligen Dokumentation des Befehls, ob Ihre Eingabeparameter gültig sind. Wenn Sie ein neues Zertifikat erstellen, stellen Sie sicher, dass der angeforderte Signaturalgorithmus mit dem Schlüsseltyp der CA verwendet werden kann.
`InvalidStateException`	Die zugeordnete private Zertifizierungsstelle kann das Zertifikat nicht erneuern, da es sich nicht im Zustand `ACTIVE` befindet.	Versuchen Sie, Ihre private Zertifizierungsstelle wiederherzustellen. Wenn sich die private Zertifizierungsstelle außerhalb des Wiederherstellungszeitraums befindet, kann die Zertifizierungsstelle nicht wiederhergestellt und das Zertifikat kann nicht erneuert werden.
`LimitExceededException`	Jede Zertifizierungsstelle (CA) verfügt über eine bestimmte Anzahl von Zertifikaten, die	Wenden Sie sich an das AWS Support Center, um eine

Ausnahme Zurückgegeben von AWS Private CA	Description	Abhilfe
	sie ausstellen kann. Die private Zertifizierungsstelle, die dem angegebenen Zertifikat zugeordnet ist, hat ihr Kontingent erreicht. Weitere Informationen finden Sie im Allgemeine AWS-Referenz Handbuch unter Service Quotas.	Erhöhung des Kontingents zu beantragen.
`MalformedCSRException`	Die Zertifikatsignieranforderung (Certificate Signing Request, CSR), die an AWS Private CA gesendet wurde, kann nicht überprüft oder validiert werden.	Vergewissern Sie sich, dass Ihre CSR ordnungsgemäß generiert und konfiguriert wurde.
`OtherException`	Die Anforderung ist aufgrund eines internen Fehlers fehlgeschlagen.	Versuchen Sie erneut, den Befehl auszuführen. Wenn das Problem weiterhin besteht, wenden Sie sich an das AWS Support Center.
`RequestFailedException`	Ein Netzwerkproblem in Ihrer AWS Umgebung hat dazu geführt, dass die Anfrage fehlschlug.	Wiederholen Sie die Anforderung. Wenn der Fehler weiterhin besteht, überprüfen Sie Ihre Amazon VPC (VPC) -Konfiguration.
`ResourceNotFoundException`	Die private Zertifizierungsstelle, die das Zertifikat ausgestellt hat, wurde gelöscht und ist nicht mehr vorhanden.	Fordern Sie ein neues Zertifikat von einer anderen aktiven Zertifizierungsstelle an.

Ausnahme Zurückgegeben von AWS Private CA	Description	Abhilfe
`ThrottlingException`	Eine angeforderte API-Aktion ist fehlgeschlagen, da sie ein Kontingent überschritten hat.	Vergewissern Sie sich, dass Sie nicht mehr Anrufe tätigen als zulässig. AWS Private CA Ein `ThrottlingException` Fehler kann auch auftreten, weil Sie auf einen vorübergehenden Zustand gestoßen sind und nicht auf eine Überschreitung des Kontingents zurückzuführen sind. Wenn der Fehler auftritt und Sie keine Anrufe getätigt haben, die das Kontingent überschritten haben, versuchen Sie es erneut mit Ihrer Anfrage. Wenn Ihr Kontingent ausgeschöpft ist, können Sie möglicherweise eine Erhöhung beantragen. Weitere Informationen finden Sie im Allgemeine AWS-Referenz Handbuch unter Service Quotas.

Ausnahme Zurückgegeben von AWS Private CA	Description	Abhilfe
`ValidationException`	Die Eingabeparameter der Anforderung waren falsch formatiert, oder die Gültigkeitsdauer des Stammzertifikats endet vor der Gültigkeitsdauer des angeforderten Zertifikats.	Überprüfen Sie die Syntaxanforderungen der Eingabeparameter des Befehls sowie den Gültigkeitszeitraum des Stammzertifikats Ihrer Zertifizierungsstelle. Weitere Informationen zum Ändern des Gültigkeitszeitraums finden Sie unter Aktualisieren Sie eine private Zertifizierungsstelle in AWS Private Certificate Authority.

Beheben Sie AWS Private CA Matter-konforme Zertifikatsfehler

Der Matter-Konnektivitätsstandard spezifiziert Zertifikatskonfigurationen, die die Sicherheit und Konsistenz von IoT-Geräten (Internet of Things) verbessern. Java-Beispiele für die Erstellung von Matter-konformen Root-CA-, Zwischen-CA- und Entity-Zertifikaten finden Sie unter. Wird AWS Private CA zur Implementierung von Matter-Zertifikaten verwendet

Zur Unterstützung bei der Fehlerbehebung stellen die Matter-Entwickler ein Tool zur Überprüfung von Zertifikaten namens chip-cert zur Verfügung. Fehler, die das Tool meldet, sind in der folgenden Tabelle mit Abhilfemaßnahmen aufgeführt.

Fehlercode	Bedeutung	Abhilfe
0x0000035	`BasicConstraints` ,`KeyUsage`, und `ExtensionKeyUsage` Erweiterungen müssen als kritisch markiert werden.	Stellen Sie sicher, dass Sie die richtige Vorlage für Ihren Anwer ausgewählt haben.

Fehlercode	Bedeutung	Abhilfe
0x000000 0	Die Erweiterung zur Identifizierung des Autorität sschlüssels muss vorhanden sein.	AWS Private CA legt die Erweiterung zur Identifizierung des Au sschlüssels für Stammzertifikate nicht fest. Sie müssen mithilfe einen Base64-codierten AuthorityKeyIdentifier Wert generieren dann durch eine übergeben. CustomExtension Weitere Informa erhalten Sie unter Aktivieren Sie eine Root-CA für Node Operat Certificates (NOC). und Aktivieren Sie eine Product Attestation (PAA).
0x000000 E	Das Zertifikat ist abgelaufen.	Stellen Sie sicher, dass das von Ihnen verwendete Zertifikat noc abgelaufen ist.

Fehlercode	Bedeutung	Abhilfe
0x00000004	Fehler bei der Validierung der Zertifikatskette.	Dieser Fehler kann auftreten, wenn Sie versuchen, ein Matter-k Endentitätszertifikat zu erstellen, ohne die bereitgestellten Java- piele zu verwenden, die die AWS Private CA API verwenden, u ordnungsgemäß konfiguriertes Zertifikat zu übergeben. KeyUsa AWS Private CA Generiert standardmäßig KeyUsage Neun-Bit- Erweiterungswerte, wobei das neunte Bit zu einem zusätzliche führt. Matter ignoriert das zusätzliche Byte bei Formatkonvertie was zu Fehlern bei der Kettenvalidierung führt. Ein Wert Custo ensionin der `APIPassthrough` Vorlage kann jedoch verwend werden, um die genaue Anzahl der Byte im `KeyUsage` Wert fes Ein Beispiel finden Sie unter Erstellen Sie ein Node Operational te (NOC). Wenn Sie den Beispielcode ändern oder ein alternatives X.509- fsprogramm wie OpenSSL verwenden, müssen Sie eine manue Überprüfung durchführen, um Fehler bei der Kettenvalidierung z vermeiden. Um zu überprüfen, ob Konvertierungen verlustfrei sind 1. Verwenden Sie openssl, um zu überprüfen, ob ein Zertifika Knotenzertifikat (Endentitätszertifikat), eine gültige Kette en diesem Beispiel `rcac.pem` ist es das Stammzertifizierungs zertifikat, `icac.pem` das Zwischenzertifikat der Zertifizieru und `noc.pem` das Knotenzertifikat. ``` openssl verify -verbose -CAfile <(cat rcac.pem icac noc.pem ``` 2. Verwenden Sie chip-cert, um das Knotenzertifikat im PEM- das TLV-Format (Tag, Länge, Wert) und wieder zurück zu l ren.

Fehlercode	Bedeutung	Abhilfe
		`` ./chip-cert convert-cert noc.pem noc.chip -c ./chip-cert convert-cert noc.chip noc_converted.pem ``` Die Dateien `noc.pem` und `noc_converted.pem` sollten die gleichen sein, die durch ein Tool zum Vergleich von Zei tten bestätigt wurden.

Kubernetes sichern mit AWS Private Certificate Authority

Sie können AWS Private Certificate Authority damit Zertifikate für die sichere Authentifizierung und Verschlüsselung über TLS und mTLS bereitstellen. AWS Private CA stellt ein Open-Source-Plugin, AWS Private CA Connector for Kubernetes, (`aws-privateca-issuer`) für das weit verbreitete Cert-Manager-Add-on für Kubernetes bereit, das Zertifikate anfordert, sie an Kubernetes-Secrets verteilt und die Zertifikatserneuerung automatisiert.

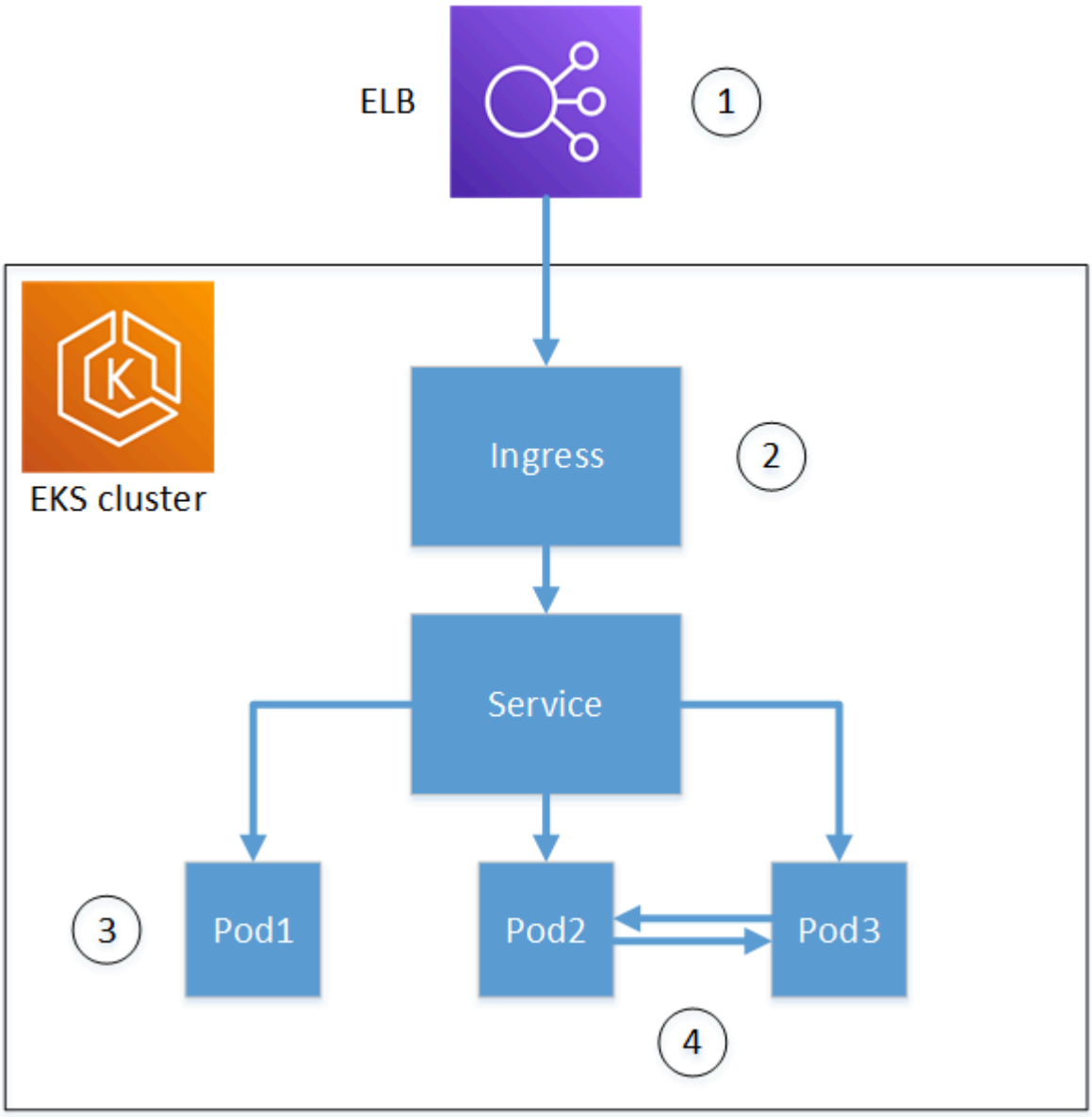
Das Plugin ermöglicht `aws-privateca-issuer` es Ihnen, Zertifikate auszustellen über. AWS Private CA `cert-manager` Sie können das Plugin mit Amazon Elastic Kubernetes Service (Amazon EKS), einem selbstverwalteten Kubernetes-Cluster auf AWS oder in einem lokalen Kubernetes-Cluster verwenden. Das Plugin funktioniert sowohl auf x86- als auch auf ARM-Architekturen.

AWS Private CA hat HSM-gestützte Schlüssel, die nicht exportiert werden können. Wenn Sie gesetzliche Anforderungen für die Zugriffskontrolle und die Prüfung Ihrer CA-Operationen haben, können Sie diese AWS Private CA zur Verbesserung der Überprüfbarkeit und zur Einhaltung der Vorschriften nutzen.

> **Note**
>
> Wenn Sie Amazon EKS verwenden, empfehlen wir Ihnen, die `aws-privateca-connector-for-kubernetes` Add-Ons `cert-manager` und Addons für eine verwaltete Installation zu verwenden. Weitere Informationen finden Sie unter AWS Add-Ons.

Konzepte

Das folgende Diagramm zeigt einige der Optionen, die für die Verwendung von TLS in einem Amazon EKS-Cluster verfügbar sind. Der Beispielcluster befindet sich hinter einem Load Balancer. Die Zahlen identifizieren mögliche Endpunkte für TLS-gesicherte Kommunikation.



1. Kündigung am Load Balancer

   Elastic Load Balancing (Elastic Load Balancing) ist in den AWS Certificate Manager Service integriert. Sie müssen es nicht `cert-manager` auf dem Load Balancer installieren. Sie können ACM mit einer privaten CA bereitstellen, ein Zertifikat mit der privaten CA signieren und das Zertifikat mit der Elastic Load Balancing Console installieren. AWS Private CA Zertifikate werden automatisch erneuert.

   Als Alternative können Sie einem AWS Nicht-Load-Balancer ein privates Zertifikat zur Verfügung stellen, um TLS zu beenden.

Dies ermöglicht eine verschlüsselte Kommunikation zwischen einem Remote-Client und dem Load Balancer. Daten nach dem Load Balancer werden unverschlüsselt an den Amazon EKS-Cluster übergeben.

2. Kündigung am Kubernetes-Ingress-Controller

   Der Ingress-Controller befindet sich im Amazon EKS-Cluster und fungiert als Load Balancer und Router. Um den Ingress-Controller als Endpunkt des Clusters für die externe Kommunikation zu verwenden, müssen Sie:

   - Installieren Sie beide `cert-manager` und `aws-privateca-issuer`
   - Stellen Sie dem Controller ein privates TLS-Zertifikat von der bereit AWS Private CA.

   Die Kommunikation zwischen dem Load Balancer und dem Ingress Controller ist verschlüsselt, Daten werden unverschlüsselt an die Ressourcen des Clusters übertragen.

3. Kündigung an einem Pod

   Jeder Pod ist eine Gruppe von einem oder mehreren Containern, die sich Speicher- und Netzwerkressourcen teilen. Wenn Sie sowohl als auch installieren `cert-manager` `aws-privateca-issuer` und den Cluster mit einer privaten CA ausstatten, kann Kubernetes bei Bedarf ein signiertes privates TLS-Zertifikat auf Pods installieren. Eine TLS-Verbindung, die an einem Pod endet, ist für andere Pods im Cluster standardmäßig nicht verfügbar.

4. Sichere Kommunikation zwischen Pods.

   Sie können mehrere Pods mit Zertifikaten ausstatten, damit sie miteinander kommunizieren können. Die folgenden Szenarien sind möglich:

   - Bereitstellung mit von Kubernetes generierten selbstsignierten Zertifikaten. Dadurch wird die Kommunikation zwischen Pods gesichert, selbstsignierte Zertifikate erfüllen jedoch nicht die HIPAA- oder FIPS-Anforderungen.
   - Bereitstellung mit Zertifikaten, signiert von. AWS Private CA Dazu müssen sowohl als auch `cert-manager` installiert werden. `aws-privateca-issuer` Kubernetes kann dann nach Bedarf signierte mTLS-Zertifikate auf den Pods installieren.

Überlegungen

Beachten Sie bei der Verwendung AWS Private Certificate Authority mit Kubernetes die folgenden Überlegungen.

Kontoübergreifende Verwendung von Cert-Manager

Administratoren mit kontenübergreifendem Zugriff auf eine Zertifizierungsstelle können das `cert-manager` Add-On für Kubernetes verwenden, um Zertifikate für einen Cluster bereitzustellen, der die gemeinsame CA verwendet. Weitere Informationen finden Sie unter Bewährte Sicherheitsmethoden für den kontoübergreifenden Zugriff auf private CAs.

In kontoübergreifenden Szenarien können Sie nur bestimmte AWS Private CA Zertifikatsvorlagen verwenden.

In der folgenden Tabelle sind AWS Private CA Vorlagen aufgeführt, die Sie mit cert-manager verwenden können, um einen Kubernetes-Cluster bereitzustellen.

Für Kubernetes unterstützte Vorlagen	Support für kontoübergreifende Nutzung
BlankEndEntityCertificate_ /V1-Definition CSRPassthrough	Nein
CodeSigningCertificate/V1-Definition	Nein
EndEntityCertificate/V1-Definition	Ja
EndEntityClientAuthCertificate/V1-Definition	Ja
EndEntityServerAuthCertificate/V1-Definition	Ja
OCSPSigningZertifikat/V1-Definition	Nein

Beginnen Sie mit AWS Private CA Connector for Kubernetes.

Die folgenden Themen zeigen, wie Sie die Kommunikation in einem AWS Private CA Kubernetes-Cluster sichern können. Ein weiteres Beispiel finden Sie unter Encryption in transit for Kubernetes on. GitHub

Sie können eine private Zertifizierungsstelle verwenden, um die Kommunikation mit Ihren Amazon EKS-Clustern zu sichern. Beginnen Sie erst, wenn Folgendes vorliegt:

- Ein AWS Konto mit entsprechenden Berechtigungen im Rahmen Ihrer Sicherheitsrichtlinien.

Amazon EKS clusters

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAM",
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:AttachRolePolicy",
        "iam:GetRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "EKS",
      "Effect": "Allow",
      "Action": [
        "eks:CreateAddon",
        "eks:DescribeAddon",
        "eks:CreatePodIdentityAssociation",
        "eks:DescribeCluster"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::*:role/CertManagerPrivateCARole"
    }
  ]
}
```

Other clusters

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GetAndIssuePCACertificates",
      "Effect": "Allow",
      "Action": [
        "acm-pca:GetCertificate",
        "acm-pca:IssueCertificate"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RolesAnywhere",
      "Effect": "Allow",
      "Action": [
        "rolesanywhere:CreateProfile"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAM",
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:AttachRolePolicy"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::*:role/CertManagerPrivateCARole"
    }
  ]
}
```

- Ein Kubernetes-Cluster. Informationen zum Erstellen eines Amazon Elastic Kubernetes Service Service-Clusters finden Sie in der Amazon EKS-Schnellstartanleitung. Der Einfachheit halber erstellen Sie eine Umgebungsvariable, die den Clusternamen enthält:

  ```
  export CLUSTER=aws-privateca-demo
  ```

- Der AWS-Region Ort, an dem sich Ihr CA- und Amazon EKS-Cluster befinden. Erstellen Sie der Einfachheit halber eine Umgebungsvariable für die Region:

  ```
  export REGION=aws-region
  ```

- Der Amazon-Ressourcenname (ARN) einer AWS Private CA privaten Zertifizierungsstelle. Erstellen Sie der Einfachheit halber eine Umgebungsvariable, die den privaten CA-ARN ARN:

  ```
  export CA_ARN="arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
  certificate/certificate_ID"
  ```

  Informationen zum Erstellen einer privaten Zertifizierungsstelle finden Sie unter https://docs.aws.amazon.com/privateca/latest/userguide/create-CA.html Erstellen einer privaten Zertifizierungsstelle in AWS Private CA

- Ein Computer, auf dem die folgende Software installiert ist:
  - AWS CLI v2 konfiguriert
  - kubectl v1.13+
  - Helm v3 für Nicht-Amazon-EKS-Cluster

Installieren Sie Cert-Manager

Um eine private Zertifizierungsstelle zu verwenden, müssen Sie das `cert-manager>` Add-on installieren, das Zertifikate anfordert, verteilt und die Zertifikatserneuerung automatisiert. Sie müssen auch das `aws-private-ca-issuer` Plugin installieren, mit dem Sie private Zertifikate ausstellen können. AWS Private CA Gehen Sie wie folgt vor, um das Add-on und das Plugin zu installieren.

Amazon EKS clusters

`cert-manager`Als Amazon EKS-Add-on installieren:

```
aws eks create-addon \
```

```
  --cluster-name $CLUSTER \
  --addon-name cert-manager \
  --region $REGION
```

Other clusters

Installation `cert-manager` mit Helm:

```
helm repo add jetstack https://charts.jetstack.io
helm repo update

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --set crds.enabled=true
```

Konfigurieren Sie die IAM-Berechtigungen.

Das `aws-privateca-issuer` Plugin benötigt die Erlaubnis, mit dem Sie interagieren können AWS Private CA. Für Amazon EKS-Cluster verwenden Sie die Pod-Identität. Für andere Cluster verwenden Sie AWS Identity and Access Management Roles Anywhere.

Erstellen Sie zunächst eine IAM-Richtlinie. Die Richtlinie verwendet die `AWSPrivateCAConnectorForKubernetesPolicy` verwaltete Richtlinie. Weitere Informationen zu der Richtlinie finden Sie AWSPrivateCAConnectorForKubernetesPolicyim Referenzhandbuch für AWS verwaltete Richtlinien.

Amazon EKS clusters

1. Erstellen Sie eine Datei `trust-policy.json` mit dem Namen, die die folgende Vertrauensrichtlinie enthält:

   JSON

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

```
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

2. Führen Sie die folgenden Befehle aus, um eine IAM-Rolle zu erstellen:

```
ROLE_ARN=$(aws iam create-role \
  --role-name CertManagerPrivateCARole \
  --assume-role-policy-document file://trust-policy.json \
  --region $REGION \
  --output text \
  --query "Role.Arn")

 aws iam attach-role-policy \
  --role-name CertManagerPrivateCARole \
  --policy-arn arn:aws:iam::aws:policy/AWSPrivateCAConnectorForKubernetesPolicy
```

Other clusters

1. Erstellen Sie einen Vertrauensanker, der der privaten CA vertraut, die in gespeichert ist. `CA_ARN` Anweisungen finden Sie unter Erste Schritte mit IAM Roles Anywhere. Erstellen Sie eine Umgebungsvariable, um den Vertrauensanker-ARN zu speichern:

```
export TRUST_ANCHOR_ARN=trustAnchorArn
```

2. Erstellen Sie eine Datei mit dem Name`trust-policy.json`, die die folgende Vertrauensrichtlinie enthält:

   JSON

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

```
        {
            "Sid": "TrustPolicyForSelfManagedOrOnPremiseClusters",
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity",
                "sts:TagSession"
            ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": [
                        "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TRUST_ANCHOR_ARN"
                    ]
                },
                "StringEquals": {
                    "aws:PrincipalTag/x509Subject/CN": "aws-privateca-
issuer"
                }
            }
        }
    ]
}
```

3. Führen Sie die folgenden Befehle aus, um eine IAM-Rolle zu erstellen:

```
ROLE_ARN=$(aws iam create-role \
  --role-name CertManagerPrivateCARole \
  --assume-role-policy-document file://trust-policy.json \
  --query "Role.Arn" \
  --region $REGION \
  --output text)

 aws iam attach-role-policy \
  --role-name CertManagerPrivateCARole \
  --region $REGION \
  --policy-arn arn:aws:iam::aws:policy/AWSPrivateCAConnectorForKubernetesPolicy
```

Installieren und konfigurieren Sie den AWS Private CA Cluster-Issuer

Verwenden Sie die folgenden Befehle, um das `aws-privateca-connector-for-kubernetes` Add-on zu installieren:

Amazon EKS clusters

Erstellen Sie das Add-on:

```
aws eks create-addon --region $REGION \
  --cluster-name $CLUSTER \
  --addon-name aws-privateca-connector-for-kubernetes \
  --pod-identity-associations "[{
    \"serviceAccount\": \"aws-privateca-issuer\",
    \"roleArn\": \"$ROLE_ARN\"
  }]"
```

Warten Sie dann, bis das Add-on aktiv ist:

```
aws eks describe-addon \
  --cluster-name $CLUSTER \
  --addon-name aws-privateca-connector-for-kubernetes \
  --region $REGION \
  --query 'addon.status'
```

Other clusters

1. Erstellen Sie ein Profil in IAM Roles Anywhere:

   ```
   PROFILE_ARN=$(aws rolesanywhere create-profile \
     --name "privateca-profile" \
     --role-arns "$ROLE_ARN" \
     --region "$REGION" \
     --query 'profile.profileArn' \
     --enabled \
     --output text)
   ```

2. Generieren Sie ein Client-Zertifikat zur Verwendung mit dem Connector für Kubernetes und IAM Roles Anywhere zur Authentifizierung mit: AWS Private CA

   a. Generieren Sie einen privaten Schlüssel für das Client-Zertifikat:

```
openssl genrsa -out client.key 2048
```

b. Generieren Sie eine Zertifikatsignieranforderung (CSR) für das Client-Zertifikat:

```
openssl req -new \
  -key client.key \
  -out client.csr \
  -subj "/CN=aws-privateca-issuer"
```

c. Stellen Sie das Client-Zertifikat aus von AWS Private CA:

```
CERT_ARN=$(aws acm-pca issue-certificate \
  --signing-algorithm SHA256WITHRSA \
  --csr fileb://client.csr \
  --validity Value=1,Type=DAYS \
  --certificate-authority-arn "$CA_ARN" \
  --region "$REGION" \
  --query 'CertificateArn' \
  --output text)
```

d. Speichern Sie das Client-Zertifikat lokal:

```
aws acm-pca get-certificate \
  --certificate-authority-arn $CA_ARN \
  --certificate-arn $CERT_ARN \
  --region $REGION \
  --query 'Certificate'
  --output text > pca-issuer-client-cert.pem
```

3. Installieren Sie den AWS Private CA Aussteller im Cluster mit dem Client-Zertifikat:

a. Fügen Sie das `awspca`-Helm-Repository hinzu:

```
helm repo add awspca https://cert-manager.github.io/aws-privateca-issuer
helm repo update
```

b. Erstellen Sie einen Namespace:

```
kubectl create namespace aws-privateca-issuer
```

c. Verwahren Sie das zuvor erstellte Zertifikat in ein Geheimnis:

```
kubectl create secret tls aws-privateca-credentials \
  -n aws-privateca-issuer \
  --cert=pca-issuer-client-cert.pem \
  --key=client.key
```

4. Installieren Sie den AWS Private CA Aussteller mit IAM Roles Anywhere:

   a. Erstellen Sie eine Datei mit dem Namen`values.yaml`, um das AWS Private CA Issuer-Plugin für die Verwendung mit folgenden Komponenten zu konfigurieren: IAM Roles Anywhere

```
cat > values.yaml <<EOF
env:
  AWS_EC2_METADATA_SERVICE_ENDPOINT: "http://127.0.0.1:9911"

extraContainers:
  - name: "rolesanywhere-credential-helper"
    image: "public.ecr.aws/rolesanywhere/credential-helper:latest"
    command: ["aws_signing_helper"]
    args:
      - "serve"
      - "--private-key"
      - "/etc/cert/tls.key"
      - "--certificate"
      - "/etc/cert/tls.crt"
      - "--role-arn"
      - "$ROLE_ARN"
      - "--profile-arn"
      - "$PROFILE_ARN"
      - "--trust-anchor-arn"
      - "$TRUST_ANCHOR_ARN"
    volumeMounts:
      - name: cert
        mountPath: /etc/cert/
        readOnly: true

volumes:
  - name: cert
    secret:
      secretName: aws-privateca-credentials
EOF
```

b. Installieren Sie den AWS Private CA Emittenten mit: IAM Roles Anywhere

```
helm install aws-privateca-issuer awspca/aws-privateca-issuer \
  -n aws-privateca-issuer \
  -f values.yaml
```

Warten Sie, bis der Emittent bereit ist. Verwenden Sie den folgenden Befehl:

```
kubectl wait --for=condition=ready pods --all -n aws-privateca-issuer --timeout=120s
```

Überprüfen Sie anschließend die Installation, um sicherzustellen, dass alle Pods den folgenden `READY` Status erreicht haben:

```
kubectl -n aws-privateca-issuer get all
```

Um das zu konfigurieren`aws-private-ca-cluster-issuer`, erstellen Sie eine YAML-Datei `cluster-issuer.yaml` mit dem Namen, die die Konfiguration des Ausstellers enthält:

```
cat > cluster-issuer.yaml <<EOF
apiVersion: awspca.cert-manager.io/v1beta1
kind: AWSPCAClusterIssuer
metadata:
  name: aws-privateca-cluster-issuer
spec:
  arn: "$CA_ARN"
  region: "$REGION"
EOF
```

Wenden Sie als Nächstes die Cluster-Konfiguration an:

```
kubectl apply -f cluster-issuer.yaml
```

Überprüfen Sie den Status des Emittenten:

```
kubectl describe awspcaclusterissuer aws-privateca-cluster-issuer
```

Es wird eine Antwort ähnlich der folgenden angezeigt:

```
Status:
```

```
  Conditions:
    Last Transition Time:  2025-08-13T21:00:00Z
    Message:               AWS PCA Issuer is ready
    Reason:                Verified
    Status:                True
    Type:                  Ready
```

Verwalten Sie das AWS Private CA Client-Zertifikat mit cert-manager

Wenn Sie keinen Amazon EKS-Cluster verwenden, können `aws-privateca-issuer` Sie nach dem manuellen Bootstrap eines vertrauenswürdigen Zertifikats zu einem Client-Authentifizierungszertifikat wechseln, das von verwaltet wird`cert-manager`. Dadurch kann `cert-manager` das Client-Authentifizierungszertifikat automatisch erneuert werden.

1. Erstellen Sie eine Datei mit dem Namen`pca-auth-cert.yaml`:

   ```
   cat > pca-auth-cert.yaml <<EOF
   apiVersion: cert-manager.io/v1
   kind: Certificate
   metadata:
     name: aws-privateca-client-cert
     namespace: aws-privateca-issuer
   spec:
     secretName: aws-privateca-credentials
     duration: 168h
     renewBefore: 48h
     commonName: aws-privateca-issuer
     privateKey:
       algorithm: ECDSA
       size: 256
       rotationPolicy: Always
     usages:
       - client auth
     issuerRef:
       name: aws-privateca-cluster-issuer
       kind: AWSPCAClusterIssuer
       group: awspca.cert-manager.io
   EOF
   ```

2. Erstellen Sie das neue verwaltete Client-Authentifizierungszertifikat:

   ```
   kubectl apply -f pca-auth-cert.yaml
   ```

3. Stellen Sie sicher, dass das Zertifikat erstellt wurde:

```
kubectl get certificate aws-privateca-client-cert -n aws-privateca-issuer
```

Es wird eine Antwort ähnlich der folgenden angezeigt:

```
NAME                        READY   SECRET                       AGE
aws-privateca-client-cert   True    aws-privateca-credentials   19m
```

Stellen Sie Ihr erstes TLS-Zertifikat aus

Nachdem die `cert-manager` und installiert `aws-privateca-issuer` sind, können Sie ein Zertifikat ausstellen.

Erstellen Sie eine YAML-Datei `certificate.yaml` mit dem Namen, die die Zertifikatsressource enthält:

```
cat > certificate.yaml <<EOF
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: example-certificate
  namespace: default
spec:
  secretName: example-certificate-tls
  issuerRef:
    name: aws-privateca-cluster-issuer
    kind: AWSPCAClusterIssuer
    group: awspca.cert-manager.io
  commonName: example.internal
  dnsNames:
    - example.internal
    - api.example.internal
  duration: 2160h # 90 days
  renewBefore: 360h # 15 days
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

Wenden Sie das Zertifikat mit dem folgenden Befehl an:

```
kubectl apply -f certificate.yaml
```

Anschließend können Sie den Status des Zertifikats mit den folgenden Befehlen überprüfen:

```
kubectl get certificate example-certificate
kubectl describe certificate example-certificate
```

Sie sollten eine Antwort ähnlich der folgenden sehen:

```
NAME                  READY   SECRET                     AGE
example-certificate   True    example-certificate-tls    30s
```

Sie können das ausgestellte Zertifikat mit dem folgenden Befehl überprüfen:

```
kubectl get secret example-certificate-tls -o yaml
```

Sie können das Zertifikat auch mit dem folgenden Befehl dekodieren und überprüfen:

```
kubectl get secret example-certificate-tls -o jsonpath='{.data.tls\.crt}' | base64 -d |
 openssl x509 -text -noout
```

Beispiele

Die folgenden Beispiele zeigen, wie es mit AWS Private CA Kubernetes-Clustern verwendet werden kann.

- Beispiel: Verschlüsselung bei der Übertragung für Kubernetes
- TLS-fähige Kubernetes-Cluster mit und Amazon EKS AWS Private CA
- Einrichtung der end-to-end TLS-Verschlüsselung auf Amazon EKS mit dem neuen Load AWS Balancer Controller

Überwachen Sie Kubernetes mit AWS Private CA

Verwenden Sie die unter beschriebenen Techniken, um die private Zertifizierungsstelle des Kubernetes-Clusters zu überwachen. AWS Private CA Ressourcen überwachen Sie können Folgendes verwenden, um eine private CA zu überwachen:

- AWS Private CA CloudWatch Metriken
- Überwachung AWS Private CA mit CloudWatch Ereignissen
- Protokollieren von AWS Private Certificate Authority API-Aufrufen mit AWS CloudTrail

Fehlerbehebung bei Kubernetes mit AWS Private CA

Sie können die Protokolle für `aws-private-ca-issuer` mit dem folgenden Verfahren abrufen:

1. Ermitteln Sie den Namen des Pods:

```
kubectl get pods -A
```

2. Verwenden Sie den folgenden Befehl, um die Issuer-Logs einzusehen:

```
kubectl logs -n aws-privateca-issuer <pod-name> aws-privateca-issuer
```

3. Verwenden Sie den folgenden Befehl, um die IAM Roles Anywhere Protokolle anzuzeigen:

```
kubectl logs -n aws-privateca-issuer <pod-name> rolesanywhere-credentials-helper
```

Verwenden Sie eine der folgenden Methoden, um den Status Ihres AWS Private CA Emittenten zu überprüfen:

Verwenden Sie den folgenden Befehl, um zu überprüfen, ob Ihr Emittent bereit ist:

```
kubectl get AWSPCAClusterIssuers -o json | jq '.items[].status
```

Die Antwort sollte der folgenden ähneln:

```
{
  "conditions": [
    {
      "lastTransitionTime": "2024-07-03T13:56:37Z",
      "message": "Issuer verified",
      "reason": "Verified",
      "status": "True",
      "type": "Ready"
    }
  ]
```

```
}
```

Wenn sich der Emittent nicht in dem `Ready` Bundesstaat befindet, enthält das `message` Feld Informationen darüber, warum der Emittent den Bundesstaat nicht erreichen konnte. `Ready`

Verwenden Sie den folgenden Befehl, um zu überprüfen, ob Ihr Zertifikat bereit ist:

```
kubectl get certificates -o json | jq '.items[].status'
```

Die Antwort sollte der folgenden ähneln:

```
{
  "conditions": [
    {
      "lastTransitionTime": "2024-07-03T13:58:13Z",
      "message": "Certificate is up to date and has not expired",
      "observedGeneration": 1,
      "reason": "Ready",
      "status": "True",
      "type": "Ready"
    }
  ],
  "notAfter": "2024-10-01T13:58:12Z",
  "notBefore": "2024-07-03T12:58:12Z",
  "renewalTime": "2024-09-16T13:58:12Z",
  "revision": 1
}
```

Wenn sich das Zertifikat nicht im `Ready` Bundesstaat befindet, enthält das `message` Feld Informationen darüber, warum das Zertifikat den `Ready` Bundesstaat nicht erreichen konnte.

AWS Private CA Konnektor für Active Directory

AWS Private CA kann Zertifikate ausstellen und verwalten, die für erforderlich sind AWS Managed Microsoft AD. Mithilfe des AWS Private CA Connectors für Active Directory (Connector für AD) können Sie ein lokales Unternehmen oder einen anderen Drittanbieter CAs durch eine verwaltete private Zertifizierungsstelle ersetzen, die Ihnen gehört, sodass Benutzer, Gruppen und Maschinen, die von Ihrem AD verwaltet werden, die Zertifikatsregistrierung erhalten.

Sie können den Connector für AD verwenden, um die lokale Infrastruktur AWS Managed Microsoft AD zu eliminieren, indem Sie Ihr AD und die Public-Key-Infrastruktur in die Cloud migrieren. Für Kunden, die ihr lokales AD verwenden AWS Private CA möchten, lässt sich diese Funktion auch in Connector integrieren. AWS Managed Microsoft AD

Topics

Sind Sie zum ersten Mal ein Connector für AD-Benutzer?

Wenn Sie Connector for AD zum ersten Mal verwenden, empfehlen wir Ihnen, zunächst die folgenden Abschnitte zu lesen:

- Was ist AWS Private CA?
- Was ist Directory Service?

Access Connector für AD

Sie können über die Konsole,, AWS CLI und auf Connector for AD zugreifen APIs. Sie können von der Konsole oder von Ihrer Konsole aus auf den AWS Private CA Connector in der Directory Service

Konsole zugreifen oder indem Sie in der AWS-Managementkonsole Suchleiste nach Connector for AD suchen.

Preisgestaltung

Connector für AD wird als Funktion ohne zusätzliche Kosten angeboten. AWS Private CA Sie zahlen nur für die privaten Zertifizierungsstellen und die Zertifikate, die Sie über diese ausstellen.

Die neuesten AWS Private CA Preisinformationen finden Sie unter AWS Private Certificate Authority Preise. Sie können auch den AWS Preisrechner verwenden, um die Kosten zu schätzen.

Connector für AD einrichten

Die Schritte in diesem Abschnitt sind Voraussetzungen für die Verwendung von Connector for AD. Es wird davon ausgegangen, dass Sie bereits ein AWS Konto erstellt haben. Nachdem Sie die Schritte auf dieser Seite abgeschlossen haben, können Sie mit der Erstellung eines Connectors für AD beginnen.

Schritt 1: Erstellen Sie eine private CA mit AWS Private CA

Richten Sie eine private Zertifizierungsstelle (CA) für die Ausstellung von Zertifikaten für Ihre Verzeichnisobjekte ein. Weitere Informationen finden Sie unter Zertifizierungsstellen in AWS Private CA.

Die private CA muss sich in dem `Active` Status befinden, in dem ein Connector für AD erstellt werden kann. Der Betreffname der privaten CA muss einen allgemeinen Namen enthalten. Die Connectorerstellung schlägt fehl, wenn Sie versuchen, einen Connector mit einer privaten CA ohne gemeinsamen Namen zu erstellen.

Schritt 2: Richten Sie ein Active Directory ein

Zusätzlich zu einer privaten CA benötigen Sie ein Active Directory in einer Virtual Private Cloud (VPC). Connector for AD unterstützt die folgenden Verzeichnistypen, die angeboten werden von Directory Service:

- AWS Verwaltetes Microsoft Active Directory: Mit können Directory Service Sie Microsoft Active Directory (AD) als verwalteten Dienst ausführen. AWS Directory Service for Microsoft Active Directory auch bezeichnet als AWS Managed Microsoft AD, wird von Windows Server 2019

unterstützt. Mit AWS Managed Microsoft AD können Sie verzeichnissensitive Workloads in Microsoft Sharepoint und AWS Cloud benutzerdefinierten.Net- und SQL Server-basierten Anwendungen ausführen.

- Active Directory Connector: AD Connector ist ein Verzeichnisgateway, das Verzeichnisanfragen an Ihr lokales Microsoft Active Directory umleiten kann, ohne Informationen in der Cloud zwischenzuspeichern. AD Connector unterstützt die Verbindung zu einer bei Amazon gehosteten Domain EC2

(Nur Active Directory Connector) Schritt 3: Delegieren Sie Berechtigungen an das Dienstkonto

> **Note**
>
> Wenn Sie AWS Managed Microsoft AD die zusätzlichen Berechtigungen verwenden, werden sie automatisch delegiert, wenn Sie den Connector for AD-Dienst mit Ihrem Verzeichnis autorisieren. Sie können diesen vorausgesetzten Schritt überspringen.

Wenn Sie den Directory Service AD Connector verwenden, müssen Sie zusätzliche Berechtigungen an das Dienstkonto delegieren. Legen Sie für das Dienstkonto eine Zugriffskontrollliste (ACL) fest, um Folgendes zu ermöglichen:

- Fügt einen Service Principal Name (SPN) zu sich selbst hinzu und entfernt ihn
- Erstellen und aktualisieren Sie Zertifizierungsstellen in den folgenden Containern:

```
#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
```

- Erstellen und aktualisieren Sie ein NTAuth Certificate Certification Authority (CA) -Objekt. Hinweis: Wenn das CA-Objekt für NTAuth Zertifikate vorhanden ist, müssen Sie die entsprechenden Berechtigungen delegieren. Wenn das Objekt nicht existiert, müssen Sie die Fähigkeit delegieren, untergeordnete Objekte im Public Key Services-Container zu erstellen.

```
#objects
```

```
CN=NTAuthCertificates,CN=Public Key Services,CN=Services,CN=Configuration
```

Das im offiziellen Connector for Active Directory-Repository verfügbare PowerShell Skript kann verwendet werden, um die zusätzlichen Berechtigungen zu delegieren, die für das Directory Service AD Connector Connector-Dienstkonto erforderlich sind.

Dieses Skript erstellt das Zertifizierungsstellen-Objekt „ NTAuthZertifikate“.

Die neueste Version des Skripts und Einzelheiten zur Verwendung finden Sie in der README-Datei im GitHub Repository.

Schritt 4: IAM-Richtlinie erstellen

Um einen Connector für AD zu erstellen, benötigen Sie eine IAM-Richtlinie, mit der Sie Connector-Ressourcen erstellen, Ihre private CA mit dem Connector for AD-Dienst teilen und den Connector for AD-Dienst mit Ihrem Verzeichnis autorisieren können.

Dies ist ein Beispiel für eine vom Benutzer verwaltete Richtlinie:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "pca-connector-ad:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListCertificateAuthorities",
                "acm-pca:ListTags",
                "acm-pca:PutPolicy"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": "acm-pca:IssueCertificate",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APIPassthrough/V*"
                },
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "pca-connector-ad.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ds:AuthorizeApplication",
                "ds:DescribeDirectories",
                "ds:ListTagsForResource",
                "ds:UnauthorizeApplication",
                "ds:UpdateAuthorizedApplication"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ec2:DeleteTags",
                "ec2:CreateTags"
```

```
            ],
            "Resource": "arn:*:ec2:*:*:vpc-endpoint/*"
        }
    ]
}
```

Connector für AD erfordert zusätzliche AWS RAM Berechtigungen, sowohl für die Konsolen- als auch für die Befehlszeilenverwendung.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ram:CreateResourceShare",
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "ram:Principal": "pca-connector-ad.amazonaws.com",
                    "ram:RequestedResourceType": "acm-pca:CertificateAuthority"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:GetResourcePolicies",
                "ram:GetResourceShareAssociations",
                "ram:GetResourceShares",
                "ram:ListPrincipals",
                "ram:ListResources",
                "ram:ListResourceSharePermissions",
                "ram:ListResourceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

Schritt 5: Teilen Sie Ihre private CA mit Connector for AD

Sie müssen Ihre private CA mithilfe von Service Principal Sharing mit dem AWS Resource Access Manager Connectors-Service teilen.

Wenn Sie in der AWS Konsole einen Connector erstellen, wird die Ressourcenfreigabe automatisch für Sie erstellt.

Wenn Sie mit dem eine Ressourcenfreigabe erstellen AWS CLI, verwenden Sie den AWS RAM create-resource-share Befehl.

Der folgende Befehl erstellt eine Ressourcenfreigabe:

```
$  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
```

Der Dienstprinzipal, der aufruft, CreateConnector verfügt über Berechtigungen zur Ausstellung von Zertifikaten auf der PCA. Um zu verhindern, dass Dienstprinzipale, die Connector für AD verwenden, allgemeinen Zugriff auf Ihre AWS Private CA Ressourcen haben, beschränken Sie ihre Berechtigungen mithilfe von. `CalledVia`

Schritt 6: Erstellen Sie eine Verzeichnisregistrierung

Sie autorisieren den Connector for AD-Dienst mit Ihrem Verzeichnis, sodass der Connector mit Ihrem Verzeichnis kommunizieren kann. Um den Connector for AD-Dienst zu autorisieren, erstellen Sie eine Verzeichnisregistrierung. Weitere Informationen zum Erstellen einer Verzeichnisregistrierung finden Sie unter Verzeichnisregistrierungen verwalten

Schritt 7: Sicherheitsgruppen konfigurieren

Die Kommunikation zwischen Ihrer VPC und dem Connector für AD-Connector erfolgt über AWS PrivateLink eine oder mehrere Sicherheitsgruppen mit eingehenden Regeln, die Port 443 TCP auf Ihrer VPC öffnen. Sie werden nach dieser Sicherheitsgruppe gefragt, wenn Sie einen Connector erstellen. Sie können die Quelle als benutzerdefiniert angeben und den CIDR-Block Ihrer VPC auswählen. Sie können dies weiter einschränken (d. h. IP, CIDR und Sicherheitsgruppen-ID).

Schritt 8: Konfigurieren Sie den Netzwerkzugriff für Verzeichnisobjekte

Verzeichnisobjekte benötigen öffentlichen Internetzugang, um das Online Certificate Status Protocol (OCSP) und die Zertifikatssperrlisten (CRLs) aus den folgenden Domänen zu validieren:

```
*.windowsupdate.com
*.amazontrust.com
```

Erforderliche Mindestzugriffsregeln:

- Für die OCSP- und CRL-Kommunikation erforderlich:

  ```
  TCP 80: (HTTP) to 0.0.0.0/0
  ```

- Erforderlich für Connector for AD:

  ```
  TCP 443: (HTTPS) to 0.0.0.0/0
  ```

- Erforderlich für Active Directory:

  ```
  TCP 88: (Kerberos) to Domain Controller IP range
  TCP/UDP 389/636: (LDAP/LDAPS) to Domain Controller IP range, depending on Domain
   Controller configuration
  TCP/UDP 53: (DNS) to 0.0.0.0/0
  ```

Wenn die Geräte keinen öffentlichen Internetzugang haben, schlägt die Zertifikatsausstellung zeitweise fehl und es wird folgender Fehlercode angezeigt `WS_E_OPERATION_TIMED_OUT.`

> **Note**
>
> Wenn Sie eine Sicherheitsgruppe für eine EC2 Amazon-Instance konfigurieren, muss es sich nicht um dieselbe wie in Schritt 7 handeln.

Beginnen Sie mit AWS Private CA Connector für Active Directory

Mit AWS Private CA Connector for Active Directory können Sie Zertifikate von Ihrer privaten CA für Ihre Active Directory-Objekte zur Authentifizierung und Verschlüsselung ausstellen. Wenn Sie einen

Connector erstellen, AWS Private Certificate Authority erstellt er einen Endpunkt für Sie in Ihrer VPC, damit Ihre Verzeichnisobjekte Zertifikate anfordern können.

Um Zertifikate auszustellen, erstellen Sie einen Connector und AD-kompatible Vorlagen für den Connector. Wenn Sie eine Vorlage erstellen, können Sie Registrierungsberechtigungen für Ihre AD-Gruppen festlegen.

Topics

Bevor Sie beginnen

Das folgende Tutorial führt Sie durch den Prozess der Erstellung eines Connectors für AD und einer Connector-Vorlage. Um diesem Tutorial folgen zu können, müssen Sie zunächst die im Abschnitt aufgeführten Voraussetzungen erfüllen.

Schritt 1: Erstellen Sie einen Konnektor

Informationen zum Erstellen eines Connectors finden Sie unterEinen Connector für Active Directory erstellen.

Schritt 2: Microsoft Active Directory-Richtlinien konfigurieren

Connector for AD kann die Konfiguration des Gruppenrichtlinienobjekts (GPO) des Kunden nicht anzeigen oder verwalten. Das GPO steuert die Weiterleitung von AD-Anfragen an Kunden AWS Private CA oder an andere Authentifizierungs- oder Zertifikatsverkaufsserver. Eine ungültige GPO-Konfiguration kann dazu führen, dass Ihre Anfragen falsch weitergeleitet werden. Es liegt an den Kunden, den Connector für die AD-Konfiguration zu konfigurieren und zu testen.

Gruppenrichtlinien sind einem Connector zugeordnet, und Sie können mehrere Connectors für ein einzelnes AD erstellen. Es liegt an Ihnen, die Zugriffskontrolle für jeden Connector zu verwalten, falls die Gruppenrichtlinienkonfigurationen unterschiedlich sind.

Die Sicherheit der Aufrufe auf der Datenebene hängt von Kerberos und Ihrer VPC-Konfiguration ab. Jeder, der Zugriff auf die VPC hat, kann Datenebenenanrufe tätigen, sofern er beim entsprechenden AD authentifiziert ist. Dies geschieht außerhalb der Grenzen von AWSAuth und die Verwaltung der Autorisierung und Authentifizierung liegt bei Ihnen, dem Kunden.

Verwenden Sie bei der Verwendung den Directory Service enable-ca-enrollment-policy Befehl zur Konfiguration GPOs auf dem Domänencontroller der AWS Managed Microsoft AD Instanz. AWS Managed Microsoft AD

Der folgende Befehl ermöglicht die Registrierung von Domänencontrollern:

```
$  aws ds enable-ca-enrollment-policy \
    --pca-connector-arn MyPcaConnectorAdArn \
    --directory-id MyDirectoryId
```

Wenn Sie AD Connector verwenden, gehen Sie wie folgt vor, um ein Gruppenrichtlinienobjekt zu erstellen, das auf den URI verweist, der beim Erstellen eines Connectors generiert wurde. Dieser Schritt ist erforderlich, um Connector for AD von der Konsole oder der Befehlszeile aus zu verwenden.

Konfigurieren. GPOs

1. Öffnen Sie den Server-Manager auf dem DC
2. Gehen Sie zu Tools und wählen Sie in der oberen rechten Ecke der Konsole die Option Gruppenrichtlinienverwaltung aus.
3. Gehen Sie zu Gesamtstruktur > Domänen. Wählen Sie Ihren Domainnamen aus und klicken Sie mit der rechten Maustaste auf Ihre Domain. Wählen Sie GPO in dieser Domain erstellen aus und verknüpfen Sie es hier... und geben Sie `PCA GPO` als Namen ein.
4. Das neu erstellte Gruppenrichtlinienobjekt wird nun unter Ihrem Domainnamen aufgeführt.
5. Wählen Sie PCA GPO und dann Bearbeiten aus. Wenn ein Dialogfeld mit der Meldung Dies ist ein Link und die Meldung, dass die Änderungen global verbreitet werden, geöffnet wird, bestätigen Sie die Meldung, um fortzufahren. Der Gruppenrichtlinienverwaltungs-Editor sollte geöffnet werden.
6. Gehen Sie im Gruppenrichtlinienverwaltungs-Editor zu Computerkonfiguration > Richtlinien > Windows-Einstellungen > Sicherheitseinstellungen > Richtlinien für öffentliche Schlüssel (wählen Sie den Ordner aus).
7. Gehen Sie zum Objekttyp und wählen Sie Certificate Services Client — Certificate Enrollment Policy

8. Ändern Sie in den Optionen das Konfigurationsmodell auf Aktiviert.
9. Vergewissern Sie sich, dass die Active Directory-Registrierungsrichtlinie aktiviert und aktiviert ist. Wählen Sie Hinzufügen aus.
10. Das Fenster Certificate Enrollment Policy Server sollte geöffnet werden.
11. Geben Sie den Endpunkt des Zertifikatsregistrierungsrichtlinienservers, der bei der Erstellung Ihres Connectors generiert wurde, in das Feld Registrierungsserver-Richtlinien-URI eingeben ein.
12. Belassen Sie den Authentifizierungstyp auf Windows integrated.
13. Wählen Sie „Validieren“. Wählen Sie nach erfolgreicher Überprüfung Hinzufügen aus. Das Dialogfenster wird geschlossen.
14. Gehen Sie zurück zu Certificate Services Client — Certificate Enrollment Policy und aktivieren Sie das Kästchen neben dem neu erstellten Connector, um sicherzustellen, dass es sich bei dem Connector um die Standardregistrierungsrichtlinie handelt
15. Wählen Sie Active Directory-Registrierungsrichtlinie und anschließend Entfernen aus.
16. Wählen Sie im Bestätigungsdialogfeld Ja aus, um die LDAP-basierte Authentifizierung zu löschen.
17. Wählen Sie im Fenster Certificate Services Client > Certificate Enrollment Policy die Optionen Anwenden und OK aus und schließen Sie es.
18. Gehen Sie zum Ordner Public Key Policies und wählen Sie Certificate Services Client — Auto-Enrollment aus.
19. Ändern Sie die Option „Konfigurationsmodell“ auf Aktiviert.
20. Vergewissern Sie sich, dass die Optionen Abgelaufene Zertifikate verlängern und Zertifikate aktualisieren aktiviert sind. Lassen Sie die anderen Einstellungen unverändert.
21. Wählen Sie Anwenden, dann OK und schließen Sie das Dialogfeld.

Konfigurieren Sie als Nächstes die Richtlinien für öffentliche Schlüssel für die Benutzerkonfiguration. Gehen Sie zu Benutzerkonfiguration > Richtlinien > Windows-Einstellungen > Sicherheitseinstellungen > Richtlinien für öffentliche Schlüssel. Folgen Sie den in Schritt 6 bis Schritt 21 beschriebenen Verfahren, um die Richtlinien für öffentliche Schlüssel für die Benutzerkonfiguration zu konfigurieren.

Sobald Sie die Konfiguration GPOs und die Public Key-Richtlinien abgeschlossen haben, fordern Objekte in der Domain Zertifikate von AWS Private CA Connector for AD an und erhalten Zertifikate, die von ausgestellt wurden AWS Private CA.

Schritt 3: Erstellen Sie eine Vorlage

Informationen zum Erstellen einer Vorlage finden Sie unterErstellen Sie eine Connector-Vorlage.

Schritt 4: Microsoft-Gruppenberechtigungen konfigurieren

Informationen zum Konfigurieren Microsoft Microsoft-Gruppenberechtigungen finden Sie unterZugriffskontrolleinträge für Connector für AD-Vorlagen verwalten.

AWS Private CA Konnektoren für Active Directory

In den Verfahren in diesem Abschnitt wird beschrieben, wie Active Directory (AD) -Connectors erstellt, Vorlagen konfiguriert AWS Private CA und Active Directory integriert werden. Sie können diese Vorgänge von der AWS Private CA Connector for AD-Konsole aus, mithilfe des Abschnitts Connector für AD von oder mithilfe der AWS Private CA Connector for AD-API ausführen. AWS CLI

> **Note**
>
> AWS Private CA Connector for AD ist zwar eng integriert AWS Private CA, die beiden Dienste sind jedoch voneinander getrennt APIs. Weitere Informationen finden Sie in der AWS Private Certificate Authority API-Referenz und der AWS Private CA Connector for Active Directory-API-Referenz.

Einen Connector für Active Directory erstellen

Verwenden Sie die folgenden Verfahren, um einen Connector mithilfe der Konsole, der Befehlszeile oder der API für AWS Private CA Connector for Active Directory zu erstellen.

Console

So erstellen Sie einen Connector mithilfe der Konsole

Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Connector for Active Directory-Konsole unter**https://console.aws.amazon.com/pca-connector-ad/home** .

1. Wählen Sie auf der Landingpage zum ersten Mal oder auf der Seite Connectors for Active Directory die Option Connector erstellen aus.

2. Geben Sie auf der Seite „Privaten CA-Connector für Active Directory erstellen“ Informationen im Abschnitt Active Directory ein.

   - Wählen Sie unter Wählen Sie Ihren Active Directory-Typ aus einen der beiden verfügbaren Typen aus:
     - AWS Directory Service for Microsoft Active Directory— Gibt ein Active Directory an, das von verwaltet wird Directory Service.
     - Lokales Active Directory mit AWS AD Connector — Verwendet AD Connector, um auf ein Active Directory zuzugreifen, das Sie lokal hosten.
   - Wählen Sie unter Verzeichnis auswählen Ihr Verzeichnis aus der Liste aus.

     Alternativ können Sie Verzeichnis erstellen wählen, wodurch die Directory Service Konsole in einem neuen Fenster geöffnet wird. Wenn Sie mit der Erstellung eines neuen Verzeichnisses fertig sind, kehren Sie zur AWS Private CA Connector for Active Directory-Konsole zurück und aktualisieren Sie die Verzeichnisliste. Ihr neues Verzeichnis sollte zur Auswahl stehen.

     > **Note**
     >
     > Beachten Sie beim Erstellen eines Verzeichnisses, dass Connector for AD nur die folgenden Verzeichnistypen unterstützt, die in der Directory Service Konsole angeboten werden:
     >
     > - AWS Verwaltetes Microsoft AD
     > - AD Connector

   - Wählen Sie unter Sicherheitsgruppen für VPC-Endpunkt auswählen eine Sicherheitsgruppe aus der Liste aus.

     Alternativ können Sie Sicherheitsgruppe erstellen wählen, wodurch die EC2 Amazon-Konsole mit der Seite Sicherheitsgruppe erstellen in einem neuen Fenster geöffnet wird. Wenn Sie mit der Erstellung einer Sicherheitsgruppe fertig sind, kehren Sie zur AWS Private CA Connector for Active Directory-Konsole zurück und aktualisieren Sie die Liste der Sicherheitsgruppen. Ihre neue Sicherheitsgruppe sollte zur Auswahl stehen.

3. Wählen Sie im Abschnitt IP-Adresstyp eine der folgenden Optionen aus:

   - IPv4- Aktiviert die IPv4 Konnektivität mit dem Dienst. Wählen Sie diese Option nur, wenn alle Subnetze, die Ihr Verzeichnis hosten, IPv4 Adressbereiche haben.

- Dualstack — Ermöglicht beides IPv4 und die IPv6 Konnektivität zum Dienst. Wählen Sie diese Option nur, wenn alle Subnetze, die Ihr Verzeichnis hosten, sowohl über Adressbereiche als auch über Adressbereiche IPv4 verfügen. IPv6

4. Wählen Sie im Abschnitt Private Zertifizierungsstelle eine private Zertifizierungsstelle aus der Liste aus.

   Alternativ können Sie Create Private CA wählen, wodurch die AWS Private CA Konsole mit der Seite Private Zertifizierungsstellen in einem neuen Fenster geöffnet wird. Wenn Sie mit der Erstellung einer CA fertig sind, kehren Sie zur AWS Private CA Connector for Active Directory-Konsole zurück und aktualisieren Sie die Liste der CAs. Ihre neue Zertifizierungsstelle sollte zur Auswahl stehen.

5. Im Bereich Tags — optional können Sie Metadaten auf Ihre AD-Ressource anwenden und entfernen. Tags sind Zeichenkettenpaare zwischen Schlüssel und Wert, wobei der Schlüssel für die Ressource eindeutig sein muss und der Wert optional ist. In dem Bereich werden alle vorhandenen Tags für die Ressource in einer Tabelle angezeigt. Folgende Aktionen werden unterstützt.

   - Wählen Sie „Tags verwalten“, um die Seite „Tags verwalten“ zu öffnen.
   - Wählen Sie Neues Tag hinzufügen, um ein Tag zu erstellen. Füllen Sie das Schlüsselfeld und optional das Feld Wert aus. Wählen Sie Änderungen speichern, um das Tag anzuwenden.
   - Klicken Sie auf die Schaltfläche Entfernen neben einem Tag, um es zum Löschen zu markieren, und wählen Sie zur Bestätigung Änderungen speichern.

6. Nachdem Sie die erforderlichen Informationen eingegeben und Ihre Optionen überprüft haben, wählen Sie Connector erstellen. Dadurch wird die Detailseite Connectors für Active Directory geöffnet, auf der Sie den Status Ihres Connectors bei der Erstellung verfolgen können.

   Nachdem der Prozess der Erstellung eines Connectors abgeschlossen ist, weisen Sie ihm einen Dienstprinzipalnamen zu.

API

Um einen Connector mithilfe der API zu erstellen

Um einen Connector für Active Directory mit der API zu erstellen, verwenden Sie die CreateConnectorAktion in der AWS Private CA Connector for Active Directory-API.

CLI

Um einen Connector mit dem zu erstellen AWS CLI

Um einen Connector für Active Directory mit der CLI zu erstellen, verwenden Sie den Befehl AWS Private CA create-connector im Abschnitt Connector für Active Directory von. AWS CLI

Erstellen Sie eine Connector-Vorlage

Eine Vorlage ist eine Liste von Konfigurationen, die festlegen, wie das Zertifikat nach der Ausstellung aussehen soll und wie der Client mit den Zertifikaten umgehen soll. In den folgenden Verfahren wird erklärt, wie eine Vorlage erstellt wird.

Console

Um eine Vorlage mit der Konsole zu erstellen

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Connector for Active Directory-Konsole unter **https://console.aws.amazon.com/pca-connector-ad/home** .
2. Wählen Sie einen Connector aus der Liste Connectors für Active Directory aus und klicken Sie dann auf Details anzeigen.
3. Suchen Sie auf der Detailseite für den Connector den Abschnitt Vorlagen und wählen Sie dann Vorlage erstellen aus.
4. Wählen Sie auf der Seite Vorlage erstellen im Abschnitt Methode zur Vorlagenerstellung eine der Methodenoptionen aus.

   - Mit einer vordefinierten Vorlage beginnen (Standard) — Wählen Sie aus einer Liste vordefinierter Vorlagen für AD-Anwendungen aus:
     - Codesignatur
     - Computer
     - Domänencontroller-Authentifizierung
     - EFS-Wiederherstellungsagent
     - Registrierungsagent
     - Registrierungsagent (Computer)
     - IPSec

- Kerberos-Authentifizierung
- RAS- und IAS-Server
- Smartcard-Anmeldung
- Signierung von Vertrauenslisten
- Signatur des Benutzers
- Workstation-Authentifizierung

- Beginnen Sie mit einer vorhandenen Vorlage, die Sie erstellt haben — Wählen Sie aus einer Liste von benutzerdefinierten Vorlagen aus, die Sie zuvor erstellt haben.
- Mit einer leeren Vorlage beginnen — Wählen Sie diese Option, um mit der Erstellung einer komplett neuen Vorlage zu beginnen.

5. Definieren Sie im Abschnitt Zertifikatseinstellungen die folgenden Einstellungen für Zertifikate, die auf dieser Vorlage basieren.

- Zertifikatstyp — Geben Sie an, ob Benutzer - oder Computerzertifikate erstellt werden sollen.
- Automatische Registrierung — Wählen Sie aus, ob die automatische Registrierung für Zertifikate aktiviert werden soll, die auf dieser Vorlage basieren.
- Gültigkeitszeitraum — Geben Sie die Gültigkeitsdauer eines Zertifikats als Ganzzahl aus Stunden, Tagen, Wochen, Monaten oder Jahren an. Der Mindestwert beträgt 2 Stunden.
- Verlängerungszeitraum — Geben Sie einen Verlängerungszeitraum für das Zertifikat als Ganzzahl aus Stunden, Tagen, Wochen, Monaten oder Jahren an. Der Verlängerungszeitraum darf nicht mehr als 75% des Gültigkeitszeitraums betragen.
- Betreffname — Wählen Sie auf der Grundlage der in Active Directory enthaltenen Informationen eine oder mehrere Optionen aus, die in den Betreffnamen aufgenommen werden sollen.

> **Note**
>
> Es muss mindestens eine Option für den Betreffnamen oder einen alternativen Betreffnamen angegeben werden.

- Allgemeiner Name
- DNS als allgemeiner Name

- Verzeichnispfad
- E-Mail

- Alternativer Betreffname — Wählen Sie auf der Grundlage der in Active Directory enthaltenen Informationen eine oder mehrere Optionen aus, die in den alternativen Betreffnamen aufgenommen werden sollen.

> **Note**
>
> Es muss mindestens eine Option für den Betreffnamen oder einen alternativen Betreffnamen angegeben werden.

- Verzeichnis-GUID
- DNS-Name
- Domain-DNS
- E-Mail
- Dienstprinzipalname (SPN)
- Benutzerprinzipalname (UPN)

6. Geben Sie im Abschnitt Optionen für die Bearbeitung von Zertifikatsanfragen und die Registrierung den Zweck von Zertifikaten an, die auf der Vorlage basieren, und wählen Sie eine der folgenden Optionen aus.

   - Signature
   - Verschlüsselung
   - Signatur und Verschlüsselung
   - Signatur und Smartcard-Anmeldung

   Wählen Sie als Nächstes aus, welche der folgenden Funktionen aktiviert werden sollen. Die Optionen variieren je nach Verwendungszweck des Zertifikats.

   - Ungültige Zertifikate löschen (nicht archivieren)
   - Schließen Sie symmetrische Algorithmen ein
   - Exportierbarer privater Schlüssel

Wählen Sie abschließend eine Option für die Zertifikatsregistrierung aus. Die Optionen variieren je nach Verwendungszweck des Zertifikats.

- Keine Benutzereingabe erforderlich
- Den Benutzer bei der Registrierung auffordern
- Den Benutzer bei der Registrierung auffordern und Benutzereingabe anfordern

7. Wählen Sie im Abschnitt Anwendungsrichtlinien alle zutreffenden Anwendungsrichtlinien aus. Die verfügbaren Richtlinien sind auf mehreren Seiten aufgelistet. Einige Richtlinien sind möglicherweise aufgrund früherer Einstellungen vorausgewählt.
8. Im Abschnitt Benutzerdefinierte Anwendungsrichtlinien können Sie der Vorlage benutzerdefinierte OIDs Richtlinien hinzufügen und angeben, ob Anwendungsrichtlinienerweiterungen wichtig sind.
9. Wählen Sie im Abschnitt Kryptografieeinstellungen die folgenden Kategorien von Kryptografieeinstellungen für Zertifikate aus, die auf dieser Vorlage basieren.
10. Im Abschnitt Gruppen und Berechtigungen können Sie sich die Vorlagen ansehen, die vorhandenen Gruppen und Berechtigungen für die Registrierung enthalten, oder Sie können auf die Schaltfläche Neue Gruppen und Berechtigungen hinzufügen klicken, um neue Gruppen und Berechtigungen hinzuzufügen. Die Schaltfläche öffnet ein Formular, das die folgenden Informationen benötigt:

    - Anzeigename
    - Sicherheits-ID (SID)
    - Registrieren Sie sich mit den Optionen ALLOW | DENY | NOT SET
    - Automatische Registrierung mit den Optionen ZULASSEN | VERWEIGERN | NICHT GESETZT

11. Im Abschnitt Vorlagen ersetzen können Sie Active Directory darüber informieren, dass die aktuelle Vorlage eine oder mehrere in AD erstellte Vorlagen ersetzt. Wenden Sie die ablösende Vorlage an, indem Sie „Zu ersetzende Vorlage aus Active Directory hinzufügen“ auswählen und den allgemeinen Namen der ablösenden Vorlage angeben.
12. Im Bereich Tags — optional können Sie Metadaten auf Ihre AD-Ressource anwenden und entfernen. Tags sind Zeichenkettenpaare zwischen Schlüssel und Wert, wobei der Schlüssel für die Ressource eindeutig sein muss und der Wert optional ist. In dem Bereich werden alle vorhandenen Tags für die Ressource in einer Tabelle angezeigt. Folgende Aktionen werden unterstützt.

- Wählen Sie „Tags verwalten“, um die Seite „Tags verwalten“ zu öffnen.
- Wählen Sie Neues Tag hinzufügen, um ein Tag zu erstellen. Füllen Sie das Schlüsselfeld und optional das Feld Wert aus. Wählen Sie Änderungen speichern, um das Tag anzuwenden.
- Klicken Sie auf die Schaltfläche Entfernen neben einem Tag, um es zum Löschen zu markieren, und wählen Sie zur Bestätigung Änderungen speichern.

13. Nachdem Sie die erforderlichen Informationen eingegeben und Ihre Auswahl überprüft haben, wählen Sie Vorlage erstellen. Dadurch werden die Vorlagendetails geöffnet, in denen Sie die Einstellungen der neuen Vorlage überprüfen, die Vorlage bearbeiten oder löschen, Gruppen und Berechtigungen verwalten, veraltete Vorlagen verwalten, Tags verwalten und die automatische Neuregistrierung für Zertifikatsinhaber einrichten können.

API

Um eine Connector-Vorlage mithilfe der API zu erstellen

Verwenden Sie die CreateTemplateAktion in der AWS Private CA Connector for Active Directory-API.

CLI

Um eine Connector-Vorlage mit dem zu erstellen AWS CLI

Verwenden Sie den Befehl create-template im Abschnitt AWS Private CA Connector für Active Directory von. AWS CLI

Aktualisieren Sie eine Vorlage für Active Directory

Verwenden Sie die folgenden Verfahren, um eine Vorlage mithilfe der Konsole, der Befehlszeile oder der API für AWS Private CA Connector for Active Directory zu aktualisieren.

Console

Um eine Vorlage mithilfe der Konsole zu aktualisieren

Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Connector for Active Directory-Konsole unter**https://console.aws.amazon.com/pca-connector-ad/home** .

1. Wählen Sie in der Liste Ihrer Connectors für Active Directory den Connector aus, dessen Vorlage Sie aktualisieren möchten. Wählen Sie Bearbeiten, um die Vorlagen des Connectors anzuzeigen und zu ändern.
2. Wählen Sie auf der Seite mit den Vorlagendetails Ihres Connectors die Option Bearbeiten aus. Folgen Sie den Anweisungen, um Ihre Aktualisierungen vorzunehmen. Wenn Sie mit der Bearbeitung eines Bereichs fertig sind, wählen Sie Speichern, um Ihre Änderungen zu speichern.

API

Um eine Vorlage mithilfe der API zu aktualisieren

Um eine Vorlage für Active Directory mit der API zu aktualisieren, verwenden Sie die UpdateTemplateAktion in der AWS Private CA Connector for Active Directory-API.

CLI

Um eine Vorlage mit dem zu aktualisieren AWS CLI

Um einen Connector für Active Directory mit der CLI zu aktualisieren, verwenden Sie den Befehl update-template im Abschnitt AWS Private CA Connector für Active Directory von. AWS CLI

Wie Connector für Active Directory Ihre Vorlagenänderungen weitergibt

AWS Private CA wendet die Vorlage auf Ihre Richtlinie an, wenn Ihr Client den Richtliniencache aktualisiert, was alle acht Stunden der Fall ist. Dies beinhaltet Änderungen an den Zugriffskontrolleinträgen für Vorlagengruppen. Wenn Ihr Client den Cache aktualisiert, fragt er den Connector nach verfügbaren Vorlagen ab. Bei der automatischen Aktualisierung der Registrierung stellt der Client Zertifikate aus, die eine oder beide der folgenden Bedingungen erfüllen:

- Das Zertifikat befindet sich innerhalb des Verlängerungszeitraums.
- Das Zertifikat ist auf dem Client-Gerät nicht vorhanden.

Für die manuelle Aktualisierung fragt der Client den Connector ab, und Sie müssen die Vorlage so einrichten, dass sie ausgestellt wird.

Beim Debuggen können Sie den Richtliniencache manuell leeren, um sofort die Änderungen an der Vorlage zu sehen. Führen Sie dazu den folgenden Powershell-Befehl auf Ihrem Client aus.

```
certutil -f -user -policyserver * -policycache delete
```

Konnektoren für Active Directory auflisten

Sie können die Konsole AWS Private CA Connector für Active Directory verwenden oder AWS CLI um die Connectors aufzulisten, die Sie besitzen.

Console

Um Ihre Connectors mithilfe der Konsole aufzulisten

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Connector for Active Directory-Konsole unter**https://console.aws.amazon.com/pca-connector-ad/home** .
2. Überprüfen Sie die Informationen in der Liste Connectors für Active Directory. Mithilfe der Seitenzahlen oben rechts können Sie durch mehrere Seiten mit Konnektoren navigieren. Jeder Konnektor belegt standardmäßig eine Zeile, in der die folgenden Informationsspalten angezeigt werden.

- Connector-ID — Die eindeutige ID des Connectors.
- Verzeichnisname — Die dem Connector zugeordnete Active Directory-Ressource.
- Konnektorstatus — Konnektorstatus. Mögliche Werte sind: Erstellen | Aktiv | Löschen | Fehlgeschlagen.
- Status des Dienstprinzipalnamens — Status des Dienstprinzipalnamens (SPN), der dem Connector zugeordnet ist. Mögliche Werte sind: Erstellen | Aktiv | Löschen | Fehlgeschlagen.
- Registrierungsstatus des Verzeichnisses — Registrierungsstatus des stellvertretenden Direktors. Mögliche Werte sind: Erstellt | Aktiv | Löschen | Fehlgeschlagen.
- Erstellt am — Zeitstempel bei der Erstellung des Connectors.

Wenn Sie das Zahnradsymbol in der oberen rechten Ecke der Konsole auswählen, können Sie die Anzahl der auf einer Seite angezeigten Konnektoren mithilfe der Einstellung Seitengröße anpassen.

API

Um Ihre Konnektoren aufzulisten, die die API verwenden

Verwenden Sie die ListConnectorsAktion in der AWS Private CA Connector for Active Directory-API.

CLI

Um Ihre Konnektoren mit dem aufzulisten AWS CLI

Verwenden Sie den Befehl list-connectors, um Ihre Steckverbinder aufzulisten.

Connector-Vorlagen auflisten

Sie können die Konsole AWS Private CA Connector für Active Directory verwenden oder AWS CLI Vorlagen für Connectors auflisten, die Sie besitzen. Connector-Vorlagen basieren auf AWS Private CA _BlankEndEntityCertificate_ APIPassthrough /V1-Vorlagen.

Console

Um Ihre Vorlagen mithilfe der Konsole aufzulisten

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Connector for Active Directory-Konsole unter**https://console.aws.amazon.com/pca-connector-ad/home** .
2. Wählen Sie einen Connector aus der Liste Connectors für Active Directory aus und klicken Sie dann auf Details anzeigen.
3. Überprüfen Sie auf der Seite mit den Connector-Details die Informationen im Abschnitt Vorlagen. Mithilfe der Seitenzahlen oben rechts können Sie durch mehrere Seiten mit Vorlagen navigieren. Jede Vorlage nimmt eine Zeile ein, in der die folgenden Informationsspalten angezeigt werden.

- Vorlagenname — Der für Menschen lesbare Name der Vorlage.
- Vorlagenstatus — Status der Vorlage. Mögliche Werte sind: Aktiv | Löschen.
- Vorlagen-ID — Die eindeutige Kennung der Vorlage.

API

Um Ihre Konnektoren mithilfe der API aufzulisten

Verwenden Sie die ListTemplatesAktion in der AWS Private CA Connector for Active Directory-API, um Vorlagen für den angegebenen Connector aufzulisten.

CLI

Um Ihre Konnektoren aufzulisten, verwenden Sie den AWS CLI

Verwenden Sie den Befehl list-templates, um Vorlagen für den angegebenen Konnektor aufzulisten.

Konnektordetails anzeigen

Gehen Sie wie folgt vor, um die Konfigurationsdetails eines Connectors in der Konsole, Befehlszeile oder API für AWS Private CA Connector for Active Directory anzuzeigen.

Console

So zeigen Sie Details für einen Connector mithilfe der Konsole an

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Connector for Active Directory-Konsole unter**https://console.aws.amazon.com/pca-connector-ad/home** .
2. Wählen Sie einen Connector aus der Liste Connectors für Active Directory aus und klicken Sie dann auf Details anzeigen.
3. Überprüfen Sie auf der Seite mit den Connector-Details die Informationen im Bereich Connector-Details, der Folgendes umfasst:

   - Connector-ID
   - Status des Steckverbinders
   - Zusätzliche Statusdetails
   - Anschluss ARN
   - Serverendpunkt für die Zertifikatsregistrierungsrichtlinie
   - Name des Verzeichnisses
   - Verzeichnis-ID
   - AWS Private CA Betreff
   - AWS Private CA status
   - IP-Adresstyp

- VPC-Endpunkt und Sicherheitsgruppen

4. Im Bereich Vorlagen können Sie Vorlagen erstellen oder verwalten, die dem Connector zugeordnet sind.
5. Im Bereich Service Principal Name (SPN) können Sie den mit dem Connector verknüpften Dienstprinzipalnamen anzeigen.
6. Im Bereich Verzeichnisregistrierung können Sie die dem Connector zugeordnete Verzeichnisregistrierung anzeigen oder ändern.
7. Im optionalen Bereich Tags können Sie dem Connector zugeordnete Tags erstellen oder verwalten.

API

Um Ihre Konnektoren mithilfe der API aufzulisten

Verwenden Sie die GetConnectorAktion in der AWS Private CA Connector for Active Directory-API.

CLI

Um Ihre Konnektoren mit dem aufzulisten AWS CLI

Verwenden Sie den Befehl get-connector im Abschnitt AWS Private CA Connector für Active Directory von. AWS CLI

Details zur Connector-Vorlage anzeigen

Gehen Sie wie folgt vor, um die Konfigurationsdetails einer Connector-Vorlage mithilfe der Konsole, der Befehlszeile oder der API für AWS Private CA Connector for Active Directory anzuzeigen

Console

So zeigen Sie Details für eine Connector-Vorlage mithilfe der Konsole an

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Connector for Active Directory-Konsole unter**https://console.aws.amazon.com/pca-connector-ad/home** .
2. Wählen Sie einen Connector aus der Liste Connectors für Active Directory aus und klicken Sie dann auf Details anzeigen.

3. Überprüfen Sie auf der Seite mit den Connector-Details die Informationen im Abschnitt Vorlagen und wählen Sie die Vorlage aus, die Sie überprüfen möchten. Wählen Sie dann Details anzeigen.
4. Auf der Detailseite werden im Bereich mit den Vorlagendetails die folgenden Informationen zur Vorlage angezeigt:

   - Name der Vorlage
   - Vorlagen-ID
   - Status der Vorlage
   - Version des Vorlagenschemas
   - Version der Vorlage
   - Vorlage ARN
   - Art des Zertifikats
   - Die automatische Registrierung ist aktiviert
   - Gültigkeitszeitraum
   - Verlängerungszeitraum
   - Anforderungen an den Betreffnamen
   - Anforderungen an alternative Namen für den Betreff
   - Einstellungen für Zertifikatsanforderung und Registrierung
   - Kategorie des Kryptografieanbieters
   - Schlüsselalgorithmus
   - Minimale Schlüsselgröße (Bits)
   - Hash-Algorithmus
   - Anbieter von Kryptografie
   - Einstellungen für wichtige Nutzungserweiterungen

   In diesem Bereich können Sie mit den Schaltflächen Bearbeiten, Löschen und Aktionen auch die folgenden Aktionen ausführen.

   - Edit (Bearbeiten)
   - Löschen
   - Gruppen und Berechtigungen verwalten — Weitere Informationen finden Sie unter Gruppen und Berechtigungen konfigurieren.

- Ersetzte Vorlagen verwalten — Weitere Informationen finden Sie unter Überprüfen und Erstellen.
- Stichwörter verwalten — Weitere Informationen finden Sie unter. Connector für AD-Ressourcen taggen
- Alle Zertifikatsinhaber erneut registrieren — Mit dieser Einstellung kann die Hauptversion einer Vorlage automatisch erhöht werden. Alle Mitglieder von Active Directory-Gruppen, die sich mit einer Vorlage registrieren dürfen, erhalten ein neues Zertifikat, das anhand dieser Vorlage ausgestellt wurde. Weitere Informationen finden Sie in der UpdateTemplate-API.

5. Im unteren Bereich wird eine Reihe von Registerkarten angezeigt, auf denen Sie die Konfiguration der Vorlage ändern können.

   - Gruppen und Berechtigungen — Berechtigungen für Active Directory-Gruppen zur Registrierung von Zertifikaten mithilfe dieser Vorlage anzeigen und verwalten. Weitere Informationen finden Sie unter Gruppen und Berechtigungen konfigurieren
   - Anwendungsrichtlinien — Vorlagenrichtlinien für Anwendungen anzeigen und verwalten. Weitere Informationen finden Sie unter Anwendungsrichtlinien zuweisen.
   - Ersetzte Vorlagen — Ersetzte Vorlagen anzeigen und verwalten. Weitere Informationen finden Sie unter Überprüfen und Erstellen.
   - Tag optional — Tagging in dieser Vorlage anzeigen und verwalten. Weitere Informationen finden Sie unter Connector für AD-Ressourcen taggen.

API

Um Ihre Konnektoren aufzulisten, die die API verwenden

Verwenden Sie die GetTemplateAktion in der AWS Private CA Connector for Active Directory-API.

CLI

Um Ihre Konnektoren mit dem aufzulisten AWS CLI

Verwenden Sie den Befehl get-template im Abschnitt AWS Private CA Connector für Active Directory von. AWS CLI

Verzeichnisregistrierungen verwalten

Console

So verwalten Sie Verzeichnisregistrierungen mit der Konsole

Verzeichnisregistrierungen für Connectoren können von der obersten Ebene der AWS Private CA Connector for Active Directory-Konsole aus verwaltet werden. In diesem Thema werden die verfügbaren Verwaltungsoptionen beschrieben.

1. Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die AWS Private CA Connector for Active Directory-Konsole unter**https://console.aws.amazon.com/pca-connector-ad/home** .
2. Wählen Sie im linken Navigationsbereich die Option Verzeichnisregistrierungen aus.
3. Auf der Seite „Verzeichnisregistrierungen“ wird eine Tabelle mit registrierten Verzeichnissen mit den folgenden Feldern angezeigt:

   - Verzeichnis-ID — Die eindeutige ID des Verzeichnisses
   - Verzeichnisname — Der Name der Verzeichnisdomäne
   - Verzeichnistyp
   - Registriert — Der Status der Registrierung. Unterstützte Werte sind CREATING | ACTIVE | DELETING | FAILED.
   - Verzeichnisstatus — Der Status des Verzeichnisses

   Der Benutzer kann das Verzeichnis registrieren verwenden, um eine neue Registrierung zu erstellen.
4. Sie können eine der aufgelisteten Registrierungen auswählen, um sie zu verwalten. Dadurch werden die Schaltflächen Registrierungsdetails anzeigen und Verzeichnis abmelden aktiviert. Mit der Schaltfläche Registrierungsdetails anzeigen wird die Detailseite für die Registrierung geöffnet.
5. Im Bereich mit den Details zur Verzeichnisregistrierung werden die folgenden Informationen angezeigt:

   - Name der Verzeichnisdomäne
   - Verzeichnis-ID — Die eindeutige ID des Verzeichnisses. Wenn Sie den Link wählen, gelangen Sie zur AWS Directory Service Konsole.

- Typ des Verzeichnisses
- Status — Status des Verzeichnisses
- Verzeichnisregistrierung ARN — Der Amazon-Ressourcenname der Verzeichnisregistrierung
- Zusätzliche Statusinformationen

6. Im Bereich Konnektoren und Dienstprinzipalname (SPNs) können Sie SPNs den Connector verwalten. Weitere Informationen finden Sie unter Konnektor-Details anzeigen.
7. Im Bereich Tags — optional können Sie Metadaten auf Ihre AD-Ressource anwenden und entfernen. Tags sind Zeichenkettenpaare zwischen Schlüssel und Wert, wobei der Schlüssel für die Ressource eindeutig sein muss und der Wert optional ist. In dem Bereich werden alle vorhandenen Tags für die Ressource in einer Tabelle angezeigt. Folgende Aktionen werden unterstützt.

   - Wählen Sie „Tags verwalten“, um die Seite „Tags verwalten“ zu öffnen.
   - Wählen Sie Neues Tag hinzufügen, um ein Tag zu erstellen. Füllen Sie das Schlüsselfeld und optional das Feld Wert aus. Wählen Sie Änderungen speichern, um das Tag anzuwenden.
   - Klicken Sie auf die Schaltfläche Entfernen neben einem Tag, um es zum Löschen zu markieren, und wählen Sie zur Bestätigung Änderungen speichern.

API

Um Verzeichnisregistrierungen mithilfe der API zu verwalten

Erstellen: CreateDirectoryRegistrationAktion in der AWS Private CA Connector für die Active Directory-API.

Abrufen: GetDirectoryRegistrationAktion in der AWS Private CA Connector für die Active Directory-API.

Liste: ListDirectoryRegistrationsAktion in der AWS Private CA Connector für die Active Directory-API.

Löschen: DeleteDirectoryRegistrationAktion in der AWS Private CA Connector für die Active Directory-API.

CLI

So verwalten Sie Verzeichnisregistrierungen mit der CLI

Erstellen: Verwenden Sie den create-directory-registrationBefehl im Abschnitt AWS Private CA Connector für Active Directory von AWS CLI.

Rufen Sie den get-directory-registrationBefehl ab: im Abschnitt AWS Private CA Connector für Active Directory von AWS CLI.

Liste: list-directory-registrationsBefehl im Abschnitt AWS Private CA Connector für Active Directory von AWS CLI.

Löschen: delete-directory-registrationBefehl im Abschnitt AWS Private CA Connector für Active Directory von AWS CLI.

Zugriffskontrolleinträge für Connector für AD-Vorlagen verwalten

Ein Zugriffskontrolleintrag ermöglicht die Steuerung, welche Active Directory-Gruppen Zertifikate für eine bestimmte Connector for AD-Vorlage registrieren können oder nicht. Wenn Sie Gruppen und Berechtigungen in Connector for AD erstellen oder verwalten können, müssen Sie die Sicherheits-ID (SID) des Gruppenobjekts aus Active Directory angeben. Sie können die SID mit dem folgenden PowerShell Befehl abrufen. Informationen dazu SIDs finden Sie in der Dokumentation zu Microsoft Directory Domain Services unter So funktionieren Sicherheitskennungen.

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

Die folgenden Verfahren veranschaulichen, wie Gruppeneinträge für den Connector für AD-Vorlagenzugriff erstellt und verwaltet werden.

Console

So verwalten Sie Vorlagengruppenberechtigungen mit der Konsole

Sie können Gruppen verwalten, und die Berechtigungen für eine vorhandene Vorlage können auf der Detailseite einer Vorlage verwaltet werden. Weitere Informationen finden Sie unter Connector-Vorlagendetails anzeigen.

Legen Sie die Berechtigungen fest, mit denen Gruppen Zertifikate für die jeweilige Vorlage registrieren können oder nicht. Sie geben die Sicherheits-ID (SID) der Gruppe an. Anschließend legen Sie die Anmelde- und Autoregistrierungsberechtigungen für die Gruppe fest. Für die automatische Registrierung müssen sowohl die Registrierung als auch die automatische Registrierung auf „Zulassen“ gesetzt sein.

API

Um Berechtigungen für Vorlagengruppen mithilfe der API zu verwalten

Erstellen: CreateTemplateGroupAccessControlEntryAktion in der AWS Private CA Connector für die Active Directory-API.

Update: UpdateTemplateGroupAccessControlEntryAktion im AWS Private CA Connector für die Active Directory-API.

Abrufen: GetTemplateGroupAccessControlEntryAktion in der AWS Private CA Connector für die Active Directory-API.

Liste: ListTemplateGroupAccessControlEntriesAktion in der AWS Private CA Connector für die Active Directory-API.

Löschen: DeleteTemplateGroupAccessControlEntryAktion in der AWS Private CA Connector für die Active Directory-API.

CLI

So verwalten Sie Vorlagengruppenberechtigungen mit der CLI

Erstellen Sie den Befehl create-template-group-access-control-entry im Abschnitt AWS Private CA Connector für Active Directory von. AWS CLI

Update: Befehl update-template-group-access-control-entry im Abschnitt AWS Private CA Connector für Active Directory von. AWS CLI

Rufen Sie den Befehl get-template-group-access-control-entry im Abschnitt AWS Private CA Connector für Active Directory von ab. AWS CLI

Führen Sie den Befehl list-template-group-access-control-entries im Abschnitt AWS Private CA Connector für Active Directory von auf. AWS CLI

Löschen Sie den Befehl: delete-template-group-access-control-entries im Abschnitt AWS Private CA Connector für Active Directory von. AWS CLI

Konfiguration des Dienstprinzipalnamens

Erfahren Sie, wie Sie den Dienstprinzipalnamen für den Connector konfigurieren.

Console

Um Dienstprinzipalnamen mithilfe der Konsole zu verwalten

Der Service Principal Name (SPN) eines vorhandenen AD-Connectors kann auf der Detailseite des Connectors verwaltet werden. Weitere Informationen finden Sie unter Verzeichnisregistrierung verwalten Connectordetails anzeigen

API

Um Dienstprinzipalnamen mithilfe der API zu verwalten

Erstellen: CreateServicePrincipalNameAktion in der AWS Private CA Connector für die Active Directory-API.

Abrufen: GetServicePrincipalNameAktion in der AWS Private CA Connector für die Active Directory-API.

Liste: ListServicePrincipalNamesAktion in der AWS Private CA Connector für die Active Directory-API.

Löschen: DeleteServicePrincipalNameAktion in der AWS Private CA Connector für die Active Directory-API.

CLI

So verwalten Sie Dienstprinzipalnamen mit der CLI

Create: create-service-principal-nameBefehl im Abschnitt AWS Private CA Connector für Active Directory von AWS CLI.

Rufen Sie den get-service-principal-nameBefehl ab: im Abschnitt AWS Private CA Connector für Active Directory von AWS CLI.

Liste: list-service-principal-namesBefehl im Abschnitt AWS Private CA Connector für Active Directory von AWS CLI.

Löschen: delete-service-principal-nameBefehl im Abschnitt AWS Private CA Connector für Active Directory von AWS CLI.

Connector für AD-Ressourcen taggen

Sie können Tags auf Ihre Connectoren, Vorlagen und Verzeichnisregistrierungen anwenden. Durch Tagging werden einer Ressource Metadaten hinzugefügt, die bei der Organisation und Verwaltung helfen können.

Console

Um das Ressourcen-Tagging mit der Konsole zu verwalten

Das Markieren vorhandener Ressourcen wird auf der Detailseite der Ressource verwaltet. Weitere Informationen finden Sie in den folgenden Verfahren:

- Details zur Connector-Vorlage anzeigen
- Verwaltung von Verzeichnisregistrierungen

API

Um das Ressourcen-Tagging mithilfe der API zu verwalten

Tag: TagResourceAktion in der AWS Private CA Connector für die Active Directory-API.

Tags auflisten: ListTagsForResourceAktion in der AWS Private CA Connector für die Active Directory-API.

Untag: UntagResourceAktion in der AWS Private CA Connector für die Active Directory-API.

Wichtig — Die Verwendung von Tags zur Kennzeichnung von Objekten, die vertrauliche Daten enthalten, ist zulässig. Die Tags selbst sollten jedoch keine persönlich identifizierbaren Informationen (PII), sensiblen oder vertraulichen Informationen enthalten.

CLI

So verwalten Sie das Ressourcen-Tagging mit der CLI

Tag: Befehl tag-resource im Abschnitt AWS Private CA Connector für Active Directory von. AWS CLI

Tags auflisten: list-tags-for-resourceBefehl im Abschnitt AWS Private CA Connector für Active Directory von. AWS CLI

Untag: Befehl untag-resource im Abschnitt AWS Private CA Connector für Active Directory von. AWS CLI

Integration von Connector for AD in ereignisgesteuerte Anwendungen mithilfe von Amazon EventBridge

Sie können Connector for AD in ereignisgesteuerte Anwendungen (EDAs) integrieren, die Ereignisse, die in Connector for AD auftreten, verwenden, um zwischen Anwendungskomponenten zu kommunizieren und nachgelagerte Prozesse zu initiieren.

Sie könnten beispielsweise andere AWS Dienste oder benutzerdefinierte Komponenten aufrufen, wenn die folgenden Connector for AD-Ereignisse in Ihrem Konto auftreten:

- Ein Zertifikat wird erstellt oder wenn die Erstellung fehlschlägt.
- Ein Zertifikat ist registriert oder die Registrierung schlägt fehl.

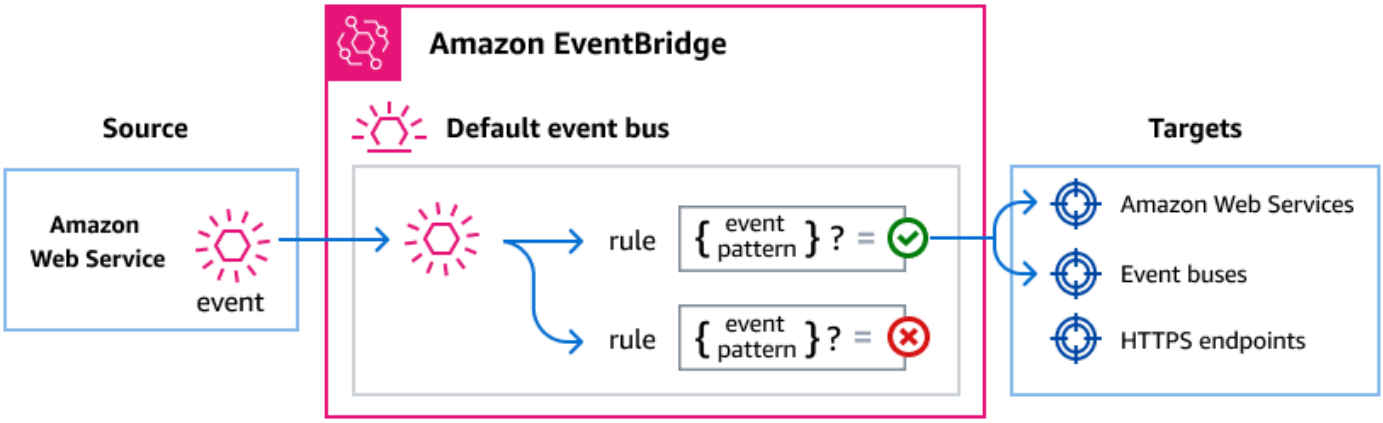
Dazu verwenden Sie Amazon, EventBridge um Ereignisse von Connector for AD an andere Softwarekomponenten weiterzuleiten. Amazon EventBridge ist ein serverloser Service, der Ereignisse verwendet, um Anwendungskomponenten miteinander zu verbinden, sodass Sie AWS Dienste wie Connector for AD einfacher in ereignisgesteuerte Architekturen ohne zusätzlichen Code und Operationen integrieren können.

Wie EventBridge leitet Connector für AD-Ereignisse weiter

So EventBridge funktioniert Connector für AD-Ereignisse:

Wie bei vielen AWS Diensten generiert Connector for AD Ereignisse und sendet sie an den EventBridge Standard-Eventbus. Ein Eventbus ist ein Router, der Ereignisse empfängt und sie an die von Ihnen angegebenen Ziele weiterleitet. Ziele können andere AWS Dienste, benutzerdefinierte Anwendungen und SaaS-Partneranwendungen umfassen.

EventBridge leitet Ereignisse gemäß den Regeln weiter, die Sie im Event-Bus erstellen. Für jede Regel geben Sie einen Filter oder ein Ereignismuster an, um nur die gewünschten Ereignisse auszuwählen. Jedes Mal, wenn ein Ereignis an den Event-Bus gesendet wird, wird es mit den einzelnen Regeln EventBridge verglichen. Wenn das Ereignis der Regel entspricht EventBridge , wird das Ereignis an die angegebenen Ziele weitergeleitet.



Konnektor für AD-Ereignisse

Eine Liste der Connector for AD-Ereignisse, an die gesendet wurden EventBridge, finden Sie im Thema Connector für AD in der EventBridge Ereignisreferenz.

Ereignisstruktur

Alle Ereignisse von AWS Diensten enthalten zwei Arten von Daten:

- Ein allgemeiner Satz von Feldern, die Metadaten über das Ereignis enthalten, z. B. den AWS Dienst, der die Quelle des Ereignisses ist, den Zeitpunkt, zu dem das Ereignis generiert wurde, das Konto und die Region, in der das Ereignis stattgefunden hat, und andere. Definitionen dieser allgemeinen Felder finden Sie unter Event-Struktur in der Amazon EventBridge Events-Referenz.
- Ein `detail` Feld, das Daten enthält, die für dieses spezielle Serviceereignis spezifisch sind.

Erstellen von Ereignismustern, die mit Connector für AD-Ereignissen übereinstimmen

Ereignismuster sind Filter, die angeben, welche Daten die Ereignisse, die Sie auswählen möchten, enthalten sollen.

Jedes Ereignismuster ist ein JSON-Objekt, das Folgendes enthält:

- Ein `source`-Attribut, das den Service identifiziert, der das Ereignis sendet. Für Connector for AD-Ereignisse lautet die Quelle`aws.pca-connector-ad`.

- (Optional): Ein `detail-type` Attribut, das ein Array mit den passenden Ereignisnamen enthält.
- (Optional): Ein `detail`-Attribut, das alle anderen Ereignisdaten für den Abgleich enthält.

Das folgende Ereignismuster würde beispielsweise alle Certificate Policy Enrollment Succeeded-Ereignisse aus Connector for AD auswählen:

```
{
  "source": ["aws.pca-connector-ad"],
  "detail-type": ["Certificate Policy Enrollment Succeeded"]
}
```

Weitere Informationen zum Schreiben von Ereignismustern finden Sie unter Ereignismuster im EventBridge Benutzerhandbuch.

Empfangen von Ereignissen von EventBridge

Sie können Connector für AD-Zertifikate als Ziel für eine Regel angeben. Dadurch kann Connector for AD Ereignisse aus einer Vielzahl von Quellen empfangen, darunter andere AWS Dienste, benutzerdefinierte Anwendungen und SaaS-Partner. Weitere Informationen finden Sie im EventBridge Benutzerhandbuch unter Erstellen von Regeln, die auf Ereignisse reagieren.

Eine vollständige Liste der AWS Dienste, die Sie als Ziele angeben können, finden Sie unter Zieltypen in der EventBridge Ereignisreferenz.

Beheben Sie Probleme mit AWS Private CA Connector für Active Directory

Verwenden Sie die Informationen hier, um Probleme mit AWS Private Certificate Authority Connector for AD zu diagnostizieren und zu beheben.

Themen

- Beheben Sie die Fehlercodes von Connector für AD
- Beheben Sie Fehler bei der Erstellung von Connector for AD Connector
- Beheben Sie den Fehler bei der Erstellung des Connectors für AD SPN
- Beheben Sie Probleme mit der Aktualisierung von Connector für AD-Vorlagen

Beheben Sie die Fehlercodes von Connector für AD

Der Connector für AD sendet aus verschiedenen Gründen Fehlermeldungen. Informationen zu den einzelnen Fehlern und Empfehlungen zu deren Behebung finden Sie in der folgenden Tabelle. Sie können diese Fehler erhalten, wenn Sie Amazon EventBridge Scheduler-Ereignisse abonnieren (Ereignisquelle:`aws.pca-connector-ad`) oder indem Sie die manuelle Registrierung in Windows verwenden.

Fehlercode	Ursache	Abhilfe
0x8FFFA000	Die Kerberos-Authentifizierung ist fehlgeschlagen.	Stellen Sie sicher, dass Ihr Verzeichnis erreichbar ist und der Client entweder ein Benutzer oder ein Computer ist. Wenn Sie die automatische Registrierung verwenden, korrigieren Sie Ihren AWS Resource Service Principal . Wenn Sie die Active Directory-Benutzeroberfläche verwenden, um ein Zertifikat zu erhalten, führen Sie es aus. `gpupdate /force`
0x8FFFA001	Die SOAP-Nachricht muss einen Aktionsheader enthalten.	Fügen Sie einen Aktionsheader hinzu.
0x8FFFA002	Der Connector hat keinen Zugriff auf die private CA, mit der er verbunden ist.	Teilen Sie Ihre private CA mit dem Connector, indem Sie einen AWS Resource Access Manager (RAM) für die gemeinsame Nutzung zwischen Ihrer privaten CA und dem Connector for AD-Dienst erstellen.
0x8FFFA003	Die private CA für diesen Connector ist nicht aktiv.	Versetzen Sie die private CA in den Status Aktiv. Wenn sich Ihre private Zertifizierungsstelle im Status

Fehlercode	Ursache	Abhilfe
		„Ausstehendes Zertifikat“ befindet, installieren Sie das CA-Zertifikat.
0x8FFFA004	Die private CA für diesen Connector ist nicht vorhanden.	Versetzen Sie Ihre Zertifizierungsstelle in den Status Aktiv, wenn sie sich im Status Gelöscht befindet. Wenn Ihre private CA dauerhaft gelöscht ist, erstellen Sie einen neuen Connector mit einer anderen CA.
0x8FFFA005	In der Vorlage wurde das `directoryGuid` Attribut für den Antragsteller des Zertifikats oder der alternative Name des Antragstellers angegeben, aber das Attribut wurde im AD-Objekt für den Antragsteller nicht gefunden.	Active Directory hat keine `directoryGuid` für Ihr Verzeichnis generiert. Problembehandlung in Active Directory.
0x8FFFA006	In der Vorlage wurde das `dnsHostName` Attribut für den Antragsteller des Zertifikats oder der alternative Name des Antragstellers angegeben, aber das Attribut wurde im AD-Objekt für den Antragsteller nicht gefunden.	Fügen Sie das `dnsHostName` Attribut zu Ihrem AD-Objekt hinzu.
0x8FFFA007	In der Vorlage wurde das E-Mail-Attribut angegeben, das in den Betreff des Zertifikats oder in den alternativen Namen des Antragstellers aufgenommen werden soll, aber das Attribut wurde nicht im AD-Objekt für den Antragsteller gefunden.	Fügen Sie das E-Mail-Attribut zu Ihrem AD-Objekt hinzu

Fehlercode	Ursache	Abhilfe
0x8FFFA008	Die SOAP-Nachricht muss den Aktionsheader entweder oder haben. `http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy/IPolicy/GetPolicies http://schemas.microsoft.com/windows/pki/2009/01/enrollment/RST/wstep`	Aktualisieren Sie den Aktionsheader, sodass er einen der angegebenen Werte verwendet.
0x8FFFA009	Das BinarySecurityToken muss codiert sein. `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary`	Aktualisieren Sie den Typ des binären Sicherheitstokens.
0x8FFFA00A	Das ist ungültig BinarySecurityToken .	Vergewissern Sie sich, dass die CSR korrekt generiert wurde.
0x8FFFA00B	Der BinarySecurityToken muss einen Wertetyp von entweder oder haben. `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#PKCS7 http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10`	Aktualisieren Sie den Werttyp des binären Sicherheitstokens auf einen gültigen Wert.

Fehlercode	Ursache	Abhilfe
0x8FFFA00C	Das enthaltene ungültige CMS. BinarySecurityToken	Das Base64-Format ist gültig, aber die kryptografische Nachrichtensyntax (CMS) ist ungültig. Überprüfen Sie die CMS-Syntax.
0x8FFFA00D	Der BinarySecurityToken enthielt eine ungültige CSR.	Vergewissern Sie sich, dass die CSR korrekt generiert wurde.
0x8FFFA00E	Die private Zertifizierungsstelle konnte kein Zertifikat mit der spezifischen Vorlage ausstellen.	Überprüfen Sie die Validierungsausnahme von AWS Private CA. Sie können die Validierungsausnahme in Amazon EventBridge oder einsehen AWS CloudTrail.
0x8FFFA00F	Die SOAP-Nachricht muss den Anforderungstyp haben. `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue`	Stellen Sie den Anforderungstyp auf ein`http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .
0x8FFFA010	Die SOAP-Nachricht muss einen To-Header enthalten, der entweder das Feld des Connectors oder das `CertificateEnrollmentPolicyServerEndpoint` URI-Feld in der XCEP-Antwort enthält.	Setzen Sie den Header des Sicherheits-Tokens für die Anforderung entweder auf das `CertificateEnrollmentPolicyServerEndpoint` Feld oder auf das URI-Feld in der XCEP-Antwort.
0x8FFFA011	Die SOAP-Nachricht darf nur einen Aktionsheader haben.	Überprüfen Sie den SOAP-Nachrichtenheader des Sicherheits-Tokens für die Anforderung und legen Sie den Header korrekt fest.

Fehlercode	Ursache	Abhilfe
0x8FFFA012	Die SOAP-Nachricht darf nur einen Header haben. `messageId`	Überprüfen Sie den SOAP-Nachrichtenheader des Sicherheits-Tokens für die Anforderung und legen Sie den Header korrekt fest.
0x8FFFA013	Die SOAP-Nachricht darf nur einen TO-Header haben.	Überprüfen Sie den SOAP-Nachrichtenheader des Sicherheits-Tokens für die Anforderung und legen Sie den Header korrekt fest.
0x8FFFA014	Der Anforderer hat keinen Zugriff auf die angeforderte Vorlage.	Erlauben Sie der Gruppe des Anfragenden, sich mithilfe der angeforderten Vorlage zu registrieren, indem Sie einen Access Control-Eintrag erstellen.
0x8FFFA015	Entweder die Erweiterung `CertificateTemplateInformation` oder die `CertificateTemplateName` Erweiterung muss in der vorhandenen sein. BinarySecurityToken	Fügen Sie die Sicherheitserweiterung zu Ihrer CSR hinzu.
0x8FFFA016	Die angeforderte Vorlage wurde für den angegebenen Connector nicht gefunden.	Vorlagen sind untergeordnete Ressourcen für jeden Connector . Erstellen Sie die Vorlage für den Konnektor mithilfe von`createTemplate` .
0x8FFFA017	Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.	Verlangsamen Sie die Rate der Anfragen.

Fehlercode	Ursache	Abhilfe
0x8FFFA018	Die SOAP-Nachricht muss einen Header enthalten. `to`	Überprüfen Sie den Header der SOAP-Nachricht.
0x8FFFA019	Die SOAP-Nachricht konnte aufgrund eines unbekannten Headers nicht verarbeitet werden.	Überprüfen Sie den Header der SOAP-Nachricht.
0x8FFFA01A	In der Vorlage wurde angegeben , dass das UPN-Attribut in den Zertifikatsantrag oder den alternati ven Namen des Antragstellers aufgenommen werden soll, aber das Attribut wurde im AD-Objekt für den Antragsteller nicht gefunden.	Fügen Sie dem Active Directory-Objekt einen UPN hinzu.

Beheben Sie Fehler bei der Erstellung von Connector for AD Connector

Die Erstellung eines Connectors für AD-Connectors kann aus verschiedenen Gründen fehlschlagen. Wenn die Erstellung des Connectors fehlschlägt, erhalten Sie den Grund für den Fehler in der API-Antwort. Wenn Sie die Konsole verwenden, wird der Grund für den Fehler auf der Seite mit den Connector-Details im Feld Zusätzliche Statusdetails im Container Connector-Details angezeigt. In der folgenden Tabelle werden die Gründe für Fehler und empfohlene Lösungsschritte beschrieben.

Status des Fehlers	Description	Abhilfe
`CA_CERTIFICATE_REG ISTRATION_FAILED`	Connector for AD kann keine CA-Zertifikate in Ihr Verzeichn is importieren.	Überprüfen Sie die Seite mit den Voraussetzungen und überprüfen Sie, ob Ihr Dienstkonto über die richtigen Berechtigungen verfügt. Nachdem Sie die richtigen Berechtigungen an Ihr

Status des Fehlers	Description	Abhilfe
		Dienstkonto delegiert haben, löschen Sie den ausgefall enen Connector und erstellen Sie einen neuen. Informati onen zum Delegieren von Berechtigungen finden Sie unter Delegieren von Rechten an Ihr Dienstkonto im AWS Directory Service Administr atorhandbuch.

Status des Fehlers	Description	Abhilfe
`DIRECTORY_ACCESS_DENIED`	Der Connector für AD kann nicht auf Ihr Verzeichnis zugreifen.	Sie müssen Connector for AD Zugriff auf Ihr Verzeichnis gewähren. Lesen Sie den Schritt 4: IAM-Richtlinie erstellen Abschnitt, um sicherzustellen, dass die mit Ihrem AWS Konto verknüpfte IAM-Richtlinie Ihnen den Zugriff auf und die Beschreibung von Verzeichnissen ermöglicht. Nachdem Sie Ihrer AWS Rolle die richtigen Berechtigungen erteilt haben, löschen Sie den ausgefallenen Connector und erstellen Sie einen neuen. Wenn Sie Connector for AD mit einem AWS Directory Service AD Connector verwenden, stellen Sie sicher, dass das Passwort des AD Connector Connector-Dienstkontos nicht abgelaufen und gültig ist. Informationen zu AD Connector-Dienstkonten finden Sie unter Erste Schritte mit AD Connector im AD Connector-Administrationshandbuch.

Status des Fehlers	Description	Abhilfe
`INTERNAL_FAILURE`	Beim Connector für AD ist ein interner Fehler aufgetreten.	Bitte versuchen Sie es später erneut. Löschen Sie den ausgefallenen Connector und erstellen Sie einen neuen.
`INSUFFICIENT_FREE_ADDRESSES`	Das VPC-Subnetz muss mindestens eine verfügbare private IP-Adresse haben.	Stellen Sie sicher, dass im Subnetz eine verfügbare private IP-Adresse vorhanden ist. Löschen Sie den ausgefallenen Connector und erstellen Sie einen neuen.
`INVALID_SUBNET_IP_PROTOCOL`	Connector for AD kann den Endpunkt auf Ihrer VPC nicht erstellen, da die mit Ihrem Verzeichnis verknüpften Subnetze den angegebenen IP-Adresstyp nicht unterstützen.	Stellen Sie sicher, dass die VPC und die Subnetze, die Ihr Verzeichnis hosten, den von Ihnen gewählten IP-Adresstyp unterstützen. Weitere Informationen finden Sie unter IP-Adresstypen. Löschen Sie den ausgefallenen Connector und erstellen Sie einen neuen mit dem unterstützten IP-Adresstyp.

Status des Fehlers	Description	Abhilfe
`PRIVATECA_ACCESS_DENIED`	Connector für AD kann nicht auf Ihre private CA zugreifen.	Überprüfen Sie die Seite mit den Voraussetzungen und überprüfen Sie, ob Sie über die erforderlichen Berechtigungen zum Erstellen eines Connectors verfügen. Weitere Informationen finden Sie unter Schritt 4: IAM-Richtlinie erstellen. Wenn Sie einen AWS CLI Connector über unsere API erstellen, überprüfen Sie die Seite mit den Voraussetzungen und überprüfen Sie, ob Sie die private CA mit Connector for AD geteilt haben AWS Resource Access Manager. Nachdem Sie die IAM-Berechtigungen und die gemeinsame Nutzung von AWS RAM Ressourcen überprüft und korrigiert haben, löschen Sie den fehlgeschlagenen Connector und erstellen Sie einen neuen.

Status des Fehlers	Description	Abhilfe
`PRIVATECA_RESOURCE_NOT_FOUND`	Der Connector für AD kann die angegebene private CA nicht finden.	Stellen Sie sicher, dass Sie den richtigen Amazon Resource Name (ARN) der privaten CA angeben, löschen Sie dann den ausgefallenen Connector und erstellen Sie einen neuen mit Ihrem beabsichtigten privaten CA-ARN.
`SECURITY_GROUP_NOT_IN_VPC`	Die Sicherheitsgruppe befindet sich nicht in der VPC, die Ihr Verzeichnis hostet.	Verwenden Sie eine Sicherheitsgruppe, die sich in der VPC befindet, die Ihr Verzeichnis hostet. Weitere Informationen finden Sie unter Schritt 7: Sicherheitsgruppen konfigurieren. Löschen Sie den ausgefallenen Connector und erstellen Sie einen neuen mit einer Sicherheitsgruppe, die sich in der VPC befindet.
`VPC_ACCESS_DENIED`	Connector for AD kann nicht auf die Amazon VPC zugreifen, die Ihr Verzeichnis hostet.	Überprüfen Sie Ihre IAM-Berechtigungen. Löschen Sie den ausgefallenen Connector und erstellen Sie einen neuen. Ein Beispiel für eine IAM-Richtlinie, die Zugriffsberechtigungen beinhaltet, finden Sie unter Schritt 4: IAM-Richtlinie erstellen

Status des Fehlers	Description	Abhilfe
`VPC_ENDPOINT_LIMIT_EXCEEDED`	Connector for AD kann keinen Endpunkt in Ihrer Amazon VPC erstellen. Sie haben das Limit an VPC-Endpunkten erreicht, die Sie für Ihr Konto erstellen können.	Löschen Sie Amazon VPC-Endpunkte oder fordern Sie eine Erhöhung des Limits an. Sobald Sie einen der beiden Schritte ausgeführt haben, löschen Sie den ausgefallenen Connector und erstellen Sie einen neuen. Informationen zu Kontingenten finden Sie unter Amazon Virtual Private Cloud Service-Kontingente.
`VPC_RESOURCE_NOT_FOUND`	Connector für AD kann die angegebene VPC nicht finden.	Stellen Sie sicher, dass Sie die richtige VPC angegeben haben und dass die VPC existiert. Löschen Sie dann den ausgefallenen Connector und erstellen Sie einen neuen mit der richtigen VPC-ID.

Beheben Sie den Fehler bei der Erstellung des Connectors für AD SPN

Die Erstellung des Dienstprinzipalnamens (Service Principal Name, SPN) kann aus verschiedenen Gründen fehlschlagen. Wenn die SPN-Erstellung fehlschlägt, erhalten Sie den Grund für den Fehler in der API-Antwort. Wenn Sie die Konsole verwenden, wird die Fehlerursache auf der Seite mit den Connector-Details im Feld Zusätzliche Statusdetails im Container Service Principal Name (SPN) angezeigt. In der folgenden Tabelle werden die Fehlerursachen und empfohlene Lösungsschritte beschrieben.

Status des Fehlers	Description	Abhilfe
`DIRECTORY_ACCESS_DENIED`	Connector für AD kann nicht auf Ihr Verzeichnis zugreifen.	

Status des Fehlers	Description	Abhilfe
		Gewähren Sie Connector for AD Zugriff auf Ihr Verzeichn is. Ein Beispiel für eine IAM-Richtlinie, die Berechtigungen beinhaltet, die Verzeichn iszugriff gewähren, finden Sie unterSchritt 4: IAM-Richtlinie erstellen.
DIRECTORY_NOT_REAC HABLE	Connector für AD kann nicht auf Ihr Verzeichnis zugreifen.	Überprüfen Sie das Netzwerk zwischen AWS und Ihrem Verzeichnis und versuchen Sie erneut, einen SPN zu erstellen.
DIRECTORY_RESOURCE _NOT_FOUND	Connector für AD kann das angegebene Verzeichnis nicht finden.	Stellen Sie sicher, dass Sie die richtige Verzeichnis-ID angeben, löschen Sie dann den ausgefallenen Connector und erstellen Sie einen neuen mit der gewünschten Verzeichnis-ID.
INTERNAL_FAILURE	Beim Connector für AD ist ein interner Fehler aufgetreten.	Bitte versuchen Sie es später erneut.
SPN_EXISTS_ON_DIFF ERENT_AD_OBJECT	Der Dienstprinzipalname (Service Principal Name, SPN) ist in einem anderen Active Directory-Objekt vorhanden.	Löschen Sie den SPN aus dem Active Directory-Objekt, und versuchen Sie erneut, den SPN zu erstellen.

Status des Fehlers	Description	Abhilfe
SPN_LIMIT_EXCEEDED	Der Connector für AD kann den SPN nicht erstellen, da Sie das Limit von SPNs pro Verzeichnis erreicht haben. Die maximale Anzahl SPNs pro Verzeichnis ist 10.	Löschen Sie einen oder mehrere SPNs aus Ihrem Konto und versuchen Sie erneut, den SPN zu erstellen.

Beheben Sie Probleme mit der Aktualisierung von Connector für AD-Vorlagen

Wenn Sie Änderungen an Ihrer Vorlage oder Ihrem Eintrag für die Gruppenzugriffskontrolle vorgenommen haben, die Änderungen aber nicht angezeigt werden, liegt das möglicherweise an der Zwischenspeicherung von Richtlinien. AWS Private CA wendet die Vorlage auf Ihre Richtlinie an, wenn Ihr Client den Richtliniencache aktualisiert, was alle acht Stunden der Fall ist. Wenn Ihr Client den Cache aktualisiert, fragt er den Connector nach verfügbaren Vorlagen ab. Bei der automatischen Aktualisierung der Registrierung stellt der Client Zertifikate aus, die eine oder beide der folgenden Bedingungen erfüllen:

- Das Zertifikat befindet sich innerhalb des Verlängerungszeitraums.
- Das Zertifikat ist auf dem Client-Gerät nicht vorhanden.

Für die manuelle Aktualisierung fragt der Client den Connector ab, und Sie müssen die Vorlage so einrichten, dass sie ausgestellt wird.

Beim Debuggen können Sie den Richtliniencache manuell leeren, um sofort die Änderungen an der Vorlage zu sehen. Führen Sie dazu den folgenden Powershell-Befehl auf Ihrem Client aus.

```
certutil -f -user -policyserver * -policycache delete
```

AWS Private CA Konnektor für SCEP

Der Connector für Simple Certificate Enrollment Protocol (SCEP) stellt eine Verbindung AWS Private Certificate Authority zu Ihren SCEP-fähigen Mobilgeräten und Netzwerkgeräten her. Mit Connector for SCEP können Sie Zertifikate ausstellen und Ihre AWS Private CA SCEP-Geräte registrieren. Connector for SCEP ist für die Verwendung mit gängigen MDM-Systemen (Mobile Device Management) verfügbar und wurde für die Verwendung mit Clients oder Endpunkten entwickelt, die SCEP unterstützen.

Themen

Features

Support für das SCEP-Protokoll — SCEP ist ein weit verbreitetes Protokoll zum Abrufen digitaler Identitätszertifikate von einer Zertifizierungsstelle (CA) und deren Verteilung an mobile Geräte und Netzwerkgeräte. Sie können Connector for SCEP verwenden, um Ihre Endgeräte mit SCEP zu registrieren.

Registrierung mobiler Geräte — Sie können Connector for SCEP mit gängigen MDM-Systemen wie Microsoft Intune und Jamf Pro verwenden.

Zertifikate in großem Umfang ausstellen — Nachdem Sie Ihre SCEP-fähigen Geräte so konfiguriert haben, dass Zertifikate über den SCEP-Endpunkt des Connectors angefordert werden, können Ihre Clients automatisch Zertifikate von anfordern. AWS Private CA

Wie fange ich mit Connector for SCEP an

Starten Sie zunächst den geführten Assistenten von der Connector for SCEP-Verwaltungskonsole aus. Er hilft Ihnen dabei, einen Connector zu erstellen und die private CA festzulegen, die mit dem Connector verwendet werden soll. Nach Abschluss dieser Schritte stellt Connector for SCEP einen Endpunkt und andere Konfigurationsparameter bereit, die Sie in Ihre MDM-Systeme oder Netzwerkgeräte eingeben können. Nach der Konfiguration Ihrer MDM-Systeme oder Netzwerkgeräte fordern Ihre Kunden automatisch Zertifikate von an. AWS Private CA Weitere Informationen zu den ersten Schritten mit Connector for SCEP finden Sie unter. Beginnen Sie mit Connector für SCEP

Zugehörige Services

Connector für SCEP bezieht sich auf die folgenden AWS Dienste.

- AWS Private Certificate Authority- AWS Private CA bietet Ihnen einen hochverfügbaren privaten CA-Dienst ohne die Vorabinvestitionen und laufenden Wartungskosten, die für den Betrieb Ihrer eigenen privaten CA anfallen.
- AWS Private CA Connector für Active Directory — Connector für AD verbindet Ihr Active Directory (AD) mit. AWS Private CA Der Connector vermittelt den Austausch von Zertifikaten zwischen AWS Private CA Benutzern und Computern, die von Ihrem AD verwaltet werden.

Access Connector für SCEP

Sie können Ihre Connector für SCEP-Konnektoren über eine der folgenden Schnittstellen erstellen, darauf zugreifen und sie verwalten:

- AWS-Managementkonsole- Stellt eine Weboberfläche bereit, über die Sie auf Connector for SCEP zugreifen können. Siehe Connector für die SCEP-Managementkonsole.
- AWS Command Line Interface- Stellt Befehle für eine Vielzahl von AWS Diensten bereit, einschließlich Connector für SCEP. Das AWS CLI wird unter Windows, MacOS und Linux unterstützt. Weitere Informationen finden Sie unter AWS Command Line Interface.
- AWS SDKs- Geben Sie sprachspezifische Informationen an APIs und kümmern Sie sich um viele Verbindungsdetails, z. B. um die Berechnung von Signaturen, die Bearbeitung von

Wiederholungsversuchen von Anfragen und die Fehlerbehandlung. Weitere Informationen finden Sie unter AWS Command Line Interface.

- Konnektor für die SCEP-API — Stellt API-Aktionen auf niedriger Ebene bereit, die Sie mithilfe von HTTPS-Anfragen aufrufen. Die Verwendung des Connectors für die SCEP-API ist der direkteste Weg, auf den Dienst zuzugreifen. Der Connector für die SCEP-API erfordert jedoch, dass Ihre Anwendung grundlegende Details wie die Generierung des Hashs zum Signieren der Anfrage und die Fehlerbehandlung verarbeitet. Weitere Informationen finden Sie unter Referenz zur Connector for SCEP-API.

Preisgestaltung

Der Connector für SCEP wird als Funktion ohne zusätzliche AWS Private CA Kosten angeboten. Sie zahlen nur für AWS Private Certificate Authority Vorgänge und Zertifikate, die zur Erstellung und Aktualisierung von Connectoren verwendet werden.

Die neuesten AWS Private CA Preisinformationen finden Sie unter AWS Private Certificate Authority Preise. Sie können auch den AWS Preisrechner verwenden, um die Kosten zu schätzen.

Konnektor für SCEP-Konzepte

Connector für SCEP ist eine Zusatzfunktion für. AWS Private Certificate Authority

Im Folgenden sind die wichtigsten Konzepte für Connector for SCEP aufgeführt:

Anfrage zum Signieren eines Zertifikats (CSR)

Die erforderlichen Informationen, die einer Zertifizierungsstelle zur Verfügung gestellt werden, um ein digitales Zertifikat ausstellen zu lassen. Diese Informationen enthalten sowohl einen öffentlichen Schlüssel als auch eine Identität.

Passwort herausfordern

Das SCEP-Protokoll verwendet Challenge-Passwörter, um eine Anfrage zu authentifizieren, bevor ein Zertifikat von einer CA ausgestellt wird. Connector für SCEP verarbeitet SCEP-Challenge-Passwörter auf der Grundlage des Connectortyps. Weitere Informationen finden Sie unter Konfigurieren Sie Ihr MDM-System für Connector for SCEP.

Widerruf von Zertifikaten

Der Widerruf eines Zertifikats ist der Vorgang, bei dem ein ausgestelltes Zertifikat vor seinem Ablaufdatum gesperrt wird. Sie können das private CA-Zertifikat, das einem Connector zugeordnet ist, widerrufen, indem Sie die API, das AWS SDK oder AWS CloudFormation aufrufen RevokeCertificate. AWS Command Line Interface

Konnektor für SCEP

Ein Anschluss für SCEP stellt eine Verbindung AWS Private CA zu Ihren SCEP-fähigen Geräten her.

Verwaltung mobiler Geräte

Mobile Device Management (MDM) ermöglicht es IT-Administratoren, Richtlinien auf Smartphones, Tablets und anderen Endpunkten oder Geräten zu kontrollieren, zu sichern und durchzusetzen. Viele MDM-Systeme bieten integrierte Integrationen für die SCEP-basierte Zertifikatsregistrierung.

SCEP

SCEP ist ein standardisiertes Protokoll (RFC 8894) zur automatischen Verteilung von Zertifikaten. Das Protokoll bietet einen Endpunkt, über den Geräte Zertifikate von einer Zertifizierungsstelle anfordern können. SCEP verwendet Challenge-Passwörter, um die Ausstellung von Zertifikaten für Geräte zu autorisieren. SCEP wird häufig für MDM-Systeme (Mobile Device Management) und Netzwerkgeräte eingesetzt. MDM-Lösungen ermöglichen es IT-Administratoren, Richtlinien auf Smartphones, Tablets und anderen Einheiten wie Apple-Workstations zu kontrollieren, zu sichern und durchzusetzen. Die meisten MDM-Lösungen unterstützen SCEP, z. B. Microsoft Intune, Apple MDM und Jamf Pro. Die meisten Netzwerkgeräte wie Router, Loadbalancer, Wi-Fi-Hubs, VPN-Geräte und Firewalls verwenden SCEP für die automatische Zertifikatsregistrierung.

SCEP-Profil

Ein SCEP-Profil enthält Konfigurationsparameter, die zur Definition des Zertifikatsprofils verwendet werden. Dazu gehören die Gültigkeitsdauer des Zertifikats, die Schlüsselgröße, der Name der SCEP-Konfiguration, das Challenge-Passwort, die Anzahl der fehlgeschlagenen Wiederholungsversuche und das Wiederholungsintervall sowie weitere Informationen, die für die Ausstellung von Zertifikaten relevant sind. MDM-Systeme und Zertifikatsverwaltungsplattformen senden das SCEP-Profil in der Regel an den Client, der ein Zertifikat zur Authentifizierung anfordert.

Machen Sie sich mit den Überlegungen und Einschränkungen von Connector for SCEP vertraut

Beachten Sie bei der Verwendung von Connector for SCEP die folgenden Überlegungen und Einschränkungen.

Überlegungen

CA-Betriebsmodi

Sie können Connector for SCEP nur mit privaten Geräten verwenden CAs , die einen allgemeinen Betriebsmodus verwenden. Connector for SCEP stellt standardmäßig Zertifikate mit einer Gültigkeitsdauer von einem Jahr aus. Eine private Zertifizierungsstelle, die einen kurzlebigen Zertifikatsmodus verwendet, unterstützt die Ausstellung von Zertifikaten mit einer Gültigkeitsdauer von mehr als sieben Tagen nicht. Informationen zu den Betriebsmodi finden Sie unterAWS Private CA Verstehen Sie die CA-Modi.

Passwörter herausfordern

- Verteilen Sie Ihre Challenge-Passwörter sehr sorgfältig und geben Sie sie nur an vertrauenswürdige Personen und Kunden weiter. Ein einziges Challenge-Passwort kann verwendet werden, um jedes Zertifikat mit beliebigem Betreff auszustellen SANs, was ein Sicherheitsrisiko darstellt.
- Wenn Sie einen Allzweck-Connector verwenden, empfehlen wir Ihnen, Ihre Challenge-Passwörter häufig manuell zu wechseln.

Konformität mit RFC 8894

Der Connector für SCEP weicht vom RFC 8894-Protokoll ab, indem er HTTPS-Endpunkte anstelle von HTTP-Endpunkten bereitstellt.

CSRs

- Wenn eine Certificate Signing Request (CSR), die an Connector for SCEP gesendet wird, die EKU-Erweiterung (Extended Key Usage) nicht enthält, setzen wir den EKU-Wert auf. `clientAuthentication` Weitere Informationen finden Sie unter 4.2.1.12. Erweiterte Schlüsselverwendung in RFC 5280.

- Wir unterstützen `ValidityPeriod` und passen Attribute `ValidityPeriodUnits` in an. CSRs Wenn Ihre CSR kein Zertifikat enthält`ValidityPeriod`, stellen wir ein Zertifikat mit einer Gültigkeitsdauer von einem Jahr aus. Beachten Sie, dass Sie diese Attribute möglicherweise nicht in Ihrem MDM-System festlegen können. Aber wenn Sie sie einrichten können, unterstützen wir sie. Informationen zu diesen Attributen finden Sie unter Szenrollment_Name_Value_Pair.

Gemeinsame Nutzung von Endpunkten

Verteilen Sie die Endpunkte eines Connectors nur an vertrauenswürdige Parteien. Behandeln Sie die Endpunkte als geheim, da jeder, der Ihren eindeutigen, vollständig qualifizierten Domainnamen und Pfad finden kann, Ihr CA-Zertifikat abrufen kann.

Einschränkungen

Die folgenden Einschränkungen gelten für Connector for SCEP.

Dynamische Herausforderungspasswörter

Sie können statische Challenge-Passwörter nur mit Allzweck-Konnektoren erstellen. Um dynamische Passwörter mit einem Allzweck-Connector zu verwenden, müssen Sie Ihren eigenen Rotationsmechanismus erstellen, der die statischen Passwörter des Connectors verwendet. Connector für SCEP für Microsoft Intune Connectortypen bieten Unterstützung für dynamische Passwörter, die Sie mit Microsoft Intune verwalten.

HTTP

Connector für SCEP unterstützt nur HTTPS und erstellt Weiterleitungen für HTTP-Aufrufe. Wenn Ihr System auf HTTP angewiesen ist, stellen Sie sicher, dass es die HTTP-Umleitungen unterstützt, die Connector für SCEP bereitstellt.

Geteilt, privat CAs

Sie können Connector for SCEP nur mit Private verwenden, CAs dessen Eigentümer Sie sind.

Connector für SCEP einrichten

Die Verfahren in diesem Abschnitt helfen Ihnen bei den ersten Schritten mit Connector for SCEP. Es wird davon ausgegangen, dass Sie bereits ein AWS Konto erstellt haben. Nachdem Sie die Schritte auf dieser Seite abgeschlossen haben, können Sie mit der Erstellung eines Connectors für SCEP fortfahren.

Themen

Schritt 1: Erstellen Sie eine Richtlinie AWS Identity and Access Management

Um einen Connector für SCEP zu erstellen, müssen Sie eine IAM-Richtlinie erstellen, die Connector for SCEP die Möglichkeit gibt, die vom Connector benötigten Ressourcen zu erstellen und zu verwalten und Zertifikate in Ihrem Namen auszustellen. Weitere Informationen zu IAM finden Sie unter Was ist IAM? im IAM-Benutzerhandbuch.

Das folgende Beispiel ist eine vom Kunden verwaltete Richtlinie, die Sie für Connector for SCEP verwenden können.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "pca-connector-scep:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListCertificateAuthorities",
                "acm-pca:ListTags",
                "acm-pca:PutPolicy"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "acm-pca:IssueCertificate",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APICSRPassthrough/V*"
                },
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "pca-connector-scep.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:CreateResourceShare",
                "ram:GetResourcePolicies",
                "ram:GetResourceShareAssociations",
                "ram:GetResourceShares",
                "ram:ListPrincipals",
                "ram:ListResources",
                "ram:ListResourceSharePermissions",
                "ram:ListResourceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

Schritt 2: Erstellen Sie eine private Zertifizierungsstelle

Um Connector für SCEP zu verwenden, müssen Sie dem AWS Private Certificate Authority Connector eine private CA zuordnen. Wir empfehlen, dass Sie eine private Zertifizierungsstelle verwenden, die nur für den Connector vorgesehen ist, da das SCEP-Protokoll inhärente Sicherheitslücken aufweist.

Die private CA muss die folgenden Anforderungen erfüllen:

- Sie muss sich in einem aktiven Zustand befinden und den allgemeinen Betriebsmodus verwenden.

- Sie müssen die private CA besitzen. Sie können keine private Zertifizierungsstelle verwenden, die im Rahmen der kontoübergreifenden Freigabe mit Ihnen geteilt wurde.

Beachten Sie die folgenden Überlegungen, wenn Sie Ihre private CA für die Verwendung mit Connector for SCEP konfigurieren:

- DNS-Namensbeschränkungen — Erwägen Sie die Verwendung von DNS-Namensbeschränkungen, um zu kontrollieren, welche Domänen in den für Ihre SCEP-Geräte ausgestellten Zertifikaten zulässig oder verboten sind. Weitere Informationen finden Sie unter So setzen Sie DNS-Namensbeschränkungen durch in. AWS Private Certificate Authority
- Widerruf — Aktivieren Sie OCSP oder CRLs auf Ihrer privaten CA, um den Widerruf zu ermöglichen. Weitere Informationen finden Sie unter Planen Sie Ihre Methode zum Widerruf von AWS Private CA Zertifikaten.
- PII — Wir empfehlen Ihnen, Ihren CA-Zertifikaten keine personenbezogenen Daten (PII) oder andere vertrauliche oder sensible Informationen hinzuzufügen. Im Falle eines Sicherheitsangriffs trägt dies dazu bei, die Offenlegung vertraulicher Informationen zu begrenzen.
- Stammzertifikate in Vertrauensspeichern speichern — Speichern Sie Ihre Root-CA-Zertifikate in den Vertrauensspeichern Ihres Geräts, damit Sie Zertifikate und die Rückgabewerte von überprüfen können GetCertificateAuthorityCertificate. Informationen zu Trust Stores in Bezug auf AWS Private CA diese finden Sie unterStammzertifizierungsstelle.

Informationen zum Erstellen einer privaten Zertifizierungsstelle finden Sie unterErstellen Sie eine private CA in AWS Private CA.

Schritt 3: Erstellen Sie eine Ressourcenfreigabe mit AWS Resource Access Manager

Wenn Sie Connector for SCEP programmgesteuert mithilfe der AWS Command Line Interface AWS SDK- oder Connector for SCEP-API verwenden, müssen Sie Ihre private CA mithilfe von Service Principal Sharing mit Connector for SCEP teilen. AWS Resource Access Manager Dadurch erhält Connector for SCEP gemeinsamen Zugriff auf Ihre private CA. Wenn Sie in der AWS Konsole einen Connector erstellen, erstellen wir automatisch den Resource Share für Sie. Informationen zur gemeinsamen Nutzung von Ressourcen finden Sie im AWS RAM Benutzerhandbuch unter Erstellen einer Ressourcenfreigabe.

Um eine Ressourcenfreigabe mit dem zu erstellen AWS CLI, können Sie den AWS RAM create-resource-share Befehl verwenden. Der folgende Befehl erstellt eine Ressourcenfreigabe. Geben Sie den ARN der privaten CA, die Sie teilen möchten, als Wert von an*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
```

Der Dienstprinzipal, der aufruft, `CreateConnector` verfügt über Berechtigungen zur Ausstellung von Zertifikaten für die private Zertifizierungsstelle. Um zu verhindern, dass Dienstprinzipale, die Connector für SCEP verwenden, allgemeinen Zugriff auf Ihre AWS Private CA Ressourcen haben, beschränken Sie ihre Berechtigungen mithilfe von. `CalledVia`

Beginnen Sie mit Connector für SCEP

Mit AWS Private Certificate Authority Connector for SCEP können Sie Zertifikate von Ihrer privaten CA für SCEP-fähige Geräte und MDM-Systeme (Mobile Device Management) ausstellen. Wenn Sie einen Connector erstellen, AWS Private Certificate Authority erstellt er eine öffentliche SCEP-URL, über die Sie Zertifikate anfordern können, und stellt Ihnen außerdem Informationen zur Verfügung, die Sie für die Integration in Ihre MDM-Systeme verwenden können.

Um Zertifikate auszustellen, müssen Sie eine AWS Private Certificate Authority private Zertifizierungsstelle und einen Connector erstellen und anschließend Ihre SCEP-fähigen MDM-Systeme und -Geräte so konfigurieren, dass Zertifikate vom Connector angefordert werden.

Themen

Bevor Sie beginnen

Das folgende Tutorial führt Sie durch den Prozess der Erstellung eines Connectors für SCEP.

Um diesem Tutorial zu folgen, benötigen Sie eine private CA und ein SCEP-fähiges Gerät. Außerdem müssen Sie zunächst die im Abschnitt aufgeführten Voraussetzungen erfüllen. Connector für SCEP einrichten

Das folgende Verfahren zeigt Ihnen, wie Sie mithilfe der AWS Konsole einen Connector erstellen.

Aufgaben

- Schritt 1: Erstellen Sie einen Connector
- Schritt 2: Kopieren Sie die Konnektordetails in Ihr MDM-System

Schritt 1: Erstellen Sie einen Connector

Sie erstellen entweder einen Connector für allgemeine Zwecke oder einen Connector für SCEP für Microsoft Intune. Allzweck-Connectors sind für die Verwendung mit SCEP-fähigen Endpunkten konzipiert, und Sie verwalten die SCEP-Challenge-Passwörter. Connector für SCEP for Microsoft Intune sind für die Verwendung mit Microsoft Intune vorgesehen, und Sie verwalten die Challenge-Passwörter mit Microsoft Intune.

General-purpose

Um einen Connector für allgemeine Zwecke zu erstellen

Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die Connector for SCEP-Konsole unter. **https://console.aws.amazon.com/pca-connector-scep/home**

1. Wählen Sie Konnektor erstellen.
2. Geben Sie dem Connector auf der Seite „Connector erstellen“ optional einen benutzerfreundlichen Namen im Feld „Namenstag“. Der Name wird in Ihrer Konnektorliste angezeigt. Wenn Sie möchten, können Sie dem Connector weitere Tags hinzufügen, indem Sie Weitere Tags hinzufügen auswählen. Ein Tag ist eine Bezeichnung, die Sie einer AWS Ressource zuweisen. Jedes Tag besteht aus einem Schlüssel und einem optionalen Wert. Sie können Tags verwenden, um Ihre Ressourcen zu suchen und zu filtern oder Ihre AWS Kosten zu verfolgen.
3. Wählen Sie unter Connectortyp die Option Allzweck aus.

4. Wählen Sie unter Private CA die private CA aus, die mit diesem Connector verwendet werden soll. Oder erstellen Sie eine neue, indem Sie Private CA erstellen auswählen. Aufgrund der inhärenten Sicherheitslücken im SCEP-Protokoll empfehlen wir, eine private CA zu verwenden, die diesem Connector zugewiesen ist. Wenn Sie eine neue Zertifizierungsstelle erstellt haben, kehren Sie nach Abschluss der Erstellung in AWS Private CA zur Connector for SCEP-Konsole zurück und aktualisieren Sie die Liste der privaten Zertifizierungsstellen. CAs Ihre neue private Zertifizierungsstelle sollte zur Auswahl stehen.
5. Wählen Sie unter Challenge-Passwort die Option Challenge-Passwort automatisch generieren Wir generieren ein statisches Challenge-Passwort für Sie, wenn wir diesen Connector erstellen.
6. Wählen Sie Connector erstellen aus.

Microsoft Intune

So erstellen Sie Connector für SCEP für Microsoft Intune

Melden Sie sich bei Ihrem AWS Konto an und öffnen Sie die Connector for SCEP-Konsole unter. **https://console.aws.amazon.com/pca-connector-scep/home**

1. Wählen Sie Konnektor erstellen.
2. Geben Sie dem Connector auf der Seite „Connector erstellen“ optional einen benutzerfreundlichen Namen im Feld „Namenstag“. Der Name wird in Ihrer Konnektorliste angezeigt. Wenn Sie möchten, können Sie dem Connector weitere Tags hinzufügen, indem Sie Weitere Tags hinzufügen auswählen. Ein Tag ist eine Bezeichnung, die Sie einer AWS Ressource zuweisen. Jedes Tag besteht aus einem Schlüssel und einem optionalen Wert. Sie können Tags verwenden, um Ihre Ressourcen zu suchen und zu filtern oder Ihre AWS Kosten zu verfolgen.
3. Wählen Sie unter Connectortyp die Option Microsoft Intune aus.

    a. Geben Sie für Anwendungs-ID (Client) die Anwendungs-ID (Client) aus Ihrer Microsoft Entra ID-App-Registrierung ein. Informationen zur Verwendung von Microsoft Intune mit Connector für SCEP finden Sie unter. Konfigurieren Sie Ihr MDM-System für Connector for SCEP

    b. Geben Sie für Verzeichnis-ID (Mandanten-ID) oder primäre Domain entweder die Verzeichnis-ID (Mandanten-ID) oder die primäre Domain aus Ihrer Microsoft Entra ID-App-Registrierung ein.

4. Wählen Sie unter Private CA die private CA aus, die mit diesem Connector verwendet werden soll. Oder erstellen Sie eine neue, indem Sie Private CA erstellen auswählen. Aufgrund der inhärenten Sicherheitslücken im SCEP-Protokoll empfehlen wir, eine private CA zu verwenden, die diesem Connector zugewiesen ist. Wenn Sie eine neue Zertifizierungsstelle erstellt haben, kehren Sie nach Abschluss der Erstellung in AWS Private CA zur Connector for SCEP-Konsole zurück und aktualisieren Sie die Liste der privaten Zertifizierungsstellen. CAs Ihre neue private Zertifizierungsstelle sollte zur Auswahl stehen.
5. Wählen Sie Connector erstellen aus.

Schritt 2: Kopieren Sie die Konnektordetails in Ihr MDM-System

Nachdem Sie Ihren Connector erstellt haben, müssen Sie die folgenden Details aus dem Connector in Ihr MDM-System kopieren. Um die Details eines Connectors mithilfe der Konsole anzuzeigen, wählen Sie den Connector aus der Liste auf der Konsolenseite Connectors for SCEP aus.

- Öffentliche SCEP-URL — Dies ist der Endpunkt des Connectors, von dem Ihre SCEP-Clients Zertifikate anfordern. Achten Sie darauf, diesen Endpunkt nur vertrauenswürdigen Entitäten zur Verfügung zu stellen.
- (Allgemein) Challenge-Passwort — Wählen Sie unter Challenge-Passwörter das Passwort aus, das Sie im vorherigen Verfahren automatisch generiert haben, und wählen Sie dann Passwort anzeigen aus, um das Passwort einzusehen. Um ein zusätzliches Passwort zu erstellen, wählen Sie Passwort erstellen aus. Achten Sie darauf, Passwörter sorgfältig und nur an vertrauenswürdige Personen und Kunden zu verteilen. Ein einziges Challenge-Passwort kann verwendet werden, um jedes Zertifikat mit beliebigem Betreff auszustellen und sollte daher mit Vorsicht behandelt werden. SANs
- (Microsoft Intune) Open ID-Werte — Wenn Sie mit Microsoft Intune integrieren, müssen Sie den Open ID-Aussteller, den Open ID-Betreff und die Open ID-Zielgruppe in die OpenID Connect (OIDC) -Anmeldeinformationen Ihrer Microsoft Entra-App-Registrierung kopieren. Weitere Informationen finden Sie unter Konfigurieren Sie Ihr MDM-System für Connector for SCEP.

Konfigurieren Sie Ihr MDM-System für Connector for SCEP

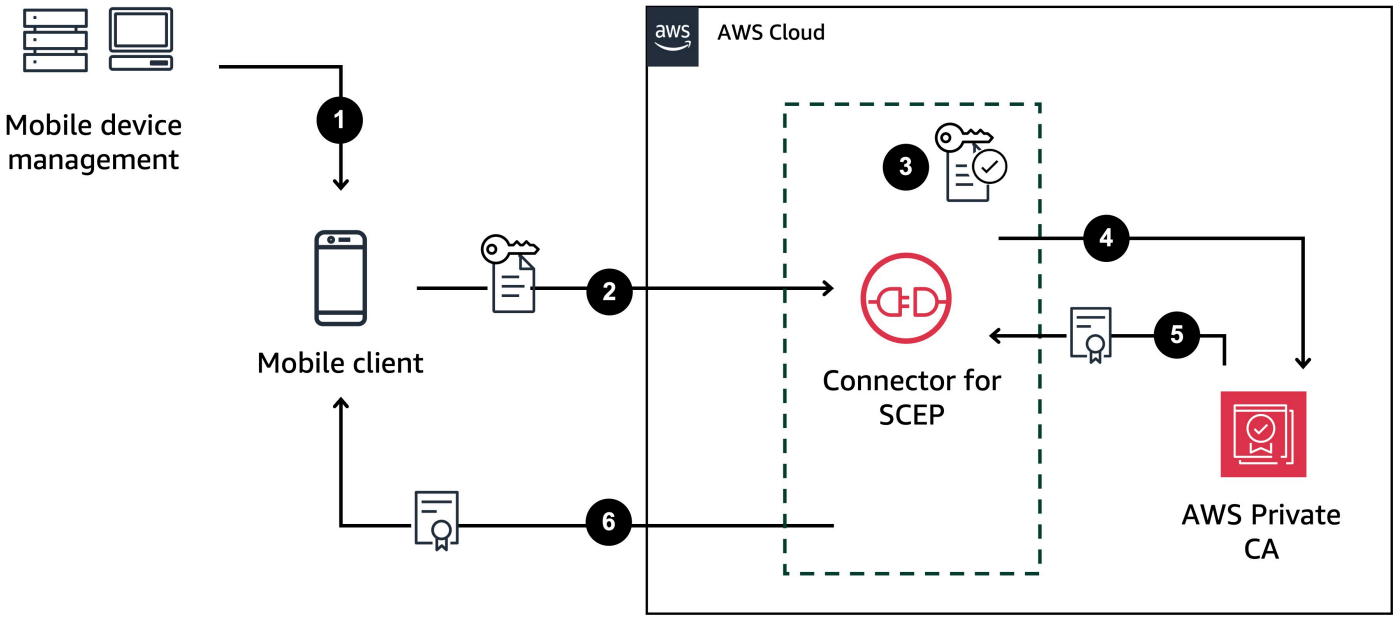
Das Simple Certificate Enrollment Protocol (SCEP) ist ein Standardprotokoll, das für die Registrierung und Verlängerung von Zertifikaten verwendet wird. Connector for SCEP ist ein auf RFC 8894 basierender SCEP-Server, der automatisch Zertifikate für Ihre SCEP-Clients ausstellt. AWS Private Certificate Authority Wenn Sie einen Connector erstellen, stellt Connector for SCEP einen HTTPS-

Endpunkt bereit, von dem SCEP-Clients Zertifikate anfordern können. Die Clients authentifizieren sich mit einem Challenge-Passwort, das als Teil ihrer Zertifikatssignieranforderung (CSR) an den Dienst enthalten ist. Sie können Connector for SCEP mit gängigen MDM-Systemen (Mobile Device Management) wie Microsoft Intune, Omnissa Workspace ONE und Jamf Pro verwenden, um mobile Geräte zu registrieren. Es ist so konzipiert, dass es mit jedem Client oder Endpunkt funktioniert, der SCEP unterstützt.

Connector for SCEP bietet zwei Arten von Anschlüssen: Allzweck-Steckverbinder und Connector für SCEP für Microsoft Intune. In den folgenden Abschnitten wird beschrieben, wie sie funktionieren und wie Sie Ihr MDM-System so konfigurieren, dass es sie verwendet.

Allzweck-Anschluss

Ein Allzweck-Connector ist für die Verwendung mit Endpunkten für mobile Geräte konzipiert, die SCEP unterstützen, mit Ausnahme von Microsoft Intune, das über einen eigenen Connector verfügt. Mit Allzweck-Konnektoren wie Jamf Pro oder Omnissa Workspace ONE verwalten Sie die SCEP-Challenge-Passwörter. In der folgenden Abbildung wird ein MDM-System (Mobile Device Management) als Beispiel verwendet. Dieselbe Funktionalität gilt jedoch auch für andere SCEP-fähige Systeme oder Geräte.



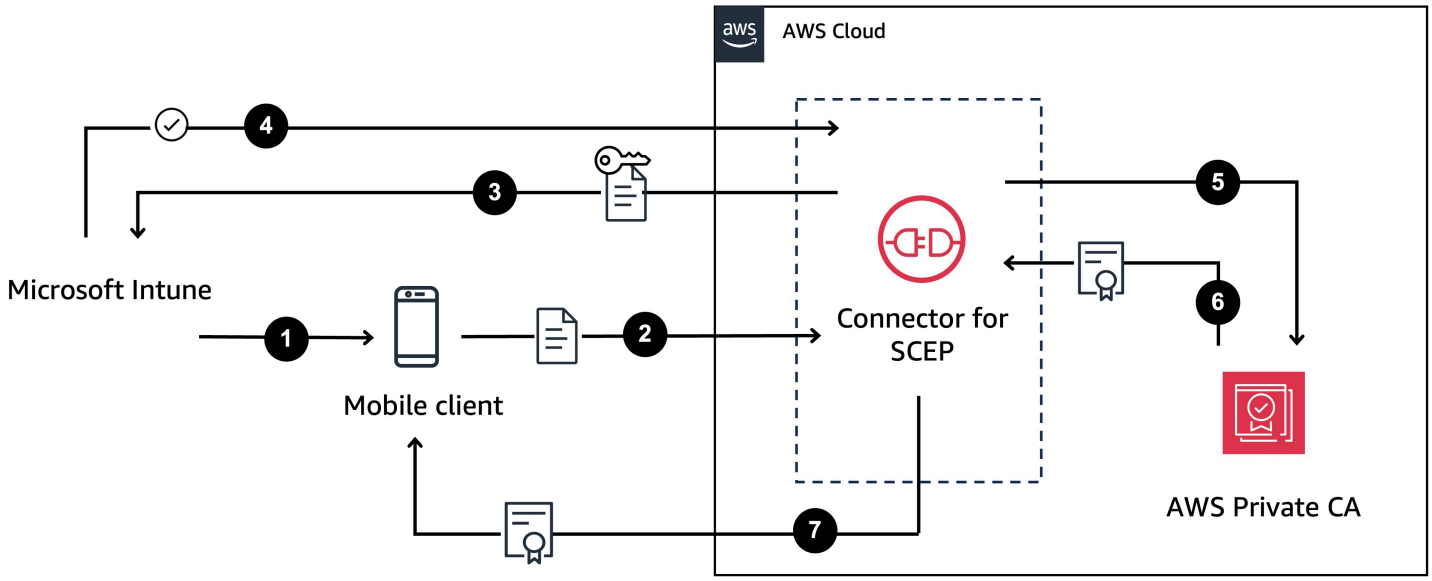
1. Das MDM-System (oder ein anderes Gerät oder System) sendet ein SCEP-Profil an den mobilen Client. Ein SCEP-Profil enthält Konfigurationsparameter, die das Zertifikatsprofil definieren, wie z. B. die Gültigkeitsdauer des Zertifikats, das Challenge-Passwort und andere Informationen, die für die Ausstellung von Zertifikaten relevant sind.

2. Der mobile Client fordert ein Zertifikat an und sendet außerdem eine Zertifikatsignieranforderung (CSR), die ein Challenge-Passwort enthält.
3. Der Connector für SCEP validiert das Challenge-Passwort. Wenn es gültig ist, fordert der Dienst im Namen des mobilen Clients ein Zertifikat AWS Private CA an.
4. AWS Private CA stellt das Zertifikat aus und sendet es an Connector for SCEP.
5. Der Connector für SCEP sendet das ausgestellte Zertifikat an den mobilen Client.

AWS Private CA Konnektor für SCEP für Microsoft Intune

AWS Private CA Der Connector für SCEP für Microsoft Intune wurde für die Verwendung mit Microsoft Intune entwickelt. Mit dem Connectortyp Connector für SCEP für Microsoft Intune verwenden Sie Microsoft Intune, um Ihre SCEP-Challenge-Passwörter zu verwalten. Weitere Informationen zur Verwendung von Connector for SCEP mit Microsoft Intune finden Sie unter. Microsoft Intune für Connector für SCEP konfigurieren

Um Connector for SCEP mit Microsoft Intune verwenden zu können, müssen Sie bestimmte Funktionen mithilfe der Microsoft Intune-API aktivieren und über eine gültige Microsoft Intune-Lizenz verfügen. Sie sollten auch die Microsoft Intune® App Protection Policies lesen.



1. Microsoft Intune sendet ein SCEP-Profil an den mobilen Client. Das Profil enthält ein verschlüsseltes Challenge-Passwort, das der mobile Client in die CSR eingibt.
2. Der mobile Client fordert ein Zertifikat an und sendet die CSR an Connector for SCEP.
3. Connector für SCEP sendet die CSR zur Autorisierung an Microsoft Intune.

4. Microsoft Intune entschlüsselt das Challenge-Passwort in der CSR. Wenn es gültig ist, sendet Microsoft Intune die Genehmigung an Connector for SCEP, das Zertifikat für den mobilen Client auszustellen.
5. Der Connector für SCEP fordert im Namen des mobilen Clients ein Zertifikat AWS Private CA an.
6. AWS Private CA stellt das Zertifikat aus und sendet es an Connector for SCEP.
7. Der Connector für SCEP sendet das ausgestellte Zertifikat an den mobilen Client.

Themen

Konfigurieren Sie Jamf Pro für Connector für SCEP

Sie können es AWS Private CA als externe Zertifizierungsstelle (CA) mit dem Jamf Pro Mobile Device Management (MDM) -System verwenden. Dieses Handbuch enthält Anweisungen zur Konfiguration von Jamf Pro, nachdem Sie einen Allzweck-Connector erstellt haben.

Konfigurieren Sie Jamf Pro für Connector für SCEP

Dieses Handbuch enthält Anweisungen zur Konfiguration von Jamf Pro für die Verwendung mit Connector for SCEP. Nachdem Sie Jamf Pro und Connector für SCEP erfolgreich konfiguriert haben, können Sie AWS Private CA Zertifikate für Ihre verwalteten Geräte ausstellen.

Anforderungen für Jamf Pro

Ihre Implementierung von Jamf Pro muss die folgenden Anforderungen erfüllen.

- Sie müssen die Einstellung Zertifikatsbasierte Authentifizierung aktivieren in Jamf Pro aktivieren aktivieren. Einzelheiten zu dieser Einstellung finden Sie auf der Seite mit den Sicherheitseinstellungen von Jamf Pro in der Jamf Pro-Dokumentation.

Schritt 1: (Optional — empfohlen) Besorgen Sie sich den Fingerabdruck Ihrer privaten CA

Ein Fingerabdruck ist eine eindeutige Kennung für Ihre private Zertifizierungsstelle, mit der Sie die Identität Ihrer Zertifizierungsstelle überprüfen können, wenn Sie Vertrauen zu anderen Systemen oder Anwendungen aufbauen. Durch die Einbindung eines Fingerabdrucks einer Zertifizierungsstelle

(CA) können verwaltete Geräte die Zertifizierungsstelle, mit der sie sich verbinden, authentifizieren und Zertifikate ausschließlich von der erwarteten Zertifizierungsstelle anfordern. Wir empfehlen die Verwendung eines CA-Fingerabdrucks mit Jamf Pro.

Um einen Fingerabdruck für Ihre private CA zu generieren

1. Rufen Sie das Zertifikat der privaten Zertifizierungsstelle entweder über die AWS Private CA Konsole ab oder verwenden Sie die GetCertificateAuthorityCertificate. Speichern Sie es als `ca.pem` Datei.
2. Installieren Sie die OpenSSL Command Line Utilities.
3. Führen Sie in OpenSSL den folgenden Befehl aus, um den Fingerabdruck zu generieren:

```
openssl x509 -in ca.pem -sha256 -fingerprint
```

Schritt 2: In AWS Private CA Jamf Pro als externe Zertifizierungsstelle konfigurieren

Nachdem Sie einen Connector für SCEP erstellt haben, müssen Sie ihn in Jamf Pro AWS Private CA als externe Zertifizierungsstelle (CA) einrichten. Sie können ihn AWS Private CA als globale, externe Zertifizierungsstelle festlegen. Alternativ können Sie ein Jamf Pro-Konfigurationsprofil verwenden, um verschiedene Zertifikate AWS Private CA für verschiedene Anwendungsfälle auszustellen, z. B. die Ausstellung von Zertifikaten für eine Untergruppe von Geräten in Ihrer Organisation. Anleitungen zur Implementierung von Jamf Pro Konfigurationsprofilen würden den Rahmen dieses Dokuments sprengen.

Zur Konfiguration AWS Private CA als externe Zertifizierungsstelle (CA) in Jamf Pro

1. Rufen Sie in der Jamf Pro Konsole die Seite mit den Einstellungen für PKI-Zertifikate auf, indem Sie zu Einstellungen > Global > PKI-Zertifikate gehen.
2. Wählen Sie die Registerkarte Vorlage für Verwaltungszertifikate aus.
3. Wählen Sie Externe Zertifizierungsstelle aus.
4. Wählen Sie Bearbeiten aus.
5. (Optional) Wählen Sie Jamf Pro als SCEP-Proxy für Konfigurationsprofile aktivieren aus. Sie können Jamf Pro-Konfigurationsprofile verwenden, um verschiedene Zertifikate auszustellen, die auf bestimmte Anwendungsfälle zugeschnitten sind. Anleitungen zur Verwendung von Konfigurationsprofilen in Jamf Pro finden Sie in der Jamf Pro-Dokumentation unter Aktivieren von Jamf Pro als SCEP-Proxy für Konfigurationsprofile.

6. Wählen Sie Für die Registrierung von Computern und Mobilgeräten eine SCEP-fähige externe Zertifizierungsstelle verwenden aus.
7. (Optional) Wählen Sie Jamf Pro als SCEP-Proxy für die Registrierung von Computern und Mobilgeräten verwenden aus. Falls bei der Profilinstallation Fehler auftreten, finden Sie weitere Informationen unter. Beheben Sie Fehler bei der Profilinstallation
8. Kopieren Sie die öffentliche SCEP-URL des Connectors für SCEP aus den Konnektordetails und fügen Sie sie in das URL-Feld in Jamf Pro ein. Um die Details eines Connectors anzuzeigen, wählen Sie den Connector aus der Liste Connectors für SCEP aus. Alternativ können Sie die URL abrufen, indem Sie den `Endpoint` Wert aus der Antwort aufrufen GetConnectorund kopieren.
9. (Optional) Geben Sie den Namen der Instanz in das Feld Name ein. Sie können sie beispielsweise benennen AWS Private CA.
10. Wählen Sie Statisch als Challenge-Typ aus.
11. Kopieren Sie ein Challenge-Passwort von Ihrem Connector und fügen Sie es in das Challenge-Feld ein. Ein Connector kann mehrere Challenge-Passwörter haben. Um die Challenge-Passwörter Ihres Connectors einzusehen, navigieren Sie in der AWS Konsole zur Detailseite Ihres Connectors und wählen Sie die Schaltfläche Passwort anzeigen. Alternativ können Sie die Challenge-Passwörter eines Connectors abrufen, indem Sie anrufen GetChallengePasswordund einen `Password` Wert aus der Antwort kopieren. Hinweise zur Verwendung von Challenge-Passwörtern finden Sie unterMachen Sie sich mit den Überlegungen und Einschränkungen von Connector for SCEP vertraut.
12. Fügen Sie das Challenge-Passwort in das Feld Verify Challenge ein.
13. Wählen Sie eine Schlüsselgröße. Wir empfehlen eine Schlüsselgröße von 2048 oder höher.
14. (Optional) Wählen Sie Als digitale Signatur verwenden aus. Wählen Sie diese Option für Authentifizierungszwecke, um Geräten sicheren Zugriff auf Ressourcen wie WLAN und VPN zu gewähren.
15. (Optional) Wählen Sie Für Schlüsselverschlüsselung verwenden aus.
16. (Optional — empfohlen) Geben Sie eine Hexadezimalzahl in das Feld Fingerabdruck ein. Wir empfehlen, einen CA-Fingerabdruck hinzuzufügen, damit verwaltete Geräte die CA verifizieren können, und nur Zertifikate von der CA anzufordern. Anweisungen zum Generieren eines Fingerabdrucks für Ihre private Zertifizierungsstelle finden Sie unterSchritt 1: (Optional — empfohlen) Besorgen Sie sich den Fingerabdruck Ihrer privaten CA.
17. Wählen Sie Speichern aus.

Schritt 3: Richten Sie ein Signaturzertifikat für ein Konfigurationsprofil ein

Um Jamf Pro mit Connector for SCEP zu verwenden, müssen Sie die Signatur- und CA-Zertifikate für die private CA bereitstellen, die Ihrem Connector zugeordnet ist. Sie können dies tun, indem Sie einen Keystore für das Profilsignaturzertifikat auf Jamf Pro hochladen, der beide Zertifikate enthält.

Gehen Sie wie folgt vor, um einen Zertifikat-Keystore zu erstellen und ihn in Jamf Pro hochzuladen:

- Generieren Sie mithilfe Ihrer internen Prozesse eine Certificate Signing Request (CSR).
- Lassen Sie die CSR von der privaten Zertifizierungsstelle signieren, die Ihrem Connector zugeordnet ist.
- Erstellen Sie einen Keystore für das Profilsignaturzertifikat, der sowohl die Profilsignatur- als auch die CA-Zertifikate enthält.
- Laden Sie den Zertifikat-Keystore auf Jamf Pro hoch.

Indem Sie diese Schritte befolgen, können Sie sicherstellen, dass Ihre Geräte das von Ihrer privaten Zertifizierungsstelle signierte Konfigurationsprofil validieren und authentifizieren können, sodass Sie Connector for SCEP mit Jamf Pro verwenden können.

1. Das folgende Beispiel verwendet OpenSSL und AWS Certificate Manager, aber Sie können mit Ihrer bevorzugten Methode eine Zertifikatsignieranforderung generieren.

   AWS Certificate Manager console

   Um ein Profilsignaturzertifikat mit der ACM-Konsole zu erstellen

   1. Verwenden Sie ACM, um ein privates PKI-Zertifikat anzufordern. Fügen Sie Folgendes hinzu:

      - Typ — Verwenden Sie denselben privaten CA-Typ, der als SCEP-Zertifizierungsstelle für Ihr MDM-System dient.
      - Wählen Sie im Abschnitt Details zur Zertifizierungsstelle das Menü Zertifizierungsstelle aus und wählen Sie die private Zertifizierungsstelle aus, die als Zertifizierungsstelle für Jamf Pro dient.
      - Domainname — Geben Sie einen Domainnamen an, der in das Zertifikat eingebettet werden soll. Sie können einen vollqualifizierten Domänennamen (FQDN), z. B.`www.example.com`, oder einen bloßen Domainnamen oder einen Apex-Domänennamen wie `example.com` (der ausschließt`www.`) verwenden.

2. Verwenden Sie ACM, um das private Zertifikat zu exportieren, das Sie im vorherigen Schritt erstellt haben. Wählen Sie Datei für das Zertifikat, die Zertifikatskette und den verschlüsselten Schlüssel exportieren aus. Halten Sie die Passphrase griffbereit, da Sie sie im nächsten Schritt benötigen.
3. Führen Sie in einem Terminal den folgenden Befehl in einem Ordner aus, der die exportierten Dateien enthält, um das PKCS #12 -Bundle in die `output.p12` Datei zu schreiben, die mit der Passphrase codiert wurde, die Sie im vorherigen Schritt erstellt haben.

```
openssl pkcs12 -export \
  -in "Exported Certificate.txt" \
  -certfile "Certificate Chain.txt" \
  -inkey "Exported Certificate Private Key.txt" \
  -name example \
  -out output.p12 \
  -passin pass:your-passphrase \
  -passout pass:your-passphrase
```

AWS Certificate Manager CLI

So erstellen Sie ein Profilsignaturzertifikat mit der ACM-CLI

- Der folgende Befehl zeigt, wie Sie ein Zertifikat in ACM erstellen und die Dateien anschließend als PKCS #12 -Paket exportieren.

```
PCA=<Enter your Private CA ARN>

CERTIFICATE=$(aws acm request-certificate \
    --certificate-authority-arn $PCA \
    --domain-name <any valid domain name, such as test.name> \
    | jq -r '.CertificateArn')

while [[ $(aws acm describe-certificate \
  --certificate-arn $CERTIFICATE \
  | jq -r '.Certificate.Status') != "ISSUED" ]] do sleep 1; done

aws acm export-certificate \
  --certificate-arn $CERTIFICATE \
  --passphrase password | jq -r '.Certificate' > Certificate.pem
aws acm export-certificate \
```

```
  --certificate-arn $CERTIFICATE \
  --passphrase password | jq -r '.CertificateChain' > CertificateChain.pem
aws acm export-certificate \
  --certificate-arn $CERTIFICATE \
  --passphrase password | jq -r '.PrivateKey' > PrivateKey.pem

openssl pkcs12 -export \
  -in "Certificate.pem" \
  -certfile "CertificateChain.pem" \
  -inkey "PrivateKey.pem" \
  -name example \
  -out output.p12 \
  -passin pass:passphrase \
  -passout pass:passphrase
```

OpenSSL CLI

So erstellen Sie ein Profilsignaturzertifikat mit OpenSSL CLI

1. Generieren Sie mit OpenSSL einen privaten Schlüssel, indem Sie den folgenden Befehl ausführen.

```
openssl genrsa -out local.key 2048
```

2. Generieren Sie eine Zertifikatsignieranforderung (CSR):

```
openssl req -new -key local.key -sha512 -out local.csr -
subj "/CN=MySigningCertificate/O=MyOrganization" -addext
 keyUsage=critical,digitalSignature,nonRepudiation
```

3. Stellen Sie mithilfe von das AWS CLI das Signaturzertifikat mit der CSR aus, die Sie im vorherigen Schritt generiert haben. Führen Sie den folgenden Befehl aus und notieren Sie sich den Zertifikat-ARN in der Antwort.

```
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. Rufen Sie das Signaturzertifikat ab, indem Sie den folgenden Befehl ausführen. Geben Sie den Zertifikat-ARN aus dem vorherigen Schritt an.

```
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. Rufen Sie das CA-Zertifikat ab, indem Sie den folgenden Befehl ausführen.

```
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. Geben Sie mit OpenSSL den Keystore des Signaturzertifikats im p12-Format aus. Verwenden Sie die CRT-Dateien, die Sie in den Schritten vier und fünf generiert haben.

```
openssl pkcs12 -export -in local.crt -inkey local.key -certfile ca.crt -name
 "CA Chain" -out local.p12
```

7. Wenn Sie dazu aufgefordert werden, geben Sie ein Exportkennwort ein. Dieses Passwort ist Ihr Keystore-Passwort, das Sie Jamf Pro zur Verfügung stellen müssen.

2. Navigieren Sie in Jamf Pro zur Vorlage für Management-Zertifikate und anschließend zum Bereich Externe Zertifizierungsstelle.
3. Wählen Sie unten im Bereich Externe CA die Option Change Signing and CA Certificates aus.
4. Folgen Sie den Anweisungen auf dem Bildschirm, um die Signatur- und CA-Zertifikate für die externe CA hochzuladen.

Schritt 4: (Optional) Installieren Sie das Zertifikat während der vom Benutzer initiierten Registrierung

Um eine Vertrauensstellung zwischen Ihren Client-Geräten und Ihrer privaten Zertifizierungsstelle herzustellen, müssen Sie sicherstellen, dass Ihre Geräte den von Jamf Pro ausgestellten Zertifikaten vertrauen. Sie können die Einstellungen für die benutzerinitiierte Registrierung von Jamf Pro verwenden, um Ihr AWS Private CA CA-Zertifikat automatisch auf den Client-Geräten zu installieren, wenn diese während des Registrierungsprozesses ein Zertifikat anfordern.

Beheben Sie Fehler bei der Profilinstallation

Falls bei der Profilinstallation nach der Aktivierung von Jamf Pro als SCEP-Proxy verwenden für die Registrierung von Computern und Mobilgeräten Fehler auftreten, schlagen Sie in Ihren Geräteprotokollen nach und versuchen Sie Folgendes.

Fehlermeldung im Geräteprotokoll	Abhilfe
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:15001>`	Wenn Sie beim Versuch, sich zu registrieren, diese Fehlermeldung erhalten, versuchen Sie die Registrierung erneut. Es kann mehrere Versuche dauern, bis die Registrierung erfolgreich ist.
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:14006>`	Ihr Challenge-Passwort ist möglicherweise falsch konfiguriert. Stellen Sie sicher, dass das Challenge-Passwort in Jamf Pro mit dem Challenge-Passwort Ihres Connectors übereinstimmt.

Microsoft Intune für Connector für SCEP konfigurieren

Sie können es AWS Private CA als externe Zertifizierungsstelle (CA) mit dem Microsoft Intune Mobile Device Management (MDM) -System verwenden. Dieses Handbuch enthält Anweisungen zur Konfiguration von Microsoft Intune, nachdem Sie einen Connector für SCEP für Microsoft Intune erstellt haben.

Voraussetzungen

Bevor Sie einen Connector für SCEP für Microsoft Intune erstellen, müssen Sie die folgenden Voraussetzungen erfüllen.

- Erstellen Sie eine Entra-ID.
- Erstellen Sie einen Microsoft Intune-Mandanten.
- Erstellen Sie eine App-Registrierung in Ihrer Microsoft Entra ID. Informationen zur Verwaltung von Berechtigungen auf Anwendungsebene für Ihre App-Registrierung finden Sie unter Aktualisieren der angeforderten Berechtigungen einer App in Microsoft Entra ID in der Microsoft Entra-Dokumentation. Die App-Registrierung muss über die folgenden Berechtigungen verfügen:
  - Stellen Sie unter Intune scep_challenge_provider ein.
  - Legen Sie für Microsoft Graph Application.Read.All und User.Read fest.

- Sie müssen der Anwendung in Ihrer App-Registrierung die Zustimmung des Administrators erteilen. Weitere Informationen finden Sie in der Microsoft Entra-Dokumentation unter Erteilen einer mandantenweiten Administratorzustimmung für eine Anwendung.

> **Tip**
>
> Notieren Sie sich bei der Erstellung der App-Registrierung die Anwendungs-ID (Client) und die Verzeichnis-ID (Mandanten-ID) oder die primäre Domain. Wenn Sie Ihren Connector für SCEP für Microsoft Intune erstellen, geben Sie diese Werte ein. Informationen zum Abrufen dieser Werte finden Sie in der Microsoft Entra-Dokumentation unter Erstellen einer Microsoft Entra-Anwendung und eines Dienstprinzipals, der auf Ressourcen zugreifen kann.

Schritt 1: Erteilen Sie die AWS Private CA Erlaubnis zur Nutzung Ihrer Microsoft Entra ID-Anwendung

Nachdem Sie einen Connector für SCEP für Microsoft Intune erstellt haben, müssen Sie unter der Microsoft-App-Registrierung Verbundanmeldeinformationen erstellen, damit Connector für SCEP mit Microsoft Intune kommunizieren kann.

So konfigurieren Sie AWS Private CA als externe Zertifizierungsstelle in Microsoft Intune

1. Navigieren Sie in der Microsoft Entra ID-Konsole zu den App-Registrierungen.
2. Wählen Sie die Anwendung aus, die Sie für die Verwendung mit Connector for SCEP erstellt haben. Die Anwendungs- (Client-) ID der Anwendung, auf die Sie klicken, muss mit der ID übereinstimmen, die Sie bei der Erstellung des Connectors angegeben haben.
3. Wählen Sie im Dropdownmenü Verwaltet die Option Zertifikate und Geheimnisse aus.
4. Wählen Sie die Registerkarte Federated Credentials aus.
5. Wählen Sie Anmeldeinformationen hinzufügen aus.
6. Wählen Sie im Dropdownmenü Szenario mit Verbundanmeldedaten die Option Anderer Aussteller aus.
7. Kopieren Sie den OpenID-Ausstellerwert aus Ihren Connector für SCEP for Microsoft Intune-Details und fügen Sie ihn in das Feld Issuer ein. Um die Details eines Connectors anzuzeigen, wählen Sie den Connector aus der Liste Connectors für SCEP in der Konsole aus. AWS

Alternativ können Sie die URL abrufen, indem Sie anrufen GetConnectorund dann den `Issuer` Wert aus der Antwort kopieren.

8. Wählen Sie als Typ die Option Explizite Subject Identifier aus.
9. Kopieren Sie den OpenID-Betreffwert aus Ihrem Connector und fügen Sie ihn in das Feld Wert ein. Sie können den OpenID-Ausstellerwert auf der Seite mit den Connector-Details in der AWS Konsole anzeigen. Alternativ können Sie die URL abrufen, indem Sie den `Audience` Wert aus der Antwort aufrufen GetConnectorund dann kopieren.
10. (Optional) Geben Sie den Namen der Instanz in das Feld Name ein. Sie können sie beispielsweise benennen AWS Private CA.
11. (Optional) Geben Sie eine Beschreibung in das Feld Beschreibung ein.
12. Kopieren Sie den OpenID Audience-Wert aus Ihren Connector für SCEP for Microsoft Intune-Details und fügen Sie ihn in das Feld Audience ein. Um die Details eines Connectors anzuzeigen, wählen Sie den Connector aus der Liste Connectors für SCEP in der Konsole aus. AWS Alternativ können Sie die URL abrufen, indem Sie anrufen GetConnectorund dann den `Subject` Wert aus der Antwort kopieren.
13. Wählen Sie Hinzufügen aus.

Schritt 2: Richten Sie ein Microsoft Intune-Konfigurationsprofil ein

Nachdem Sie AWS Private CA die Erlaubnis zum Aufrufen von Microsoft Intune erteilt haben, müssen Sie Microsoft Intune verwenden, um ein Microsoft Intune-Konfigurationsprofil zu erstellen, das Geräte anweist, sich für die Zertifikatsausstellung an Connector for SCEP zu wenden.

1. Erstellen Sie ein vertrauenswürdiges Zertifikatskonfigurationsprofil. Sie müssen das Root-CA-Zertifikat der Kette, die Sie mit Connector for SCEP verwenden, in Microsoft Intune hochladen, um Vertrauen herzustellen. Informationen zum Erstellen eines Konfigurationsprofils für vertrauenswürdige Zertifikate finden Sie unter Vertrauenswürdige Stammzertifikatsprofile für Microsoft Intune in der Microsoft Intune-Dokumentation.
2. Erstellen Sie ein SCEP-Zertifikatkonfigurationsprofil, das Ihre Geräte auf den Connector verweist, wenn sie ein neues Zertifikat benötigen. Der Profiltyp des Konfigurationsprofils sollte SCEP-Zertifikat sein. Stellen Sie für das Stammzertifikat des Konfigurationsprofils sicher, dass Sie das vertrauenswürdige Zertifikat verwenden, das Sie im vorherigen Schritt erstellt haben.

   Kopieren Sie für SCEP-Server URLs die öffentliche SCEP-URL aus den Details Ihres Connectors und fügen Sie sie in das Feld SCEP-Server ein. URLs Um die Details eines Connectors anzuzeigen, wählen Sie den Connector aus der Liste Connectors für SCEP aus.

Alternativ können Sie die URL abrufen ListConnectors, indem Sie anrufen und dann den `Endpoint` Wert aus der Antwort kopieren. Anleitungen zum Erstellen von Konfigurationsprofilen in Microsoft Intune finden Sie unter Erstellen und Zuweisen von SCEP-Zertifikatsprofilen in Microsoft Intune in der Microsoft Intune-Dokumentation.

> **Note**
>
> Wenn Sie für Geräte ohne Mac OS und iOS keinen Gültigkeitszeitraum im Konfigurationsprofil festlegen, stellt Connector for SCEP ein Zertifikat mit einer Gültigkeit von einem Jahr aus. Wenn Sie im Konfigurationsprofil keinen Wert für Extended Key Usage (EKU) festlegen, stellt Connector for SCEP ein Zertifikat aus, bei dem der EKU-Wert auf gesetzt ist. `Client Authentication (Object Identifier: 1.3.6.1.5.5.7.3.2)` Bei macOS- oder iOS-Geräten berücksichtigt Microsoft Intune unsere `Validity` Parameter in Ihren Konfigurationsprofilen nicht. `ExtendedKeyUsage` Für diese Geräte stellt Connector for SCEP über die Client-Authentifizierung ein Zertifikat mit einer Gültigkeitsdauer von einem Jahr für diese Geräte aus.

Schritt 3: Überprüfen Sie die Verbindung zum Connector für SCEP

Nachdem Sie ein Microsoft Intune-Konfigurationsprofil erstellt haben, das auf den Connector for SCEP-Endpunkt verweist, stellen Sie sicher, dass ein registriertes Gerät ein Zertifikat anfordern kann. Stellen Sie zur Bestätigung sicher, dass keine Fehler bei der Richtlinienzuweisung vorliegen. Gehen Sie zur Bestätigung im Intune-Portal zu Geräte > Geräte verwalten > Konfiguration und stellen Sie sicher, dass unter Fehler bei der Zuweisung von Konfigurationsrichtlinien nichts aufgeführt ist. Falls ja, bestätigen Sie Ihre Einrichtung mit den Informationen aus den vorherigen Verfahren. Wenn Ihre Einrichtung korrekt ist und immer noch Fehler auftreten, finden Sie weitere Informationen unter Verfügbare Daten vom Mobilgerät sammeln.

Informationen zur Geräteregistrierung finden Sie unter Was ist Geräteregistrierung? in der Microsoft Intune-Dokumentation.

Konfigurieren Sie Omnissa Workspace ONE für Connector für SCEP

Sie können es AWS Private CA als externe Zertifizierungsstelle (CA) mit dem Omnissa Workspace ONE UEM-System (Unified Endpoint Management) verwenden. Dieses Handbuch enthält Anweisungen zur Konfiguration von Omnissa Workspace ONE, nachdem Sie einen SCEP-Connector in erstellt haben. AWS

Voraussetzungen

Bevor Sie einen SCEP-Connector für Omnissa Workspace ONE erstellen, müssen Sie die folgenden Voraussetzungen erfüllen:

- Erstellen Sie eine private Zertifizierungsstelle in der Konsole. AWS Weitere Informationen finden Sie unter Erstellen Sie eine private CA in AWS Private CA.
- Erstellen Sie einen SCEP-Connector für allgemeine Zwecke. Weitere Informationen finden Sie unter Einen Konnektor erstellen.
- Verfügen Sie über ein aktives Administratorkonto für die Omnissa Workspace ONE-Umgebung mit einer Organisationsgruppen-ID.
- Wenn Sie ein Apple-Gerät registrieren, konfigurieren Sie den Apple Push Notification Service (APNs) für MDM. Weitere Informationen finden Sie in der Omnissa-Dokumentation unter APNs Zertifikate.

Schritt 1: Definieren Sie eine Zertifizierungsstelle und eine Vorlage in Omnissa Workspace ONE

Nachdem Sie eine private CA und einen SCEP-Connector in der AWS Konsole erstellt haben, definieren Sie die Zertifizierungsstelle und die Vorlage in Omnissa Workspace ONE.

AWS Private CA Als Zertifizierungsstelle hinzufügen

1. Wählen Sie im Menü System die Option Enterprise Integration und dann Certificate Authorities aus.
2. Wählen Sie + HINZUFÜGEN und geben Sie die folgenden Informationen ein:
   - Name: AWS-Private-CA.
   - Beschreibung: AWS Private CA für die Ausstellung von Gerätezertifikaten.
   - Autoritätstyp: Wählen Sie Generic SCEP aus.
   - SCEP-URL: Geben Sie die SCEP-URL von ein. AWS Private CA
   - Challenge-Typ: Wählen Sie STATIC aus.
   - Statische Herausforderung: Geben Sie das statische SCEP-Challenge-Passwort aus der Connector-Konfiguration für die SCEP-Konfiguration in der AWS Konsole ein.
   - Geben Sie die Werte „Timeout für Wiederholungen“ und „Max. Wiederholungen“ ein.
3. Speichern Sie die Konfiguration.

Erstellen Sie eine Zertifikatsvorlage

1. Wählen Sie im Menü System die Option Enterprise Integration, wählen Sie Certificate Authorities und dann Templates aus.
2. Wählen Sie Vorlagen hinzufügen und geben Sie die folgenden Informationen ein:
   - Name der Vorlage: Device-Cert-Template.
   - Zertifizierungsstelle: Wählen Sie AWS-Private-CA.
   - Betreffname: Dies ist ein anpassbares Feld. Sie können Variablenwerte aus einer Liste von Attributen auswählen. Zum Beispiel CN= {DeviceReportedName}, O= {DevicePlatform}, OU= {1} CustomAttribute
   - Länge des privaten Schlüssels: 2048 Bit.
   - Typ des privaten Schlüssels: Wählen Sie nach Bedarf Signieren und Verschlüsseln
   - Automatische Verlängerung: Enabled/Disabled (Je nach Ihren Bedürfnissen).
3. Speichern Sie die Vorlage.

Schritt 2: Richten Sie eine Omnissa Workspace ONE UEM-Profilkonfiguration ein

Erstellen Sie ein Profil in Omnissa Workspace ONE UEM, das Geräte an Connector for SCEP weiterleitet, um ein Zertifikat auszustellen.

Erstellen Sie ein SCEP-Geräteprofil für die Zertifikatsverteilung

1. Wählen Sie im Menü Ressourcen die Option Profile & Baselines und dann Profile aus.
2. Wählen Sie „Hinzufügen“ und dann „Profil hinzufügen“
3. Wählen Sie die Geräteplattform aus (Android, iOS, macOS, Windows).
4. Stellen Sie den Verwaltungstyp und den Kontext entsprechend ein.
5. Stellen Sie den Namen ein: Device-Cert-Profile.
6. Scrollen Sie zu SCEP Payload.
7. Wählen Sie SCEP und dann +Hinzufügen.
8. Verwenden Sie die folgende Konfiguration:
   - SCEP:
     - Wählen Sie für Credential Source die Option Defined Certificate Authority (Standard) aus.
     - Wählen Sie für Certificate Authority -Private-CA aus AWS

- Wählen Sie für Certificate Template das in Schritt 1 definierte Device-Cert-Template aus.

9. Wählen Sie Weiter und wählen Sie im Abschnitt Zuweisung die richtige Smartgroup aus der Liste aus (Zuweisungsgruppe für das Gerät).
10. Wählen Sie für den Zuweisungstyp die Option auto aus, um die automatische Verlängerung zu aktivieren.
11. Speichern und veröffentlichen Sie das Profil.

> **Note**
>
> Weitere Informationen finden Sie unter SCEP in der Omnissa-Dokumentation.

Schritt 3: Geräte bei Omnissa Workspace ONE registrieren

Erstellen oder verifizieren Sie eine Smart-Gruppe

1. Wählen Sie unter Gruppen und Einstellungen die Option Gruppen und dann Zuweisungsgruppen aus.
2. Erstellen oder bearbeiten Sie die Smart-Gruppe für POC-Geräte:
   - Name: POC-Geräte.
   - Gerätetyp: Wählen Sie Alle oder eine bestimmte Plattform (z. B. Android oder iOS).
   - Kriterien: Verwendung UserGroup, Plattform und Betriebssystem, OEM und Modell, um die Kriterien für die Gruppierung der Zielgeräte festzulegen.
   - Besitz: Wählen Sie „Beliebig“ für private Geräte oder Unternehmensgeräte aus.
3. Speichern und überprüfen Sie, ob die Zielgeräte auf der Registerkarte Vorschau angezeigt werden.

Manuelle Geräteregistrierung

Android

- Laden Sie die Workspace ONE Intelligent Hub-App von Google Play herunter.
- Öffnen Sie die App und geben Sie die Registrierungs-URL ein oder scannen Sie einen QR-Code.

- Melden Sie sich an und folgen Sie den Anweisungen, um sich als MDM-verwaltetes Gerät zu registrieren.

iOS/macOS

- Öffnen Sie auf dem Gerät Safari und navigieren Sie zur Registrierungs-URL (z. B. https://<Workspace ONEUEMHostname >/enroll).
- Melden Sie sich mit Benutzeranmeldedaten an.
- Laden Sie die Workspace ONE Intelligent Hub-App aus dem App Store herunter und installieren Sie sie.
- Folgen Sie den Anweisungen zur Installation des MDM-Profils unter Einstellungen > Allgemein > VPN- und Geräteverwaltung > Profil > Installieren.

Windows

- Laden Sie den Workspace ONE Intelligent Hub vom Workspace ONE-Server oder Microsoft Store herunter.
- Melden Sie sich über den Hub mit der Registrierungs-URL und den Anmeldeinformationen an.

Weisen Sie registrierte Geräte der POC-Devices Smart Group unter Geräte > Listenansicht > Weitere Aktionen > Zu Smart Group zuweisen zu.

Weitere Informationen finden Sie unter Automatisierte Geräteregistrierung in der Omnissa-Dokumentation.

Überprüfen Sie die Registrierung

1. Gehen Sie in der Omnissa Workspace ONE UEM Console zu Geräte und dann zu Listenansicht.
2. Vergewissern Sie sich, dass Ihre registrierten Geräte mit dem Status „Registriert“ angezeigt werden.
3. Stellen Sie auf der Registerkarte Gruppen der Gerätedetails sicher, dass sich die Geräte in der Smart-Gruppe POC-Geräte befinden.

Schritt 4: Stellen Sie ein Zertifikat aus

Auslöser für die Ausstellung eines Zertifikats

1. Wählen Sie in der Gerätelistenansicht das registrierte Gerät aus.
2. Klicken Sie auf die Schaltfläche „Abfragen“, um zum Einchecken aufzufordern.

3. Sie Device-Cert-Profile sollten ein Zertifikat ausstellen über. AWS Private CA

Überprüfen Sie die Installation des Zertifikats

Android

Wählen Sie Einstellungen, dann Sicherheit, vertrauenswürdige Anmeldeinformationen und dann Benutzer, um das Zertifikat zu überprüfen.

iOS

Gehen Sie zu Einstellungen und wählen Sie dann Allgemein, dann VPN- und Geräteverwaltung und dann Konfigurationsprofil aus. Stellen Sie sicher, dass das Zertifikat von vorhanden AWS-Private-CA ist.

macOS

Öffnen Sie Keychain Access und dann System Keychain und überprüfen Sie das Zertifikat.

Windows

Öffnen Sie certmgr.msc, dann Personal und dann Certificates, um das Zertifikat zu verifizieren.

Fehlerbehebung

SCEP-Fehler (z. B. „22013 — Der SCEP-Server hat eine ungültige Antwort zurückgegeben“)

- Stellen Sie sicher, dass die SCEP-URL und das statische Challenge-Passwort in Workspace ONE übereinstimmen. AWS Private CA
- <SCEP_URL>Testen Sie die SCEP-Endpunktkonnektivität: curl.
- Überprüfen Sie die AWS CloudTrail Protokolle AWS Private CA auf Fehler (z. B. `IssueCertificate` Ausfälle).

APNs Probleme (iOS/macOS)

- Stellen Sie sicher, dass das APNs Zertifikat gültig und der richtigen Organisationsgruppe zugewiesen ist.
- Testen Sie die APNs Konnektivität: telnet gateway.push.apple.com 2195.

Fehler bei der Profilinstallation

- Vergewissern Sie sich, dass sich die Geräte in der richtigen Smartgroup befinden (Geräte, dann Listenansicht und dann Gruppen).
- Eine Profilsynchronisierung erzwingen: Weitere Aktionen, dann Senden und dann Profilliste.

Protokolle

- Android: Verwenden Sie Logcat - oder Workspace ONE-Protokolle.
- iOS/macOS: log show --predicate 'process == "mdmclient"' --last 1h (via Xcode/ AppleKonfigurator).
- Windows: Event Viewer, dann Anwendungs- und Dienstprotokolle und dann Microsoft-Windows -. DeviceManagement
- Workspace ONE UEM: Monitor, dann Reports & Analytics, dann Events und dann Device Events.

Ausführliche Informationen zum Connector für die SCEP-Überwachung finden Sie unter Monitor-Connector für SCEP. AWS

Sicherheitsüberlegungen

- Speichern Sie SCEP URLs und geheime Daten sicher. Weitere Informationen finden Sie im AWS Secrets Manager Service.
- Beschränken Sie die Smart-Group-Kriterien nur auf Zielgeräte.
- Erneuern Sie regelmäßig die Zertifikate für Apple Push Notifications (APNs) (gültig für 1 Jahr).
- Lege kurze Gültigkeitszeiträume für Zertifikate für Machbarkeitsstudien fest, um das Risiko zu minimieren.
- Stellen Sie bei privaten Geräten sicher, dass bei der Bereinigung alle Profile und Zertifikate entfernt werden.

Informationen zur Konfiguration der Omnissa Workspace ONE UEM- und CA-Integration mithilfe eines SCEP-Connectors finden Sie in der Dokumentation zu SCEP in Omnissa Workspace ONE.

Monitor-Connector für SCEP

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von Connector for SCEP und Ihren anderen Lösungen. AWS AWS bietet die folgenden Überwachungstools, um Connector for SCEP zu überwachen, zu melden, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen zu ergreifen:

- AWS CloudTrail erfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS-Konto -Kontos erfolgten, und übermittelt die Protokolldateien an einen von Ihnen

angegebenen Amazon-S3-Bucket. Sie können feststellen, welche Benutzer und Konten angerufen wurden AWS APIs, von welcher Quell-IP-Adresse aus die Anrufe getätigt wurden und wann die Anrufe erfolgten.

Wenn Sie CloudTrail Datenereignisse überwachen, enthalten die Protokolle eine Liste aller aktuellen Anfragen von Client-Geräten. Bei Datenereignissen handelt es sich um identifizierende Informationen zum Client-Gerät wie IP-Adresse, Art des ausgeführten Vorgangs, Fehlercode und detaillierte Meldungen, falls der Vorgang zu einem `failed` Status führt. Weitere Informationen finden Sie im AWS CloudTrail -Benutzerhandbuch.

- Amazon EventBridge ist ein serverloser Event-Bus-Service, der es einfach macht, Ihre Anwendungen mit Daten aus einer Vielzahl von Quellen zu verbinden. EventBridge liefert einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen, Software-as-a-Service (SaaS-) Anwendungen und AWS Diensten und leitet diese Daten an Ziele wie Lambda und CloudWatch Logs weiter. Auf diese Weise können Sie Ereignisse überwachen, die in Services auftreten, und ereignisgesteuerte Architekturen erstellen. Weitere Informationen finden Sie im EventBridge Amazon-Benutzerhandbuch.

Topics

- Automatisieren Sie Connector für SCEP mit EventBridge
- Log Connector für SCEP-API-Aufrufe mit AWS CloudTrail

Automatisieren Sie Connector für SCEP mit EventBridge

Sie können Amazon verwenden EventBridge, um Ihre AWS Services zu automatisieren und automatisch auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen zu reagieren. Ereignisse aus AWS Services werden nahezu EventBridge in Echtzeit zugestellt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie von Interesse sind und welche automatisierten Aktionen ergriffen werden sollen, wenn ein Ereignis einer Regel entspricht. EventBridge werden mindestens einmal veröffentlicht. Weitere Informationen finden Sie unter Regeln erstellen, die auf Ereignisse reagieren in EventBridge.

CloudWatch Ereignisse werden mithilfe von in Aktionen umgewandelt EventBridge. Mit können Sie Ereignisse verwenden EventBridge, um Ziele auszulösen. Weitere Informationen finden Sie unter Was ist Amazon EventBridge?

Konnektor für SCEP-Ereignistypen

Die Ausstellung des Zertifikats war erfolgreich

Der Connector für SCEP sendet ein `Certificate Issuance Succeeded` Ereignis an, EventBridge wenn wir als Antwort auf eine Anfrage ein Zertifikat ausstellen. `PkiOperationPost`

Im Folgenden finden Sie Beispieldaten für das Ereignis.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Issuance Succeeded",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "success",
       "requestType": "PkiOperationPost",
       "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
   }
}
```

Die Ausstellung des Zertifikats ist fehlgeschlagen

Der Connector für SCEP sendet ein `Certificate Issuance Failed` Ereignis, EventBridge wenn wir als Antwort auf eine Anfrage kein Zertifikat ausstellen können. `PkiOperationPost`

Im Folgenden finden Sie Beispieldaten für das Ereignis.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Issuance Failed",
   "source": "aws.pca-connector-scep",
```

```
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "failure",
       "requestType": "PkiOperationPost",
       "reason": "The certificate authority is not active."
   }
}
```

Der Abruf des Zertifikats der Zertifizierungsstelle war erfolgreich

Der Connector für SCEP sendet ein `Certificate Authority Certificate Retrieval Succeeded` Ereignis, EventBridge wenn wir eine `GetCACert` Anfrage erhalten und das private CA-Zertifikat des Connectors erfolgreich abgerufen haben.

Im Folgenden finden Sie Beispieldaten für das Ereignis.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Authority Certificate Retrieval Succeeded",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "success",
       "requestType": "GetCACert"
   }
}
```

Fehler beim Abrufen des Zertifikats der Zertifizierungsstelle

Der Connector für SCEP sendet ein `Certificate Authority Certificate Retrieval Failed` Ereignis, EventBridge wenn wir eine `GetCACert` Anfrage erhalten und das private CA-Zertifikat des Connectors nicht abrufen können. Das Ereignis enthält den Grund für den Fehler.

Im Folgenden finden Sie Beispieldaten für das Ereignis.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Authority Certificate Retrieval Failed",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "failure",
       "requestType": "GetCACert",
       "reason": "The certificate authority certificate validity must be at least one
 year from today."
   }
}
```

Der Abruf des Zertifikats der Zertifizierungsstelle war erfolgreich

Der Connector für SCEP sendet ein `Certificate Authority Certificate Retrieval Succeeded` Ereignis, EventBridge wenn wir eine `GetCACert` Anfrage erhalten und das private CA-Zertifikat des Connectors erfolgreich abgerufen haben.

Im Folgenden finden Sie Beispieldaten für das Ereignis.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Authority Certificate Retrieval Succeeded",
   "source": "aws.pca-connector-scep",
```

```
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "success",
       "requestType": "GetCACert"
   }
}
```

Der Abruf der Funktionen der Zertifizierungsstelle war erfolgreich

Der Connector für SCEP sendet ein `Certificate Authority Capabilities Retrieval Succeeded` Ereignis, EventBridge wenn wir eine `GetCACaps` SCEP-Anfrage erhalten und die Funktionen der CA erfolgreich abgerufen haben.

Im Folgenden finden Sie Beispieldaten für das Ereignis.

Fehler beim Abrufen der Funktionen der Zertifizierungsstelle

Der Connector für SCEP sendet ein `Certificate Authority Capabilities Retrieval Failed` Ereignis, EventBridge wenn wir eine `GetCACaps` SCEP-Anfrage erhalten und die Funktionen der CA nicht abrufen können. Wir geben den Grund für das Scheitern in das Ereignis ein.

Im Folgenden finden Sie Beispieldaten für das Ereignis.

```
{
 "resources":
     [
     "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
     "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector11223344-1234-1122-2233-112233445566"
     ],
 "detailType":"Certificate Authority Capabilities Retrieval Failed",
 "detail": {
     "result":"failure",
```

```
    "requestType":"GetCACaps",
    "reason":"The request was denied due to request throttling."
},
"source":"aws.pca-connector-scep","accountId":"111122223333"
}
```

Nicht unterstützter Vorgang aufgerufen

Nicht unterstützter Vorgang aufgerufen

Connector für SCEP sendet ein `Unsupported Operation Invoked` Ereignis an, EventBridge wenn der an den Connector-Endpunkt gesendete Vorgang nicht unterstützt wird oder unbekannt ist.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Unsupported Operation Invoked",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {}
}
```

Erstellen Sie eine Regel EventBridge

In können Sie Regeln erstellen EventBridge, die auf Ereignisse reagieren, die von aufgezeichnet wurden CloudTrail. Um eine Regel zu erstellen, die alle von Connector for SCEP protokollierten Ereignisse enthält, legen Sie die Quelle auf `aws.pca-connector-scep` fest. Weitere Informationen zu Regeln finden Sie unter Eine Regel in Amazon erstellen EventBridge.

Log Connector für SCEP-API-Aufrufe mit AWS CloudTrail

Connector for Simple Certificate Enrollment Protocol (SCEP) ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der von einem Benutzer, einer Rolle, einem Client oder einem Dienst ausgeführten Aktionen bereitstellt. AWS CloudTrail erfasst alle API-Aufrufe für Connector

for SCEP als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Connector für SCEP-Konsole und Codeaufrufen an den Connector für SCEP-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für Connector for SCEP. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Connector for SCEP gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

Konnektor für SCEP-Informationen in CloudTrail

CloudTrail ist auf Ihrem aktiviert AWS-Konto , wenn Sie das Konto erstellen. Wenn im Connector für SCEP eine Aktivität auftritt, wird diese Aktivität zusammen mit anderen AWS Dienstereignissen im CloudTrail Ereignisverlauf in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem AWS-Konto anzeigen, suchen und herunterladen. Weitere Informationen finden Sie unter Ereignisse mit dem CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, einschließlich der Ereignisse für Connector for SCEP, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle Connector for SCEP-Aktionen werden von der Connector for SCEP-API-Referenz protokolliert CloudTrail und sind in dieser Dokumentation dokumentiert. Beispielsweise generieren Aufrufe von `GetConnector` und `CreateChallenge` Aktionen Einträge in den CloudTrail Protokolldateien. `CreateConnector`

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM-) Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.
- Ob die Anfrage von einem SCEP-Clientgerät gestellt wurde.

Weitere Informationen finden Sie unter CloudTrail -Element userIdentity.

Konnektor für SCEP-Verwaltungsereignisse

Connector for SCEP lässt sich in Connector for SCEP integrieren, CloudTrail um API-Aktionen aufzuzeichnen, die von einem Benutzer, einer Rolle oder einem AWS Dienst ausgeführt wurden. Sie können CloudTrail den Connector für SCEP-API-Anfragen in Echtzeit überwachen und Protokolle in Amazon Simple Storage Service, Amazon CloudWatch Logs und Amazon CloudWatch Events speichern. Connector für SCEP unterstützt die Protokollierung der folgenden Aktionen als Ereignisse in CloudTrail Protokolldateien:

- CreateChallenge
- CreateConnector
- GetConnector
- GetChallengeMetadata
- GetChallengePassword
- DeleteConnector
- DeleteChallenge

Konnektor für SCEP-Datenereignisse in CloudTrail

Datenereignisse liefern Informationen über die Ressourcenoperationen, die auf oder in einer Ressource ausgeführt werden, z. B. wenn Ihr Client eine `GetCACaps` SCEP-Nachricht an einen Connector-Endpunkt sendet. Sie werden auch als Vorgänge auf Datenebene bezeichnet. Datenereignisse sind oft Aktivitäten mit hohem Volume. Standardmäßig protokolliert CloudTrail es keine Datenereignisse, und der CloudTrail Ereignisverlauf zeichnet sie nicht auf.

Für Datenereignisse werden zusätzliche Gebühren fällig. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preisgestaltung.

Sie können Datenereignisse für den `AWS::PCAConnectorSCEP::Connector` Ressourcentyp mithilfe der CloudTrail Konsole oder CloudTrail API-Operationen protokollieren. AWS CLI Weitere Informationen zum Protokollieren von Datenereignissen finden Sie unter Protokollieren von Datenereignissen mit dem AWS-Managementkonsole und Protokollieren von Datenereignissen mit dem AWS Command Line Interface im AWS CloudTrail -Benutzerhandbuch.

In der folgenden Tabelle ist der Ressourcentyp Connector für SCEP aufgeführt, für den Sie Datenereignisse protokollieren können. In der Spalte Datenereignistyp (Konsole) wird der Wert angezeigt, den Sie in der Liste Datenereignistyp auf der CloudTrail Konsole auswählen können. In der Wertspalte resources.type wird der `resources.type` Wert angezeigt, den Sie bei der Konfiguration erweiterter Event-Selektoren mithilfe von oder angeben würden. AWS CLI CloudTrail APIs In der CloudTrail Spalte APIs Protokollierte Daten werden die API-Aufrufe angezeigt, die CloudTrail für den Ressourcentyp protokolliert wurden.

Typ des Datenereignisses (Konsole)	resources.type-Wert	Daten, die APIs protokolliert wurden CloudTrail
Konnektor	`AWS::PCAConnectorSCEP::Connector`	• `PKIOperationGet` — Wird generiert, wenn eine HTTP GET SCEP-Anfrage, die eine `PKCSReq` Nachricht enthält, an den Datenebenen-Endpunkt eines Connectors gestellt wird und der Betrieb dieser Nachricht auf eingestellt ist. `PKIOperation` • `PKIOperationPost` - Wird generiert, wenn eine HTTP POST-SCEP-Anfrage, die eine `PKCSReq` Nachricht enthält, an den Datenebenen-Endpunkt eines Connectors gestellt

Typ des Datenereignisses (Konsole)	resources.type-Wert	Daten, die APIs protokolliert wurden CloudTrail
		wird und der Betrieb dieser Nachricht auf eingestellt ist. `PKIOperation` • `GetCACaps` - Wird generiert, wenn eine SCEP-Anfrage, die eine `GetCACaps` Nachricht enthält, an den Dateneben en-Endpunkt eines Connectors gestellt wird. • `GetCACert` - Wird generiert, wenn eine SCEP-Anfrage, die eine `GetCACert` Nachricht enthält, an den Dateneben en-Endpunkt eines Connectors gestellt wird.

Sie können erweiterte Event-Selektoren so konfigurieren, dass sie nach den Feldern `eventName`, `readOnly` und `resources.`ARN filtern, sodass nur die Ereignisse protokolliert werden, die für Sie wichtig sind. Das folgende Beispiel ist die JSON-Ansicht einer Datenereigniskonfiguration, die nur Ereignisse für eine bestimmte Funktion protokolliert. Weitere Informationen zu diesen Kontingenten finden Sie unter AdvancedFieldSelector in derAWS CloudTrail -API-Referenz.

```
[
  {
    "name": "connector-scep-events",
    "fieldSelectors": [
      {
        "field": "eventCategory",
        "equals": [
          "Data"
        ]
      },
      {
```

```
        "field": "resources.type",
        "equals": [
          "AWS::PCAConnectorSCEP::Connector"
        ]
      },
      {
        "field": "resources.ARN",
        "equals": [
          "arn:aws:pca-connector-scep:US West (N.
 California):111122223333:connector/11223344-1122-2233-3344-cae95a00d2a7"
        ]
      }
    ]
  }
]
```

Beispieleinträge

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Beispiel 1: Verwaltungsereignis, **`CreateConnector`**

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `CreateConnector` Aktion demonstriert.

```
{
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
          "principalId": "AABB1122CCDD4455HHJJ1:11cc33nn2a97724dc48a89071111111111",
          "arn": "arn:aws:sts::111122223333:assumed-role/Admin",
          "accountId": "111122223333",
          "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "AABB1122CCDD4455HHJJ1",
```

```
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "my-user-name"
            },
            "attributes": {
                "creationDate": "2024-08-16T17:46:41Z",
                "mfaAuthenticated": "false"
            }
          }
        },
        "eventTime": "2024-08-16T17:48:07Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "CreateConnector",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "requestParameters": {
          "ClientToken": "11223344-2222-3333-4444-666555444555",
          "CertificateAuthorityArn": "arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
        },
        "responseElements": {
          "ConnectorArn": "arn:aws:pca-connector-scep:us-east-1:111122223333:connector/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
        },
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
        }
```


Beispiel 2: Management-Ereignis, **`CreateChallenge`**

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `CreateChallenge` Aktion demonstriert.


```
{
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
```

```
          "principalId": "AABB1122CCDD4455HHJJ1:11cc33nn2a97724dc48a89071111111111",
          "arn": "arn:aws:sts::111122223333:assumed-role/Admin",
          "accountId": "111122223333",
          "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "AABB1122CCDD4455HHJJ1",
                  "arn": "arn:aws:iam::111122223333:role/Admin",
                  "accountId": "111122223333",
                  "userName": "user-name"
              },
              "attributes": {
                  "creationDate": "2024-08-16T17:46:41Z",
                  "mfaAuthenticated": "false"
              }
          }
        },
        "eventTime": "2024-08-16T17:47:52Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "CreateChallenge",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "requestParameters": {
          "ConnectorArn": "arn:aws:pca-connector-scep:us-east-1:111122223333:connector/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
          "ClientToken": "11223344-2222-3333-4444-666555444555"
        },
        "responseElements": {
          "Challenge": {
              "Arn": "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/9cac40bc-acba-412e-9a24-f255ef2fe79a/a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222",
              "ConnectorArn": "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
              "CreatedAt": 1723830472.942,
              "Password": "***",
              "UpdatedAt": 1723830472.942
          }
        },
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
```

```
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
        }
```

Beispiel 3: Verwaltungsereignis, **`GetChallengePassword`**

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `GetChallengePassword` Aktion demonstriert.

```
{
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
          "principalId": "AABB1122CCDD4455HHJJ1:11cc33nn2a97724dc48a89071111111111",
          "arn": "arn:aws:sts::111122223333:assumed-role/Admin",
          "accountId": "111122223333",
          "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "AABB1122CCDD4455HHJJ1",
                  "arn": "arn:aws:iam::111122223333:role/Admin",
                  "accountId": "905418114790",
                  "userName": "111122223333"
              },
              "attributes": {
                  "creationDate": "2024-08-16T17:55:01Z",
                  "mfaAuthenticated": "false"
              }
          },
          "invokedBy": "signin.amazonaws.com"
        },
        "eventTime": "2024-08-16T17:55:54Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "GetChallengePassword",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "requestParameters": {
          "ChallengeArn": "arn:aws:pca-connector-scep:us-east-1:111122223333:challenge/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
```

```
        },
        "responseElements": null,
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
        }
```

Beispiel 4: Datenereignis, **`PkiOperationPost`**

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der einen fehlgeschlagenen `PkiOperationPost` Anruf demonstriert. Das Protokoll enthält einen Fehlercode und eine Fehlermeldung mit einer Erklärung des Fehlers.

```
{
        "eventVersion": "1.10",
        "userIdentity": {
          "type": "FederatedUser",
          "principalId": "111122223333",
          "accountId": "111122223333"
        },
        "eventTime": "2024-08-16T17:40:09Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "PkiOperationPost",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "errorCode": "BadRequestException",
        "errorMessage": "The certificate authority is not in a valid state for issuing
 certificates (Service: AcmPca, Status Code: 400, Request ID: a1b2c3d4-5678-90ab-cdef-
EXAMPLE55555)",
        "requestParameters": null,
        "responseElements": null,
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
        "resources": [
          {
              "accountId": "111122223333",
              "type": "AWS::PCAConnectorSCEP::Connector",
```

```
            "ARN": "arn:aws:pca-connector-scep:us-east-1:111122223333:connector/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
          }
        ],
        "eventType": "AwsApiCall",
        "managementEvent": false,
        "recipientAccountId": "905418114790",
        "eventCategory": "Data",
        "tlsDetails": {
          "clientProvidedHostHeader": "111122223333-a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333.enroll.pca-connector-scep.us-east-1.api.aws"
        }
        }
```

Beheben Sie Probleme mit dem AWS Private Certificate Authority Connector für SCEP-Probleme

Möglicherweise müssen Sie Probleme im Zusammenhang mit Ihrer Connector für SCEP-Implementierung beheben. Dieses Kapitel enthält detaillierte Informationen zu den vom Dienst gesendeten HTTP- und Client-Fehlern.

Themen

- Beheben Sie HTTP-Fehler vom Connector für SCEP
- Beheben Sie Fehler beim Connector für SCEP-Clients

Beheben Sie HTTP-Fehler vom Connector für SCEP

Wenn Ihr Client eine API-Aktion für Connector for SCEP-Datenebene auslöst und dies zu einem Fehler führt, sendet Connector für SCEP einen HTTP-Antwortcode mit Informationen über den Fehler an den anfragenden Client.

Zusätzlich zu den Serviceantworten, die Ihren Kunden direkt zur Verfügung gestellt werden, können Sie die im Monitor-Connector für SCEP Abschnitt beschriebenen Überwachungstools verwenden, um Fehler, die zu einem HTTP-Fehler führen, anzuzeigen und zu debuggen.

Im Folgenden finden Sie Fehlermeldungen, die der Dienst an die SCEP-Clients zurückgibt, die möglichen Ursachen und die Schritte, die Sie zur Behebung der Probleme ergreifen können.

Ungültige HTTP 400-Anfrage

Ein HTTP 400-Antwortcode bedeutet, dass Connector für SCEP die Anfrage aufgrund eines offensichtlichen Client-Fehlers, z. B. fehlender oder ungültiger Daten in der Anfrage, nicht verarbeiten kann. Wenn der Fehler auf einen für das SCEP-Protokoll spezifischen Fehler zurückzuführen ist, nimmt Connector for SCEP die SCEP-Antwort als Binärdatei in die Nachricht auf. Connector für SCEP APIs kann aus einem der folgenden Gründe 400 Antworten zurückgeben.

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
LimitExceededException	Das Ausstellungslimit der Zertifizierungsstelle wurde überschritten.	Die dem Connector zugeordnete private Zertifizierungsstelle (CA) hat ihr Kontingent für die Anzahl der Zertifikate, die sie ausstellen kann, überschritten.	Ein SCEP-Connector kann während seiner gesamten Lebensdauer nur mit einer privaten Zertifizierungsstelle verbunden werden. Wenn Sie die Grenzen Ihrer privaten CA ausgeschöpft haben, erstellen Sie entweder einen neuen Connector oder beantragen Sie eine Erhöhung des Kontingents. Weitere Informationen zu Kontingenten für private Zertifizierungsstellen finden Sie unter AWS Private	Nein

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
			Certificate Authority Kontingente.	
ValidationException	Die Anfrage muss Base64 enthalten.	Der Connector für SCEP kann die HTTP-GET-Anfrage nicht verarbeiten, da der Hauptteil kein gültiges Base64-Format hat.	Wenn möglich, konfigurieren Sie Ihre Clients so, dass sie HTTP-POST-Nachrichten anstelle von HTTP-GET-Nachrichten verwenden. Wenn Sie HTTP GET verwenden müssen, müssen die Nachrichten das Base64-Format verwenden. Wenn Ihre Clients diese Anforderungen nicht erfüllen, wenden Sie sich an uns, AWS Supportum Unterstützung zu erhalten.	Nein
ValidationException	Die Zertifizierungsstelle ist nicht aktiv.	Die dem Connector zugeordnete private CA ist inaktiv.	Reaktivieren Sie die private CA. Weitere Informationen finden Sie unter Aktualisieren Sie eine private Zertifizierungsstelle in AWS Private Certificate Authority.	Nein

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
ValidationException	Das Zertifikat der Zertifizierungsstelle muss ab heute mindestens ein Jahr gültig sein.	Die private Zertifizierungsstelle, die dem Allzweck-Connector zugeordnet ist, muss ab heute eine Gültigkeitsdauer von einem Jahr haben.	Stellen Sie das Zertifikat mit einer Gültigkeitsdauer von mehr als einem Jahr ab heute erneut aus. Informationen zur Verwaltung von Zertifikaten finden Sie unterVerwalten Sie den privaten CA-Lebenszyklus.	Nein
ValidationException	Das in der Anfrage enthaltene Zertifikat ist abgelaufen.	Das vom Client-Gerät bei jeder Transaktion generierte vorübergehende Zertifikat war bei Empfang durch den Dienst abgelaufen.	Es ist sehr wahrscheinlich, dass die Zeiteinstellungen Ihrer Client-Geräte nicht richtig konfiguriert sind und dass sie Zertifikate erstellen, deren Datum hinter der Echtzeit zurückliegt. Wenn Sie dieses Problem nicht lösen können, wenden Sie sich an uns, um AWS SupportUnterstützung zu erhalten.	Nein

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
ValidationException	Die Anfrage enthält eine ungültige kryptografische Nachrichtensyntax.	Der Dienst konnte die SCEP-Anforderungsnachricht nicht dekodieren.	Überprüfen Sie, ob Ihre SCEP-Nachrichten der kryptografischen Nachrichtensyntax entsprechen, die in SCEP RFC 8894 definiert ist. Wenn Sie dieses Problem nicht lösen können, wenden Sie sich an uns, um Unterstützung zu erhalten. AWS Support	Nein

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
ValidationException	Der Connector ist nicht aktiv.	Der Status des Connectors ist nicht aktiv.	Den Status eines Connectors finden Sie in der Konsole oder im Statusfeld in der API. Der Status eines Connectors kann „Erstellt“, „Aktiv“, „Löschen“ oder „Fehlgeschlagen“ lauten. Wenn der Status „Wird erstellt“ lautet, versuchen Sie es später mit Ihrer Anfrage. Wenn der Status „Fehlgeschlagen“ lautet, sehen Sie sich den Grund für den Status an, um das Problem zu beheben, und erstellen Sie dann einen neuen Connector.	Nein

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
ValidationException	Die Anfrage muss ein gültiges Zertifikat enthalten.	Das in der Anforderungsnachricht des Clients enthaltene vorübergehende Zertifikat fehlte entweder oder war ungültig.	SCEP-kompatible Clients müssen ein selbstsigniertes Zertifikat bereitstellen, um sich zu authentifizieren. Wenn Ihr Kunde das erforderliche selbstsignierte Zertifikat nicht bereitstellen kann, wenden Sie sich an uns, um Unterstützung zu erhalten. AWS Support	Nein
ValidationException	Die Anforderungs-URI ist ungültig.	Der Connector für SCEP kann die Anfrage nicht analysieren, da der URI-Pfad oder die Abfrage der Anfrage ungültig sind.	Administratoren sollten die Konfigurationseinstellungen der Client-Geräte überprüfen, die in der Regel über ein MDM-System (Mobile Device Management) verwaltet werden. Weitere Informationen finden Sie unter Schritt 2: Kopieren Sie die Konnektordetails in Ihr MDM-System.	Nein

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
ValidationException	In der Anfrage ist genau ein Host-Header erforderlich.	Der Client hat in der Anfrage keinen gültigen HTTP-Host-Header angegeben, der für die Verarbeitung der Anfrage erforderlich ist.	Der HTTP-Host-Header ist erforderlich, um Anfragen zu unterscheiden, die an verschiedene Konnektoren gesendet werden. Wenn Ihr Client den erforderlichen HTTP-Host-Header nicht bereitstellen kann, wenden Sie sich an uns, um AWS SupportUnterstützung zu erhalten.	Nein
ValidationException	Die Anfrage konnte nicht dekodiert werden. Bitte senden Sie eine gültige SCEP-Anfrage.	Der Dienst konnte die CMS-Anfrage (Cryptographic Message Syntax), die Ihr Client gesendet hat, nicht dekodieren und verarbeiten.	Wenn Ihre Kunden Probleme mit unserer Implementierung von SCEP haben, notieren Sie sich die Anfrage-ID (`x-amzn-requestid`) aus der Antwort und kontaktieren Sie uns. AWS Support	Nein

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
ValidationException	Die Antwort konnte nicht mit den aus der Anfrage abgeleiteten Werten codiert werden. Bitte senden Sie eine gültige SCEP-Anfrage.	Der Dienst konnte die SCEP-Antwort nicht verschlüsseln.	Dieses Problem tritt normalerweise auf, wenn der Dienst das bereitgestellte Anforderungszertifikat nicht verwenden kann, um die SCEP-Antwortnachricht ordnungsgemäß zu verschlüsseln. Dies kann beispielsweise der Fall sein, wenn das anfordernde Zertifikat über einen ECDSA-Schlüssel (Elliptic Curve Digital Signature Algorithm) verfügt, den Connector for SCEP nicht unterstützt. Wenn dieses Problem auftritt, konfigurieren Sie zunächst Ihren MDM- oder SCEP-Client für die Verwendung von RSA. Wenn Sie das Problem immer noch nicht lösen können,	Nein

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
			notieren Sie sich die Anfrage-ID (`x-amzn-requestid`) aus der Antwort und wenden AWS SupportSie sich an uns, um Unterstützung zu erhalten.	

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
ValidationException	Algorithmus wird nicht unterstützt: <OID>	Die Anfrage wurde entweder signiert oder mit einem nicht unterstützten kryptografischen Algorithmus verschlüsselt.	Unser Service unterstützt bestimmte veraltete und schwache kryptografische Algorithmen nicht. Diese Informationen werden den Kunden im Rahmen der `GetCACaps` Anfrage mitgeteilt. Einige Clients verwenden diese Methode jedoch möglicherweise nicht, um die unterstützten Algorithmen zu überprüfen. Wenn Ihre Kunden mit den von unserem Service unterstützten kryptografischen Algorithmen nicht kompatibel zu sein scheinen, wenden Sie sich an uns, AWS Supportum Unterstützung zu erhalten.	Nein

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
ValidationException	PkiOperation MessageType wird nicht unterstützt.	Die Anforderungsnachricht enthielt einen ungültigen `PkiOperation` Nachrichtentyp und konnte vom Dienst nicht verarbeitet werden.	Unser Service unterstützt nur eine Teilmenge der in RFC 8894 definierten Nachrichtentypen des SCEP-Protokolls. Insbesondere erkennen und verarbeiten wir die folgenden Nachrichtentypen: CertRep,, PKCSReq GetCert, GetCRL und. CertPoll Wir teilen den Clients die unterstützten Nachrichtentypen über die CACaps Get-Methode mit. Leider verwenden einige Kunden diese Methode möglicherweise nicht und entsprechen möglicherweise nicht den Funktionen unseres Dienstes. Wenn Ihre Kunden mit den von unserem Service unterstützten	Nein

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
			SCEP-Nachrichtentypen nicht kompatibel zu sein scheinen, wenden Sie sich an. AWS Support	
BadRequestException	Das Challenge-Passwort ist ungültig.	Das vom Client angegebene Challenge-Passwort war für den kontaktierten Dienstendpunkt und den zugehörigen Connector ungültig. Das Challenge-Passwort ist eine erforderliche Sicherheitsmaßnahme, die im SCEP-Protokoll definiert ist, um sicherzustellen, dass nur autorisierte Clients auf den Service zugreifen können.	Stellen Sie sicher, dass Ihr Kunde in seiner Anfrage das richtige Challenge-Passwort angibt. Sie finden sie in den Connector-Details in der Konsole oder über die GetChallengePasswordAPI. Weitere Informationen finden Sie unter Schritt 2: Kopieren Sie die Konnektordetails in Ihr MDM-System.	Ja

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
BadRequestException	In der Anfrage zum Signieren des Zertifikats ist genau ein Challenge-Passwort erforderlich.	Der Client hat in seiner Anfrage entweder kein oder mehrere Challenge-Passwörter angegeben.	Stellen Sie sicher, dass Ihr Kunde in seiner Anfrage ein Challenge-Passwort angibt. Sie finden Challenge-Passwörter in den Connector-Details in der Konsole oder über die GetChallengePasswordAPI. Weitere Informationen finden Sie unter Schritt 2: Kopieren Sie die Konnektordetails in Ihr MDM-System.	Ja
BadRequestException	Der Connector hat keinen Zugriff auf Azure.	Connector für Microsoft Intune autorisiert Client-Anfragen über Microsoft Intune. Dazu müssen Sie Connector for SCEP die Erlaubnis erteilen, auf Ihre Azure-Ressourcen zuzugreifen.	Konfigurieren Sie die Berechtigungen, die unter beschrieben sind. Schritt 1: Erteilen Sie die AWS Private CA Erlaubnis zur Nutzung Ihrer Microsoft Entra ID-Anwendung	Ja

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
BadRequestException	Die Azure-Anwendung hat keinen Zugriff auf die Ausführung<action>.	Connector für Microsoft Intune autorisiert Client-Anfragen über Microsoft Intune. Dazu müssen Sie Connector for SCEP die Erlaubnis erteilen, auf Ihre Azure-Ressourcen zuzugreifen.	Konfigurieren Sie die Berechtigungen, die unter beschrieben sind. Schritt 1: Erteilen Sie die AWS Private CA Erlaubnis zur Nutzung Ihrer Microsoft Entra ID-Anwendung	Ja
BadRequestException	Die Azure-Anwendung wurde nicht gefunden.	Connector für Microsoft Intune autorisiert Client-Anfragen über Microsoft Intune. Dieser Fehler weist darauf hin, dass Ihre Microsoft Entra ID keine App-Registrierung enthält oder dass die Intune-Details Ihres Connectors falsch konfiguriert sind.	Folgen Sie den Anleitungen im Thema. Microsoft Intune für Connector für SCEP konfigurieren	Ja

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
BadRequestException	Die Überprüfung der Intune-Zertifikatsignieranforderung ist fehlgeschlagen. Grund: <reason>	Connector für Microsoft Intune autorisiert Client-Anfragen über Microsoft Intune. Diese Fehlermeldung weist darauf hin, dass der Intune-Validierungsprozess fehlgeschlagen ist, und der entsprechende Intune-Fehlercode wird bereitgestellt.	Folgen Sie den Anleitungen im Microsoft Intune für Connector für SCEP konfigurieren Thema. Wenn das Problem weiterhin besteht, wenden Sie sich an den Microsoft-Support.	Ja

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
BadRequestException	<message type>Nicht unterstützter PkiOperation messageType:.	Die Anforderungsnachricht enthielt einen ungültigen Nachrichtentyp und konnte vom Dienst nicht verarbeitet werden.	Unser Service unterstützt nur eine Teilmenge der in RFC 8894 definierten Nachrichtentypen des SCEP-Protokolls. Insbesondere erkennen und verarbeiten wir die folgenden Nachrichtentypen: CertRep,, PKCSReq GetCert, GetCRL und. CertPoll Wir teilen den Clients die unterstützten Nachrichtentypen über die CACaps Get-Methode mit. Leider verwenden einige Kunden diese Methode möglicherweise nicht und entsprechen möglicherweise nicht den Funktionen unseres Dienstes. Wenn Ihre Kunden mit den von unserem Service unterstützten SCEP-Nachrichtenty	Ja

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
			pen nicht kompatibel zu sein scheinen, wenden Sie sich an. AWS Support	
BadRequestException	Der Schlüsselalgorithmus oder die Schlüssellänge werden nicht unterstützt.	Der Dienst unterstützt den bereitgestellten öffentlichen Schlüssel, der in der Zertifikatsignieranforderung enthalten ist, nicht.	Unser Service unterstützt nur Standard-RSA-Schlüssel mit bis zu 16.384 Bit und ECDSA-Schlüssel mit bis zu 521 Bit. Wenn Ihre Kunden einen Algorithmus benötigen, der derzeit nicht unterstützt wird, wenden Sie sich bitte an uns, um Unterstützung zu erhalten. AWS Support	Ja

HTTP 401 Nicht autorisiert

Der Statuscode 401 „Nicht autorisiert“ gibt an, dass die Client-Anforderung nicht abgeschlossen wurde, da sie keine gültigen Authentifizierungsdaten für die angeforderte Ressource enthält.

Antwortheader (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?

Antworthe ader (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhalte t die SCEP-Antwort?
AccessDen iedExcept ion	Der Connector hat keinen Zugriff auf die Zertifizierungsstelle.	Der Connector für SCEP hat keinen Zugriff auf die dem Connector zugeordne te private CA.	Teilen Sie Ihre private CA mit dem Connector für SCEP, indem Sie. AWS Resource Access Manager	Nein
AccountDo esNotExis tExceptio n	Das AWS Konto existiert nicht.	Die Ressource Connector für SCEP ist nicht mehr vorhanden.	Das Konto, dem die Zielressource gehört, wurde gelöscht. Falls dies versehent lich geschehen ist, wenden Sie sich AWS Supportinnerhalb der Frist von 90 Tagen nach der Schließung an uns.	Nein

HTTP 404 nicht gefunden

Ein HTTP 404-Antwortcode bedeutet normalerweise, dass die gesuchte Ressource nicht gefunden werden konnte.

Antwort-Header (x-amzn-ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhalte t die SCEP-Antwort?
ResourceN otFoundEx ception	Die Zertifizierungsste lle ist nicht vorhanden.	Die dem Connector zugeordnete private CA wurde gelöscht.	Es gibt eine Übergangsfrist, in der eine private Zertifizi	Nein

Antwort-Header (x-amzn-ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
			erungsstelle (CA) wiederhergestellt werden kann, falls sie versehentlich gelöscht wurde. Weitere Informationen finden Sie unter Stellen Sie eine private CA wieder her.	
ResourceNotFoundException	Ein Connector mit Endpunkt <URL>ist nicht vorhanden.	Das Client-Gerät hat versucht, eine Verbindung zu einer URL herzustellen, die zu keinem vorhandenen Connector gehört.	Stellen Sie sicher, dass Ihr Client den richtigen Endpunkt für den Connector bereitstellt. Um die eines Connectors anzuzeigen`nEndpoint`, rufen Sie die GetConnectorAPI auf oder zeigen Sie sie auf der Detailseite des Connectors in der Konsole an.	Nein

HTTP 409-Konflikt

Eine Antwort auf einen HTTP 409-Konflikt signalisiert, dass sich eine private CA, die einem Connector zugeordnet ist, seit die Anfrage initiiert wurde, geändert hat.

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
ConflictException	Der Konnektor hat sich geändert, seit die Anfrage initiiert wurde.	Die dem Connector zugeordnete private CA wurde aktualisiert, wodurch eine Rotation des internen Zertifikats des Connectors ausgelöst wurde, das für die Kommunikation mit Client-Geräten über SCEP verwendet wird. Diese Zertifikatsrotation kann während des Aktualisierungszeitraums zu vorübergehenden Problemen führen, da das neue Zertifikat bereitgestellt wird. Dieser Fehler sollte jedoch rechtzeitig automatisch behoben werden.	Versuchen Sie es in ein paar Minuten erneut mit Ihrer Anfrage. Wenn das Problem nicht behoben werden kann, wenden Sie sich an uns, um AWS SupportUnterstützung zu erhalten.	Nein

HTTP 429 Zu viele Anfragen

Der Connector für SCEP hat Kontingente auf Kontoebene pro Region. Wenn Sie das Limit für Anfragen an einen Connector überschreiten, werden Ihre Anfragen mit einem HTTP 429-Fehler abgelehnt. Wenn Sie Ihr Kontingent erhöhen müssen, finden Sie weitere Informationen unter AWS Private Certificate Authority Endpunkte und Kontingente.

Antwort-Header (x-amzn-) ErrorType	Fehlermeldung (x-amzn-) ErrorMessage	Ursache	Abhilfe	Beinhaltet die SCEP-Antwort?
ThrottlingException	Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.	Es wurden zu viele Anfragen an diesen Connector gesendet, sodass einige Anfragen abgelehnt wurden. Diese Zertifikatsrotation kann während des Aktualisierungszeitraums zu vorübergehenden Problemen führen, da das neue Zertifikat bereitgestellt wird. Dieser Fehler sollte jedoch rechtzeitig automatisch behoben werden.	Wenn Sie das Limit für Anfragen an einen Connector überschreiten, werden Ihre Anfragen abgelehnt. Wenn Sie Ihr Kontingent erhöhen müssen, finden Sie weitere Informationen unter Connector für SCEP-Endpunkte und Kontingente.	Nein

Beheben Sie Fehler beim Connector für SCEP-Clients

Verwenden Sie die folgenden Anleitungen, um Client-Fehler im Zusammenhang mit Connector für SCEP zu beheben.

Beispiel für eine Nachricht	Ursache	Lösung
ECDSA-Schlüssel werden nicht unterstützt	Der Connector ist mit einer privaten CA verbunden, die einen ECDSA-Schlüssel anstelle von RSA	Erwägen Sie die Verwendung einer RSA-verschlüsselten privaten CA anstelle von ECDSA. Wenn Sie

Beispiel für eine Nachricht	Ursache	Lösung
	verwendet. Dieser Dienst unterstützt zwar ECDSA-Schlüssel, aber möglicherweise sind nicht alle Client-Geräte mit diesem Algorithmus kompatibel.	eine private CA erstellen, die RSA verwendet, müssen Sie auch einen neuen Connector erstellen. Ein Connector kann während seiner gesamten Lebensdauer nur an eine private CA gebunden werden.
Das Verschlüsselungs- oder Signaturzertifikat ist nicht vorhanden	Gemäß RFC 8894 gibt ein SCEP-Dienst CA-Zwischenzertifikate an den Client zurück. Diese Zertifikate werden vom Client verwendet, um Verschlüsselungs- und Signaturvalidierungsvorgänge als Teil des SCEP-Protokolls durchzuführen. Connector for SCEP verwendet dasselbe Zertifikat sowohl für Verschlüsselungs- als auch für Signaturvalidierungszwecke, was ein gängiger Ansatz ist. Einige Kunden erwarten jedoch möglicherweise, stattdessen über zwei separate Zertifikate zu verfügen.	Wenn Sie keine kompatiblen Clients verwenden können, wenden Sie sich an uns, um AWS SupportUnterstützung zu erhalten.

AWS Private CA Servicekontingenten

AWS Private CA weist Ihrer zulässigen Anzahl von Zertifikaten und Zertifizierungsstellen Kontingente zu. Die Anforderungsraten für API-Aktionen unterliegen ebenfalls Kontingenten. AWS Private CA Kontingente sind für ein AWS Konto und eine Region spezifisch.

AWS Private CA drosselt API-Anfragen je nach API-Vorgang unterschiedlich schnell. Drosselung bedeutet, dass eine ansonsten gültige Anfrage AWS Private CA zurückgewiesen wird, weil die Anfrage das Kontingent des Vorgangs für die Anzahl der Anfragen pro Sekunde überschreitet. Wenn eine Anforderung gedrosselt wird, wird ein Fehler zurückgegeben. AWS Private CA ThrottlingException AWS Private CA garantiert keine Mindestanforderate für. APIs

Welche Kontingente angepasst werden können, finden Sie in der Tabelle mit den AWS Private CA Kontingenten im Allgemeine AWS-Referenz.

Mithilfe von AWS Service Quotas können Sie Ihre aktuellen Kontingente einsehen und Kontingenterhöhungen beantragen.

Um eine up-to-date Liste Ihrer AWS Private CA Kontingente einzusehen

1. Loggen Sie sich in Ihr AWS Konto ein.
2. Öffnen Sie die Service Quotas-Konsole unter https://console.aws.amazon.com/servicequotas/.
3. Wählen Sie in der Liste Dienste die Option AWS Certificate Manager Private Certificate Authority (ACM PCA) aus. Jedes Kontingent in der Liste der Dienstkontingente zeigt Ihren aktuell angewendeten Kontingentwert, den Standardkontingentwert und ob das Kontingent anpassbar ist oder nicht. Wählen Sie den Namen eines Kontingents, um weitere Informationen dazu zu erhalten.

So fordern Sie eine Kontingenterhöhung an

1. Wählen Sie in der Liste der Dienstkontingente das Optionsfeld für ein einstellbares Kontingent.
2. Wählen Sie die Schaltfläche Erhöhung des Kontingents beantragen.
3. Füllen Sie das Formular „Kontingenterhöhung beantragen“ aus und senden Sie es ab.

AWS Private CA ist integriert in AWS Certificate Manager. Sie können die ACM-Konsole oder die ACM-API verwenden AWS CLI, um private Zertifikate von einer vorhandenen privaten

Zertifizierungsstelle anzufordern. Diese privaten PKI-Zertifikate, die von ACM verwaltet werden, unterliegen sowohl den PCA-Kontingenten als auch den Kontingenten, die ACM für öffentliche und importierte Zertifikate festlegt. Weitere Informationen zu den ACM-Anforderungen finden Sie unter Privates Zertifikat und Kontingente beantragen im Benutzerhandbuch. AWS Certificate Manager

Dokumentverlauf

Die folgende Tabelle beschreibt signifikante Änderungen an dieser Dokumentation seit Januar 2018. Neben den hier aufgelisteten größeren Änderungen aktualisieren wir die Dokumentation regelmäßig überarbeitet, um Beschreibungen und Beispiele zu verbessern und Ihre Rückmeldungen zu berücksichtigen. Wenn Sie über signifikante Änderungen benachrichtigt werden möchten, verwenden Sie den Link oben rechts, um den RSS-Feed zu abonnieren.

Änderung	Beschreibung	Datum
Aktualisierung der Dokumentation	Secure Kubernetes wurde AWS Private Certificate Authority mit einem neuen Verfahren für die ersten Schritte, Beispielen und Themen zur Überwachung und Fehlerbehebung aktualisiert.	1. Oktober 2025
Dual-Stack-Unterstützung	AWS Private Certificate Authority unterstützt Dual-Stack.	23. Juni 2025
Die Unterstützung untergeordneter Domänen für Connector for AD ist jetzt allgemein verfügbar	Sie können jetzt Connector für AD mit Ihrer untergeordneten Domain einrichten.	2. Juni 2025
Neue verwaltete Richtlinie: `AWSPrivateCAConnectorForKubernetesPolicy`	Neue verwaltete Richtlinie für die Verwendung mit AWS Private CA Connector for Kubernetes eingeführt.	19. Mai 2025
Aktualisierte `AWSPrivateCAPrivilegedUser` und verwaltete Richtlinien `AWSPrivateCAUser`	Ersetzt `StringLike` durch `ArnLike` in `AWSPrivateCAUser` und`AWSPrivateCAPrivilegedUser` . Die Vorlage ARN wurde	22. Januar 2025

	aktualisiert und enthält nun `arn:aws:acm-pca:::template` Platzhalter für `arn:aws:acm-pca:*:*:template` .	
Der Connector für SCEP ist jetzt allgemein verfügbar	Der Connector für SCEP ist jetzt allgemein verfügbar.	16. September 2024
Neues Thema zur Problembehandlung	Es wurde ein neues Thema hinzugefügt, das Ihnen bei der Behebung von Problemen im Zusammenhang mit der Aktualisierung Ihrer Connector für Active Directory-Vorlagen hilft.	31. Juli 2024
Es wurde hinzugefügt, wie man Connector für AD-Vorlagen aktualisiert	Es wurde ein Verfahren hinzugefügt, das beschreibt, wie eine Connector for AD-Vorlage aktualisiert wird und wie diese AWS Private CA Updates weitergegeben werden.	31. Juli 2024
Der Connector für SCEP für Omnissa Workspace ONE ist jetzt allgemein verfügbar	Der Connector für SCEP für Omnissa Workspace ONE ist jetzt allgemein verfügbar.	23. Juli 2024
Einschränkung für Auditberichte hinzugefügt	AWS Private CA unterstützt nicht die Verwendung von Amazon S3 Object Lock mit Buckets, die für Auditberichte verwendet werden.	3. Juli 2024

Unterstützt jetzt SM2 die Region China	AWS Private CA unterstützt jetzt den SM2 Signaturalgorithmus, nur für die Region China.	27. Juni 2024
AWS Private CA unterstützt jetzt Connector für SCEP (Preview)	Verwenden Sie Connector for SCEP, um eine Verbindung AWS Private CA zu Ihren SCEP-fähigen Clients und Geräten herzustellen.	11. Juni 2024
Neue Anleitung zur Fehlerbehebung bei Konnektoren	Es wurden neue Abschnitte zur Behebung von Fehlern bei der Connector- und SPN-Erstellung hinzugefügt.	4. April 2024
CDP-Erweiterung für Matter hinzugefügt	Fügt Unterstützung für die CDP-Erweiterung (Certificate Revocation List Distribution Point) für Matter hinzu.	25. Januar 2024
AWS Private CA API-Unterstützung für mDL	API-Unterstützung für die Erstellung von Zertifikaten hinzugefügt, die dem ISO/IEC-Standard für den mobilen Führerschein (MdL) entsprechen.	16. Januar 2024
AWS Private CA Konnektor für Active Directory	Benutzerhandbuch, API- und CLI-Unterstützung für Connector for AD. Weitere Informationen finden Sie in der Dokumentation zum Connector für AD.	24. August 2023

Die Namen der Sicherheitsrichtlinien werden so geändert, dass sie dem neuen Dienstnamen entsprechen	Einführung neuer Namen für AWS verwaltete IAM-Richtlinien, die Standardberechtigungen für festlegen. AWS Private CA Weitere Informationen finden Sie unter AWS Verwaltete Richtlinien.	13. Februar 2023
Change Tracker für AWS verwaltete Richtlinien hinzufügen	Es wurde eine Dokumentation hinzugefügt, um Änderungen an AWS verwalteten IAM-Richtlinien nachzuverfolgen, die Standardberechtigungen für festlegen. AWS Private CA Weitere Informationen finden Sie unter Aktualisierungen der AWS verwalteten Richtlinien für AWS Private CA.	11. November 2022
API- und CLI-Unterstützung für CAs die Ausgabe kurzlebiger Zertifikate	Mit der Einführung von CA-Nutzungsmodi kann eine Zertifizierungsstelle so konfiguriert werden, dass sie entweder allgemeine oder ausschließlich kurzlebige Zertifikate ausstellt. Weitere Informationen finden Sie unter Zertifizierungsstellenmodi.	24. Oktober 2022

Umbenennung des Dienstes und Aktualisierung der Konsole	Der Dienst wurde in AWS Private Certificate Authority (AWS Private CA) umbenannt. Die Bedienbarkeit der AWS Private CA Konsole wurde verbessert, einschließlich integrierter Hilfebereiche, die auf die vollständige Dokumentation verweisen.	27. September 2022
Unterstützung für Matter-konforme Zertifikate	Drei neue Zertifikatsvorlagen bieten Unterstützung für Matter-konforme CA- und Endentitätszertifikate. Weitere Informationen finden Sie unter Grundlegendes zu Zertifikatsvorlagen.	20. Juli 2022
Unterstützung für neue Regionen	Endpunkt für Asien-Pazifik (Jakarta) hinzugefügt. Eine vollständige Liste der AWS Private CA Endpunkte finden Sie unter ACM Private Certificate Authority Endpoints and Quotas.	4. Mai 2022
Support für benutzerdefinierte Attribute und Erweiterungen	Verwenden Sie das CustomAttributeObjekt, um benutzerdefinierte Zertifikate CAs und Zertifikate zu konfigurieren, und das CustomExtension Objekt, um benutzerdefinierte Zertifikate zu konfigurieren.	16. März 2022

Support für Managed OCSP	Informationen zu Sperroptionen, einschließlich OCSP, finden Sie unter Einrichten einer Methode zum Widerruf von Zertifikaten.	18. August 2021
Support für die S3-Funktion Block Public Access für CRLs	Weitere Informationen finden Sie unter Aktivieren der S3-Funktion zum Blockieren des öffentlichen Zugriffs.	27. Mai 2021
Beispiele für neue und aktualisierte Java-Implementierungen	Siehe Verwenden der privaten CA-API von ACM (Java-Beispiele).	9. September 2020
Unterstützung für neue Regionen	Endpunkte für Afrika (Kapstadt) und Europa (Mailand) hinzugefügt. Eine vollständige Liste der AWS Private CA Endpunkte finden Sie unter AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	27. August 2020
Kontoübergreifender Zugriff auf private Zertifizierungsstellen wird unterstützt	AWS Certificate Manager Benutzer können autorisiert werden, private Zertifikate auszustellen CAs , die sie nicht besitzen. Weitere Informationen finden Sie unter Kontoübergreifender Zugriff auf Private CAs.	17. August 2020

Unterstützung VPC VPC-Endpunkte () PrivateLink	Unterstützung für die Verwendung von VPC-Endpunkten (AWS PrivateLink) für verbesserte Netzwerksicherheit hinzugefügt. Weitere Informationen finden Sie unter ACM Private CA VPC Endpoints ().AWS PrivateLink	26. März 2020
Spezieller Sicherheitsbereich hinzugefügt	Die Sicherheitsdokumentation für AWS wurde in einem eigenen Sicherheitsbereich zusammengefasst. Informationen zur Sicherheit finden Sie unter Security in AWS Certificate Manager Private Certificate Authority.	26. März 2020
Die Vorlage ARN wurde zu den Prüfberichten hinzugefügt.	Weitere Informationen finden Sie unter Erstellen eines Auditberichts für Ihre private CA.	6. März 2020
CloudFormation Unterstützung	Support hinzugefügt für CloudFormation. Weitere Informationen finden Sie unter ACMPCA-Ressourcentypenreferenz im CloudFormation -Benutzerhandbuch.	22. Januar 2020

CloudWatch Integration von Veranstaltungen	Integration mit CloudWatch Ereignissen für asynchrone Ereignisse, einschließlich der Erstellung von Zertifizierungsstellen, der Ausstellung von Zertifikaten und der CRL-Erstellung. Weitere Informationen finden Sie unter Ereignisse verwenden. CloudWatch	23. Dezember 2019
FIPS-Endpunkte	FIPS-Endpunkte für AWS GovCloud (US-Ost) und (US-West) hinzugefügt. AWS GovCloud Eine vollständige Liste der AWS Private CA Endpunkte finden Sie unter AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	13. Dezember 2019
Tag-basierte Berechtigungen	Tag-basierte Berechtigungen werden mit dem neuen APIs `TagResource` ,`UntagResource` , und unterstützt. `ListTagsForResource` Allgemeine Informationen zur Tag-basierten Steuerung finden Sie unter Steuerung des Zugriffs auf und für IAM-Benutzer und -Rollen mithilfe von IAM-Ressourcen-Tags.	5. November 2019

Durchsetzung von Namensbeschränkungen	Unterstützung für das Erzwingen von Beschränkungen für den Subjektnamen für importierte CA-Zertifikate wurde hinzugefügt. Weitere Informationen finden Sie unter Erzwingen von Namenseinschränkungen in einer privaten CA.	28. Oktober 2019
Neue Zertifikatsvorlagen	Neue Zertifikatsvorlagen hinzugefügt, einschließlich Vorlagen für die Codesignatur mit AWS Signer. Weitere Informationen finden Sie unter Verwenden von Vorlagen.	1. Oktober 2019
Planung Ihrer Zertifizierungsstelle	Neuer Abschnitt zur Planung Ihrer PKI mithilfe von AWS Private CA hinzugefügt. Weitere Informationen finden Sie unter Planen der Bereitstellung der ACM Private CA.	30. September 2019
Zusätzlicher Support für Regionen	Regionsunterstützung für die Region AWS Asien-Pazifik (Hongkong) hinzugefügt. Eine vollständige Liste der unterstützten Regionen finden Sie unter AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	24. Juli 2019

Vollständige Unterstützung für private CA-Hierarchien hinzugefügt	Durch die Support für das Erstellen und Hosten von CAs Root ist kein externes übergeordnetes Element erforderlich.	20. Juni 2019
Zusätzlicher Support für Regionen	Regionsunterstützung für die Regionen AWS GovCloud (US-West und US-Ost) hinzugefügt. Eine vollständ ige Liste der unterstützten Regionen finden Sie unter AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	8. Mai 2019
Zusätzlicher Support für Regionen	Regionsunterstützung für die Regionen AWS Asien-Paz ifik (Mumbai und Seoul), USA West (Nordkalifornien) und EU (Paris und Stockholm) hinzugefügt. Eine vollständ ige Liste der unterstützten Regionen finden Sie unter AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	4. April 2019
Testen des Workflows zur Zertifikatserneuerung	Kunden können nun manuell die Konfiguration ihres von ACM verwaltete Erneuerun gsworkflow testen. Weitere Informationen finden Sie unter Testen der Konfiguration von ACM verwalteter Erneuerun gen.	14. März 2019

Zusätzlicher Support für Regionen	Unterstützung für Regionen für die Region AWS EU (London) hinzugefügt. Eine vollständige Liste der unterstützten Regionen finden Sie unter AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	1. August 2018
Wiederherstellung gelöscht CAs	Mit Private CA Restore können Kunden Zertifizierungsstellen (CAs) bis zu 30 Tage lang wiederherstellen, nachdem sie gelöscht wurden. Weitere Informationen finden Sie unter Wiederherstellen Ihrer privaten Zertifizierungsstelle.	20. Juni 2018

Frühere Aktualisierungen

In der folgenden Tabelle wird der Versionsverlauf der Dokumentation AWS Private Certificate Authority vor Juni 2018 beschrieben.

Änderungen	Beschreibung	Date
Neues Handbuch	Mit dieser Version wird AWS Private Certificate Authority eingeführt.	04. April 2018

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.