



User Guide

AWS Private Certificate Authority



Version latest

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

AWS Private Certificate Authority: User Guide

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

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Marken, die nicht im Besitz von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist AWS Private CA?	1
Was ist der beste Zertifikatservice für meine Anforderungen?	1
Regionen	2
Integrierte Services	3
Unterstützte Algorithmen	3
Kontingente	4
RFC-Compliance	5
Preisgestaltung	7
Sicherheit	8
IAM	9
API-Berechtigungen	10
AWS Von verwaltete Richtlinien	15
Kundenverwaltete Richtlinien	20
Eingebundene Richtlinien	21
Kontoübergreifender Zugriff	26
Ressourcenbasierte Richtlinien	27
Datenschutz	31
Speicher- und Sicherheitskonformität AWS Private CA privater Schlüssel	32
Datenverschlüsselung im AWS Private CA Connector für Active Directory	33
Compliance-Validierung	33
Erstellen eines Auditberichts	34
Sicherheit der Infrastruktur	42
VPC-Endpunkte (AWS PrivateLink)	42
Protokollierung und Überwachung	46
CloudWatch -Metriken	47
Verwenden von - CloudWatch Ereignissen	48
Verwenden von CloudTrail	55
Planung einer privaten CA	76
AWS Konto und CLI	76
Melden Sie sich für eine an AWS-Konto	77
Erstellen eines Administratorbenutzers	77
Installieren Sie das AWS Command Line Interface	78
Entwerfen einer CA-Hierarchie	79
Validierung von Endzertifikaten	81

Planung der Struktur einer CA-Hierarchie	83
Festlegung von Längenbeschränkungen für den Zertifizierungspfad	86
Verwaltung des CA-Lebenszyklus	88
Gültigkeitszeiträume wählen	88
Verwaltung der CA-Nachfolge	90
Widerrufen einer CA	92
Widerruf	92
Allgemeine Anforderungen für Sperrkonfigurationen	94
CRL-Einrichtung	95
OCSP-Anpassung	105
CA-Modi	108
GENERAL_PURPOSE (Standard)	108
SHORT_LIVED_CERTIFICATE	109
Ausfallsicherheit	109
Redundanz und Disaster Recovery	110
Bewährte Methoden	111
Dokumentieren der CA-Struktur und -Richtlinien	111
Minimieren Sie die Verwendung der Stammzertifizierungsstelle, wenn möglich	111
Geben Sie der Stammzertifizierungsstelle eine eigene AWS-Konto	112
Separate Administrator- und Ausstellerrollen	113
Implementieren des verwalteten Widerrufs von Zertifikaten	113
AWS CloudTrail aktivieren	113
Rotieren des privaten Schlüssels der Zertifizierungsstelle	113
Löschen ungenutzter CAs	114
Den öffentlichen Zugriff auf Ihre CRLs blockieren	114
Bewährte Methoden für Amazon-EKS-Anwendungen	114
CA-Verwaltung	115
Erstellen einer privaten Zertifizierungsstelle	116
Konsolenprozedur	116
CLI-Prozedur	123
Verwenden von CloudFormation	137
Installieren des CA-Zertifikats	137
Kompatible Signaturalgorithmen	138
Installieren eines Stammzertifizierungsstellenzertifikats	140
Installieren eines untergeordneten CA-Zertifikats, das von gehostet wird AWS Private CA ...	147

Installieren eines untergeordneten CA-Zertifikats, das von einer externen übergeordneten CA signiert wurde	149
Steuern des Zugriffs	150
Erstellen von Einzelkontoberechtigungen für einen IAM-Benutzer	150
Anfügen einer Richtlinie für den kontoübergreifenden Zugriff	154
Auflisten privater CAs	156
Anzeigen einer Zertifizierungsstelle	158
Hinzufügen von Tags	162
Aktualisieren einer Zertifizierungsstelle	164
Aktualisieren des CA-Status	164
Aktualisieren einer Zertifizierungsstelle (Konsole)	168
Aktualisieren einer CA (CLI)	172
Löschen einer Zertifizierungsstelle	180
Wiederherstellen einer Zertifizierungsstelle	182
Wiederherstellen einer privaten Zertifizierungsstelle (Konsole)	182
Wiederherstellen einer privaten Zertifizierungsstelle (AWS CLI)	183
Verwaltung von Zertifikaten	185
Ausstellen privater Endentitätszertifikate	185
Ausstellen eines Standardzertifikats (AWS CLI)	187
Ein Zertifikat mit einem benutzerdefinierten Betreffnamen mithilfe einer APIPassthrough-Vorlage ausstellen	189
Ausstellen eines Zertifikats mit benutzerdefinierten Erweiterungen mithilfe einer APIPassthrough-Vorlage	192
Abrufen eines privaten Zertifikats	193
Auflisten privater Zertifikate	194
Exportieren eines Zertifikats	199
Widerrufen eines privaten Zertifikats	200
Widerrufene Zertifikate und OCSP	201
Widerrufene Zertifikate in einer Zertifikatssperreliste	201
Widerrufene Zertifikate in einem Auditbericht	202
Automatisieren des Exports	203
Zertifikatvorlagen	204
Vorlagenvarianten	205
Reihenfolge der Operationen in Vorlagen	216
Vorlagendefinitionen	217
Verwenden der API (Java-Beispiele)	262

Programmgesteuertes Erstellen und Aktivieren einer Root-CA	263
Programmgesteuertes Erstellen und Aktivieren einer untergeordneten Zertifizierungsstelle	272
CreateCertificateAuthority	281
Verwenden von CreateCertificateAuthority zur Unterstützung von Active Directory	285
CreateCertificateAuthorityAuditReport	294
CreatePermission	296
DeleteCertificateAuthority	299
DeletePermission	301
DeletePolicy	303
DescribeCertificateAuthority	306
DescribeCertificateAuthorityAuditReport	308
GetCertificate	311
GetCertificateAuthorityCertificate	313
GetCertificateAuthorityCsr	316
GetPolicy	318
ImportCertificateAuthorityCertificate	320
IssueCertificate	323
ListCertificateAuthorities	326
ListPermissions	331
ListTags	333
PutPolicy	335
RestoreCertificateAuthority	337
RevokeCertificate	339
TagCertificateAuthorities	341
UntagCertificateAuthority	344
UpdateCertificateAuthority	346
Erstellen von Zertifizierungsstellen und Zertifikaten mit benutzerdefinierten Betreffnamen	348
CA erstellen mit CustomAttribute	349
Zertifikat ausstellen mit CustomAttribute	353
Erstellen Sie Zertifikate mit benutzerdefinierten Erweiterungen	356
Aktivieren Sie eine untergeordnete CA mit dem NameConstraints Ausdehnung	357
Ausstellen eines Zertifikats mit der Erweiterung der QC-Anweisung	367
Implementieren von Boler (Java-Beispiele)	372
Aktivieren einer Product Attestation Authority (PAA)	373
Aktivieren eines Product Attestation Intermediate (PAI)	383
Erstellen eines Gerätebescheinigungszertifikats (DAC)	394

Aktivieren Sie eine Root CA für Node Operational Certificates (NOC)	398
Aktivieren einer untergeordneten Zertifizierungsstelle für Betriebszertifikate für Knoten (NOC) .	408
Erstellen eines Node Operational Certificate (NOC)	418
Implementierung von mDL (Java-Beispiele)	424
Aktivieren eines IAM-Zertifikats (ausstellende Zertifizierungsstelle)	424
Erstellen eines Dokumentsigniererzertifikats	434
Verwenden einer externen Zertifizierungsstelle	439
Sichern von Kubernetes	443
Kontoubergreifende Verwendung des cert-manager	445
Unterstützte Zertifikatsvorlagen	446
Beispiellösungen	446
Connector für AD	33
Was ist Connector für AD?	447
Sind Sie ein erstmaliger Konnektor für AD-Benutzer?	447
Zugreifen auf Connector für AD	447
Preise für Connector für AD	448
Erste Schritte	448
Voraussetzungen	448
Erstellen eines Konnektors	456
Konfigurieren von AD	456
Erstellen einer Vorlage	458
Verwalten von AD-Gruppenberechtigungen	458
Verfahren	458
Konnektor erstellen	459
Vorlage erstellen	462
Verbinder auflisten	470
Vorlagen auflisten	471
Konnektor anzeigen	472
Vorlage anzeigen	473
Registrierung des Verzeichnisses	475
Gruppen und Berechtigungen	477
Service-Prinzipalname	478
Tags (Markierungen)	479
Fehlerbehebung	481
Signieren einer CSR	481
Latenz bei OCSP-Antworten	481

Amazon S3 blockiert den CRL-Bucket	482
Widerrufen eines selbstsignierten CA-Zertifikats	482
Umgang mit Ausnahmen	482
Verwenden Sie den Matter-Standard	486
Konnektor für AD-Fehler und -Ausfälle	489
Fehler	489
Fehler bei der Erstellung des Connectors	495
Fehler bei der SPN-Erstellung	499
Fehler bei der Erstellung des Connectors für AD-Connector	495
Begriffe und Konzepte	502
Vertrauensstellung	502
TLS-Serverzertifikate	502
Zertifikatsignatur	503
Zertifizierungsstelle	503
Stammzertifizierungsstelle	503
CA-Zertifikat	504
CA-Stammzertifikat	505
Endentitätszertifikat	505
Selbstsignierte Zertifikate	506
Privates Zertifikat	506
Zertifikatpfad	507
Einschränkung der Pfadlänge	507
Dokumentverlauf	509
Frühere Aktualisierungen	517
.....	dxviii

Was ist AWS Private CA?

AWS Private CA ermöglicht die Erstellung von privaten Zertifizierungsstellen (Certificate Authority, CA)-Hierarchien, einschließlich Stamm- und untergeordneter Zertifizierungsstellen – ohne die Investitions- und Wartungskosten, die beim Betrieb einer On-Premise-Zertifizierungsstelle anfallen. Ihre privaten Zertifizierungsstellen können X.509-Endentitätszertifikate ausstellen, die 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

Sie können über die AWS Management Console-Konsole auf AWS Private CA-Operationen zugreifen, indem Sie die AWS Private CA-API oder die AWS CLI verwenden.

Themen

- [Was ist der beste Zertifikatservice für meine Anforderungen?](#)
- [Regionen](#)
- [In AWS Private Certificate Authority integrierte Services](#)
- [Unterstützte kryptografische Algorithmen](#)
- [Kontingente](#)
- [RFC-Compliance](#)
- [Preisgestaltung](#)

Was ist der beste Zertifikatservice für meine Anforderungen?

Es gibt zwei AWS-Services zum Ausstellen und Bereitstellen von X.509-Zertifikaten. Wählen Sie den Service, der Ihren Bedürfnissen am besten entspricht. Dabei sollten Sie überlegen, was Sie benötigen: öffentliche oder private Zertifikate, angepasste Zertifikate, Zertifikate, die Sie in anderen AWS-Services bereitstellen möchten, oder eine automatisierte Zertifikatsverwaltung und -verlängerung.

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 eine eigene CA-Hierarchie erstellen und Zertifikate für die Authentifizierung von internen Benutzern, Computern, Anwendungen, Services, Servern und anderen Geräten sowie für die Computercode-Signatur ausstellen. 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 Zertifikatthemenamen 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 Service verwaltet Zertifikate für Unternehmenskunden, die mithilfe von TLS eine öffentlich vertrauenswürdige sichere Webpräsenz 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 verwenden, AWS Private CA um eine Zertifizierungsstelle zu erstellen, kann ACM die Ausstellung von Zertifikaten von dieser privaten Zertifizierungsstelle verwalten und Zertifikatserneuerungen automatisieren.

Weitere Informationen finden Sie im [AWS Certificate Manager-Benutzerhandbuch](#).

Regionen

Wie bei den meisten AWS-Ressourcen, handelt es sich bei privaten Zertifizierungsstellen (CAs) um regionale Ressourcen. Wenn Sie private CAs in mehreren Regionen verwenden möchten, müssen

Sie die CAs in diesen Regionen erstellen. Sie können keine privaten CAs zwischen Regionen hin und her kopieren. Besuchen Sie [AWS Regionen und -Endpunkte](#) in der Allgemeine AWS-Referenz oder die [Tabelle 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.

In AWS Private Certificate Authority integrierte Services

Wenn Sie verwenden, AWS Certificate Manager um ein privates Zertifikat anzufordern, können Sie dieses Zertifikat jedem -Service zuordnen, der in ACM integriert ist. Dies gilt sowohl für Zertifikate, die mit einem AWS Private CA-Stamm verkettet sind, als auch für solche, die mit einem externen Stamm verkettet sind. Weitere Informationen finden Sie unter [Integrierte Services](#) im AWS Certificate Manager -Benutzerhandbuch.

Sie können private CAs auch in Amazon Elastic Kubernetes Service integrieren, um die Ausstellung von Zertifikaten innerhalb eines Kubernetes-Clusters zu ermöglichen. Weitere Informationen finden Sie unter [Sichern von Kubernetes mit AWS Private CA](#).

 Note

Amazon Elastic Kubernetes Service ist kein in ACM integrierter Service.

Wenn Sie die AWS Private CA -API oder verwenden AWS CLI, um ein Zertifikat auszustellen oder ein privates Zertifikat aus ACM zu exportieren, können Sie das Zertifikat überall installieren, wo Sie möchten.

Unterstützte kryptografische Algorithmen

AWS Private CA unterstützt die folgenden kryptografischen Algorithmen für die Generierung privater Schlüssel und die Zertifikatsignierung.

Unterstützter Algorithmus

Algorithmen für private Schlüssel	Signaturalgorithmen
RSA_2048	SHA256WITHECDSA
RSA_4096	SHA384WITHECDSA
EC_prime256v1	SHA512WITHECDSA
EC_secp384r1	SHA256WITHRSA SHA384WITHRSA SHA512WITHRSA

Diese Liste gilt nur für Zertifikate, die direkt von AWS Private CA über die Konsole, API oder Befehlszeile ausgestellt wurden. Wenn Zertifikate mit einer Zertifizierungsstelle von aus aus AWS Certificate Manager ausstellt AWS Private CA, unterstützt es einige, aber nicht alle dieser Algorithmen. Weitere Informationen finden Sie unter [Anfordern eines privaten Zertifikats](#) im AWS Certificate Manager-Benutzerhandbuch.

Note

In allen Fällen muss die angegebene Signaturalgorithmusfamilie (RSA oder ECDSA) mit der Algorithmusfamilie des privaten Schlüssels der Zertifizierungsstelle übereinstimmen.

Kontingente

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 spezifisch für ein AWS Konto und eine Region.

AWS Private CA drosselt API-Anfragen je nach API-Vorgang mit unterschiedlichen Raten. Ablehnung bedeutet, dass AWS Private CA eine ansonsten gültige Anfrage abweist, weil mit der Anfrage das zulässige Kontingent der Operation für die Anzahl von Anfragen pro Sekunde überschritten wird. Wenn eine Anforderung gedrosselt wird, gibt einen [ThrottlingException](#) Fehler AWS Private CA zurück. garantiert AWS Private CA keine Mindestanforderungsrate für APIs .

Informationen dazu, welche Kontingente angepasst werden können, finden Sie in der [AWS Private CA Tabelle der Kontingente](#) im Allgemeine AWS-Referenz.

Sie können Ihre aktuellen Kontingente anzeigen und Kontingenterhöhungen mit AWS Service Quotas beantragen.

So zeigen Sie eine up-to-date Liste Ihrer AWS Private CA Kontingente an

1. Melden Sie sich bei Ihrem -AWS-Konto an.
2. Öffnen Sie die Service-Quotas-Konsole unter <https://console.aws.amazon.com/servicequotas/>.
3. Wählen Sie in der Liste Services die Option AWS Certificate Manager Private Certificate Authority (ACM PCA) aus. Jedes Kontingent in der Liste Service Quotas zeigt Ihren aktuell angewendeten Kontingentwert, den Standardkontingentwert und ob das Kontingent anpassbar ist oder nicht. Wählen Sie den Namen eines Kontingents aus, um weitere Informationen darüber zu erhalten.

So fordern Sie eine Kontingenterhöhung an

1. Wählen Sie in der Liste Service Quotas das Optionsfeld für ein anpassbares Kontingent aus.
2. Wählen Sie die Schaltfläche Kontingenterhöhung anfordern.
3. Füllen Sie das Formular zur Erhöhung des Kontingents aus und senden Sie es ab.

AWS Private CA ist in AWS Certificate Manager integriert. 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 PCA-Kontingenten als auch den Kontingenten, die ACM auf öffentlichen und importierten Zertifikaten platziert. Weitere Informationen zu ACM-Anforderungen finden Sie unter [Anfordern eines privaten Zertifikats](#) und von [Kontingenten](#) im AWS Certificate Manager-Benutzerhandbuch.

RFC-Compliance

AWS Private CA setzt bestimmte Einschränkungen, die in [RFC 5280](#) definiert sind, nicht durch. 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.
- [Basic constraints](#) (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 zu dem Flag `critical` muss `CA=true` festgelegt werden. AWS Private CA erzwingt grundlegende Einschränkungen durch einen Fehler mit einer Validierungsausnahme aus den folgenden Gründen:

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

Die Pfadlänge ([pathLenConstraint](#)) bestimmt, wie viele untergeordnete CAs nach dem importierten Zertifizierungsstellenzertifikat vorhanden sein können. AWS Private CA erzwingt die Pfadlänge, indem es aus 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.

Nicht erzwungen

- [Policy constraints](#) (Richtlinieneinschränkungen). Diese Einschränkungen begrenzen die Kapazität einer Zertifizierungsstelle zum Ausstellen untergeordneter CA-Zertifikate.
- [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 dem 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.
- [SubjectPublicKeyInfo](#) und [Subject Alternative Name \(SAN\)](#). Bei der Ausstellung eines Zertifikats kopiert AWS Private CA die SAN-Erweiterungen `SubjectPublicKeyInfo` und aus der bereitgestellten CSR, ohne eine Validierung durchzuführen.

Preisgestaltung

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 umfasst 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 Services](#) wie Elastic Load Balancing CloudFront und API Gateway verwendet werden.

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

Sicherheit in AWS Private Certificate Authority

Cloud-Sicherheit bei AWS hat höchste Priorität. Als - AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die entwickelt wurden, um die Anforderungen der sicherheitskritischsten Organisationen zu erfüllen.

Sicherheit ist eine geteilte Verantwortung zwischen AWS und Ihnen. Das [Modell der geteilten Verantwortung](#) beschreibt dies als Sicherheit der Cloud selbst und Sicherheit in der Cloud:

- Sicherheit der Cloud – AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Services in der AWS Cloud ausführt. AWS stellt Ihnen außerdem Services bereit, die Sie sicher nutzen können. Externe Prüfer testen und überprüfen im Rahmen der [AWS Compliance-Programme](#) regelmäßig die Wirksamkeit unserer Sicherheit. Informationen zu den Compliance-Programmen, die für gelten AWS Private Certificate Authority, finden Sie unter [AWS-Services im Geltungsbereich nach Compliance-Programm](#) .
- Sicherheit in der Cloud – Ihre Verantwortung wird durch den - AWS Service bestimmt, den Sie verwenden. 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 geteilten Verantwortung bei der Verwendung von einsetzen können AWS Private CA. Die folgenden Themen veranschaulichen, wie Sie konfigurieren, AWS Private CA um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere verwenden AWS-Services , die Ihnen bei der Überwachung und Sicherung Ihrer - AWS Private CA Ressourcen helfen.

Themen

- [Identity and Access Management \(IAM\) für AWS Private Certificate Authority](#)
- [Bewährte Methoden für die Sicherheit für den kontoübergreifenden Zugriff auf private CAs](#)
- [Datenschutz in AWS Private Certificate Authority](#)
- [Compliance-Validierung für AWS Private Certificate Authority](#)
- [Infrastruktursicherheit in AWS Private Certificate Authority](#)
- [Protokollierung und Überwachung in AWS Private Certificate Authority](#)

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

Für den Zugriff auf sind Anmeldeinformationen AWS Private CA erforderlich, die zur Authentifizierung Ihrer Anforderungen verwenden AWS kann. In den folgenden Themen erfahren Sie, wie Sie Ihre privaten Zertifizierungsstellen (CAs) mithilfe von [AWS Identity and Access Management \(IAM\)](#) sichern können, indem Sie den Zugriff darauf steuern.

Ein ist die primäre Ressource AWS Private CA, 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 Prinzipal-Entität des AWS Kontos, in dem eine - AWS Ressource erstellt wird. Die Funktionsweise wird anhand der folgenden Beispiele deutlich.

- Wenn Sie die Anmeldeinformationen Ihres 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 zu verwenden, um CAs zu erstellen.
 - Wir empfehlen dringend die Verwendung der Multi-Faktor-Authentifizierung (MFA), wenn Sie auf zugreifen AWS Private CA.
- Wenn Sie einen IAM-Benutzer in Ihrem AWS Konto erstellen, können Sie diesem Benutzer die Berechtigung zum Erstellen einer privaten Zertifizierungsstelle erteilen. Jedoch besitzt das Konto, zu dem dieser Benutzer gehört, die Zertifizierungsstelle.
 - Wenn Sie eine IAM-Rolle in Ihrem AWS Konto erstellen und ihr die Berechtigung zum Erstellen einer privaten Zertifizierungsstelle erteilen, kann jeder, der die Rolle übernehmen kann, die Zertifizierungsstelle 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

Diese Dokumentation behandelt die Verwendung von IAM im Kontext von 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 einer IAM-Identität anfügen 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 Zertifizierungsstelle mit Tags erstellt wird.)	arn:aws:acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ <i>11223344-1234-1122-2233-112233445566</i>
CreateCertificateAuthorityAuditReport	acm-pca:CreateCertificateAuthorityAuditReport	arn:aws:acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ <i>11223344-1234-1122-2233-112233445566</i>

AWS Private CA API-Operationen	Erforderliche Berechtigungen	Ressourcen
CreatePermission	acm-pca:CreatePermission	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
DeleteCertificateAuthority	acm-pca:DeleteCertificateAuthority	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
DeletePermission	acm-pca:DeletePermission	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
DeletePolicy	acm-pca:DeletePolicy	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
DescribeCertificateAuthority	acm-pca:DescribeCertificateAuthority	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566

AWS Private CA API-Operationen	Erforderliche Berechtigungen	Ressourcen
DescribeCertificateAuthorityAuditReport	acm-pca:DescribeCertificateAuthorityAuditReport	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
GetCertificate	acm-pca:GetCertificate	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
GetCertificateAuthorityCertificate	acm-pca:GetCertificateAuthorityCertificate	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
GetCertificateAuthorityCsr	acm-pca:GetCertificateAuthorityCsr	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
GetPolicy	acm-pca:GetPolicy	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566

AWS Private CA API-Operationen	Erforderliche Berechtigungen	Ressourcen
ImportCertificateAuthorityCertificate	acm-pca:ImportCertificateAuthorityCertificate	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
IssueCertificate	acm-pca:IssueCertificate	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
ListCertificateAuthorities	acm-pca:ListCertificateAuthorities	N/A
ListPermissions	acm-pca:ListPermissions	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
ListTags	acm-pca:ListTags	N/A
PutPolicy	acm-pca:PutPolicy	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566

AWS Private CA API-Operationen	Erforderliche Berechtigungen	Ressourcen
RevokeCertificate	acm-pca:RevokeCertificate	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
TagCertificateAuthority	acm-pca:TagCertificateAuthority	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
UntagCertificateAuthority	acm-pca:UntagCertificateAuthority	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :111122223333 :certificate-authority/ 11223344-1234-1122-2233-112233445566
UpdateCertificateAuthority	acm-pca:UpdateCertificateAuthority	arn: <i>aws</i> :acm-pca: <i>us-east-1</i> :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 Anweisungen unter [Erstellen einer Rolle für einen externen Identitätsanbieter \(Verbund\)](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter [Erstellen einer Rolle für einen IAM-Benutzer](#) 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 Von verwaltete Richtlinien

AWS Private CA enthält 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 Beispielrichtliniencode anzuzeigen.

AWSPriateCAFullAccess

Gewährt uneingeschränkte administrative Kontrolle.

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

AWSPriateCARedOnly

Gewährt Zugriff, der auf schreibgeschützte API-Operationen beschränkt ist.

```
{
  "Version":"2012-10-17",
  "Statement":{
    "Effect":"Allow",
    "Action":[
      "acm-pca:DescribeCertificateAuthority",
      "acm-pca:DescribeCertificateAuthorityAuditReport",
      "acm-pca:ListCertificateAuthorities",
      "acm-pca:GetCertificateAuthorityCsr",
      "acm-pca:GetCertificateAuthorityCertificate",
      "acm-pca:GetCertificate",
      "acm-pca:GetPolicy",
      "acm-pca:ListPermissions",
      "acm-pca:ListTags"
    ],
    "Resource":"*"
  }
}
```

AWSPriateCAPrivilegedUser

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.

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



```

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

```

AWSPriateCAUser

Erteilen 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 PrivilegedUser Richtlinie gegenseitig aus.

```

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

```

```

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

```

```
}
```

AWSPriateCAAuditor

Gewähren Sie Zugriff auf schreibgeschützte API-Operationen und die Berechtigung zum Generieren eines CA-Auditberichts.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "acm-pca:CreateCertificateAuthorityAuditReport",
        "acm-pca:DescribeCertificateAuthority",
        "acm-pca:DescribeCertificateAuthorityAuditReport",
        "acm-pca:GetCertificateAuthorityCsr",
        "acm-pca:GetCertificateAuthorityCertificate",
        "acm-pca:GetCertificate",
        "acm-pca:GetPolicy",
        "acm-pca:ListPermissions",
        "acm-pca:ListTags"
      ],
      "Resource":"arn:aws:acm-pca:*:*:certificate-authority/*"
    },
    {
      "Effect":"Allow",
      "Action":[
        "acm-pca:ListCertificateAuthorities"
      ],
      "Resource":""
    }
  ]
}
```

Aktualisierungen der von AWS verwalteten Richtlinien für AWS Private CA

Zeigen Sie in der folgenden Tabelle Details zu Aktualisierungen der von AWS verwalteten Richtlinien für an, AWS Private CA seit der Service mit der Verfolgung dieser Änderungen begonnen hat. Um automatische Warnungen über alle Änderungen an zu erhalten AWS Private CA, abonnieren Sie den RSS-Feed auf der [Dokumentverlauf](#) Seite.

Änderungen an verwalteten Richtlinien

Änderung	Beschreibung	Datum
Neue Richtliniennamen: <ul style="list-style-type: none"> • <code>AWSPprivateCAFullAccess</code> • <code>AWSPprivateCAReadOnly</code> • <code>AWSPprivateCAPrivilegedUser</code> • <code>AWSPprivateCAAuditor</code> • <code>AWSPprivateCAUser</code> 	Richtliniennamenpräfixe wurden von <code>AWSCertificateManagerPrivateCA</code> in geändert <code>AWSPprivateCA</code> . Die Funktionalität bleibt unverändert.	13. Februar 2023

Kundenverwaltete Richtlinien

Als bewährte Methode verwenden Sie nicht für Root-Benutzer des AWS-Kontos die Interaktion mit AWS, einschließlich 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 schränkt Benutzer ein, nur die von Ihnen angegebenen AWS Private CA Aktionen auszuführen.

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.

```
{
  "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 anfügen, um einzuschränken, welche AWS Private CA Aktionen der Benutzer ausführen kann. Informationen zum Anwenden 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 zum Ausführen von AWS Private CA Aktionen zuweisen. Allgemeine Informationen zu eingebundenen Richtlinien finden Sie unter [Arbeiten mit eingebundenen Richtlinien](#) im [IAM-Benutzerhandbuch](#). Sie können die AWS Management Console, die AWS Command Line Interface (AWS CLI) oder die IAM-API verwenden, um eingebundene Richtlinien zu erstellen und einzubetten.

Important

Wir empfehlen dringend die Verwendung der Multi-Faktor-Authentifizierung (MFA), wenn Sie auf zugreifen AWS Private CA.

Themen

- [Auflisten privater CAs](#)
- [Abrufen eines privaten CA-Zertifikats](#)
- [Importieren eines privaten CA-Zertifikats](#)
- [Löschen einer privaten Zertifizierungsstelle](#)
- [T ag-on-create: Anhängen von Tags an eine Zertifizierungsstelle zum Zeitpunkt der Erstellung](#)
- [T ag-on-create: Eingeschränktes Tagging](#)
- [Steuern des Zugriffs auf Private CA mithilfe von Tags](#)
- [Schreibgeschützter Zugriff auf AWS Private CA](#)
- [Vollzugriff auf AWS Private CA](#)
- [Administratorzugriff auf alle AWS -Ressourcen](#)

Auflisten privater CAs

Mit der folgenden Richtlinie können Benutzer alle privaten Zertifizierungsstellen in einem Konto auflisten.

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

Abrufen eines privaten CA-Zertifikats

Mit der folgenden Richtlinie können Benutzer ein bestimmtes privates Zertifizierungsstellenzertifikat abrufen.

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

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

Importieren eines privaten CA-Zertifikats

Mit der folgenden Richtlinie kann ein Benutzer ein privates Zertifizierungsstellenzertifikat importieren.

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

Löschen einer privaten Zertifizierungsstelle

Mit der folgenden Richtlinie können Benutzer eine bestimmte private Zertifizierungsstelle löschen.

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

Tag-on-create: Anhängen von Tags an eine Zertifizierungsstelle zum Zeitpunkt der Erstellung

Die folgende Richtlinie erlaubt es einem Benutzer, Tags während der CA-Erstellung anzuwenden.

```
{
```

```

"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-Paares Environment=Prod während der CA-Erstellung. Das Markieren mit anderen Schlüssel-Wert-Paaren ist zulässig.

```

{
  "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 gewährt nur Zugriff auf CAs mit dem Schlüssel-Wert-Paar `Environment=PreProd`. Sie erfordert auch, dass neue CAs dieses Tag enthalten.

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

Schreibgeschützter 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.

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

Vollzugriff auf AWS Private CA

Mit der folgenden Richtlinie können Benutzer beliebige AWS Private CA Aktionen ausführen.

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

Administratorzugriff auf alle AWS -Ressourcen

Mit der folgenden Richtlinie kann ein Benutzer jede Aktion für jede AWS Ressource ausführen.

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

Bewährte Methoden für die Sicherheit für den kontoübergreifenden Zugriff auf private CAs

Ein AWS Private CA Administrator kann eine Zertifizierungsstelle für Prinzipale (Benutzer, Rollen usw.) in einem anderen AWS Konto freigeben. Wenn eine Freigabe empfangen und akzeptiert wurde,

kann der Prinzipal die CA verwenden, um Endentitätszertifikate mithilfe von - AWS Private CA oder - AWS Certificate Manager Ressourcen auszustellen. Der Prinzipal kann die CA verwenden, um untergeordnete CA-Zertifikate mit auszustellen AWS Private CA.

Important

Gebühren im Zusammenhang mit einem Zertifikat, das in einem kontoübergreifenden Szenario ausgestellt wurde, werden dem AWS Konto in Rechnung gestellt, das das Zertifikat ausstellt.

Um den Zugriff auf eine Zertifizierungsstelle freizugeben, können AWS Private CA Administratoren eine der folgenden Methoden wählen:

- Verwenden Sie AWS Resource Access Manager (RAM), um die CA als Ressource für einen Prinzipal in einem anderen Konto oder für freizugeben AWS Organizations. RAM ist eine Standardmethode für die gemeinsame Nutzung von AWS Ressourcen über -Konten hinweg. 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 Zertifizierungsstelle anzufügen und so Zugriff auf einen Prinzipal in einem anderen Konto zu gewähren. Weitere Informationen finden Sie unter [Ressourcenbasierte Richtlinien](#).

Der [Steuern des Zugriffs auf eine private Zertifizierungsstelle](#) Abschnitt dieses Handbuchs enthält Workflows für die Gewährung des Zugriffs auf CAs in konto- und kontoübergreifenden Szenarien.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind Berechtigungsrichtlinien, die Sie erstellen und manuell an eine Ressource (in diesem Fall eine private Zertifizierungsstelle) und nicht an eine Benutzeridentität oder Rolle anfügen. Anstatt Ihre eigenen Richtlinien zu erstellen, können Sie auch AWS verwaltete Richtlinien für verwenden AWS Private CA. Mit AWS RAM zur Anwendung einer ressourcenbasierten Richtlinie kann ein AWS Private CA Administrator den Zugriff auf eine CA direkt oder über für einen Benutzer in einem anderen AWS Konto freigeben AWS Organizations. Alternativ kann ein AWS Private CA Administrator die PCA-APIs [PutPolicy](#), [GetPolicy](#) und [DeletePolicy](#) oder die entsprechenden AWS CLI Befehle [put-policy](#), [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 zur [Bibliothek Verwaltete Berechtigungen](#) in der - AWS Resource Access Manager Konsole und suchen Sie nach CertificateAuthority. Wie bei jeder Richtlinie empfehlen wir, die Richtlinie vor der Anwendung in einer Testumgebung anzuwenden, um sicherzustellen, dass sie Ihren Anforderungen entspricht.

AWS Certificate Manager (ACM)-Benutzer mit kontoübergreifendem 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 Endentitätszertifikate:

- [EndEntityCertificate/V1](#)
- [EndEntityClientAuthCertificate/V1](#)
- [EndEntityServerAuthCertificate/V1](#)
- [BlankEndEntityCertificate_APIPassthrough /V1](#)
- [BlankEndEntityCertificate_APICSRPassthrough /V1](#)
- [SubordinateCACertificate_PathLen0/V1](#)

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 Zertifizierungsstelle stellt der Administrator eine AWS Konto-ID oder eine AWS Organizations ID bereit, die Zugriff auf die Zertifizierungsstelle erhält. Der JSON-Code jeder der folgenden Richtlinien ist zur besseren Lesbarkeit als Datei formatiert, kann aber auch als Inline-CLI-Argumente bereitgestellt werden.

Note

Die Struktur der unten gezeigten ressourcenbasierten JSON-Richtlinien muss genau eingehalten werden. Nur die ID-Felder für die Prinzipale (die AWS Kontonummer oder die AWS Organizations-ID) und die CA-ARNs können von Kunden konfiguriert werden.

1. Datei: policy1.json – Freigeben des Zugriffs auf eine CA für einen Benutzer in einem anderen Konto

Ersetzen Sie **55555555555** durch die AWS Konto-ID, die die CA gemeinsam verwendet.

Ersetzen Sie für den Ressourcen-ARN Folgendes durch Ihre eigenen Werte:

- **aws** – Die AWS Partition. Zum Beispiel `aws`, `aws-us-gov`, `aws-cnusw`.
- **us-east-1** – Die AWS Region, in der die Ressource verfügbar ist, z. B. `us-west-1`.
- **111122223333** – Die AWS Konto-ID des Ressourcenbesitzers.
- **11223344-1234-1122-2233-112233445566** – Die Ressourcen-ID der Zertifizierungsstelle.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExampleStatementID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "55555555555"
      },
      "Action": [
        "acm-pca:DescribeCertificateAuthority",
        "acm-pca:GetCertificate",
        "acm-pca:GetCertificateAuthorityCertificate",
        "acm-pca:ListPermissions",
        "acm-pca:ListTags"
      ],
      "Resource": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
```

```

    },
    {
      "Sid": "ExampleStatementID2",
      "Effect": "Allow",
      "Principal": {
        "AWS": "555555555555"
      },
      "Action": [
        "acm-pca:IssueCertificate"
      ],
      "Resource": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "Condition": {
        "StringEquals": {
          "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
EndEntityCertificate/V1"
        }
      }
    }
  ]
}

```

2. Datei: policy2.json – Freigeben des Zugriffs auf eine CA über AWS Organizations

Ersetzen Sie *o-a1b2c3d4z5* durch die AWS Organizations ID.

Ersetzen Sie für den Ressourcen-ARN Folgendes durch Ihre eigenen Werte:

- *aws* – Die AWS Partition. Zum Beispiel *aws*, *aws-us-gov*, *aws-cn*sw.
- *us-east-1* – Die AWS Region, in der die Ressource verfügbar ist, z. B. *us-west-1*.
- *111122223333* – Die AWS Konto-ID des Ressourcenbesitzers.
- *11223344-1234-1122-2233-112233445566* – Die Ressourcen-ID der Zertifizierungsstelle.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExampleStatementID3",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "acm-pca:IssueCertificate",

```

```

    "Resource": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
    "Condition": {
      "StringEquals": {
        "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
EndEntityCertificate/V1",
        "aws:PrincipalOrgID": "o-a1b2c3d4z5"
      },
      "StringNotEquals": {
        "aws:PrincipalAccount": "111122223333"
      }
    }
  },
  {
    "Sid": "ExampleStatementID4",
    "Effect": "Allow",
    "Principal": "*",
    "Action": [
      "acm-pca:DescribeCertificateAuthority",
      "acm-pca:GetCertificate",
      "acm-pca:GetCertificateAuthorityCertificate",
      "acm-pca:ListPermissions",
      "acm-pca:ListTags"
    ],
    "Resource": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
    "Condition": {
      "StringEquals": {
        "aws:PrincipalOrgID": "o-a1b2c3d4z5"
      },
      "StringNotEquals": {
        "aws:PrincipalAccount": "111122223333"
      }
    }
  }
]
}

```

Datenschutz in AWS Private Certificate Authority

Das AWS [Modell der geteilten Verantwortung](#) gilt für den Datenschutz in AWS Private Certificate Authority. Wie in diesem Modell beschrieben, AWS ist für den Schutz der globalen Infrastruktur verantwortlich, die alle ausführt 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 Ihnen, -Anmeldeinformationen zu schützen AWS-Konto und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einzurichten. 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).
- Verwenden Sie SSL/TLS für die Kommunikation mit - AWS Ressourcen. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API- und Benutzeraktivitätsprotokollierung mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen in AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder eine API FIPS-140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter [Federal Information Processing Standard \(FIPS\) 140-2](#).

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 AWS Private CA oder anderen AWS-Services über die Konsole, API AWS CLI oder AWS SDKs arbeiten. 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.

Speicher- und Sicherheitskonformität AWS Private CA privater Schlüssel

Die privaten Schlüssel für private CAs werden in AWS verwalteten Hardware-Sicherheitsmodulen (HSMs) gespeichert. Die HSMs entsprechen den Sicherheitsanforderungen von FIPS PUB 140-2 Level 3 für kryptografische Module.

Datenverschlüsselung im AWS Private CA Connector für Active Directory

AWS Private CA Connector for AD speichert Kundenkonfigurationsdaten zu Connectors, Vorlagen, Verzeichnisregistrierungen, Serviceprinzipalnamen und Vorlagengruppen-Zugriffskontrolleinträgen. Diese Daten werden während der Übertragung und im Ruhezustand verschlüsselt. Informationen zu Zertifikaten, die über Connector for AD ausgestellt wurden, können mithilfe der [-GetCertificate](#)Aktion in der AWS Private CA API ermittelt werden. Es werden keine Informationen über die ausgestellten Zertifikate oder über den Client oder Computer, der ein Zertifikat anfordert, von gespeichert AWS.

Compliance-Validierung für AWS Private Certificate Authority

Externe Prüfer bewerten im AWS Private Certificate Authority Rahmen verschiedener AWS - Compliance-Programme die Sicherheit und Compliance von . Hierzu zählen unter anderem SOC, PCI, FedRAMP und HIPAA.

Eine Liste der - AWS Services, die in den Geltungsbereich bestimmter Compliance-Programme fallen, finden Sie unter [AWS -Services im Geltungsbereich nach Compliance-Programm](#) Allgemeine Informationen finden Sie unter [AWS Compliance-Programme](#)

Sie können Auditberichte von Drittanbietern mit herunterladen AWS Artifact. Weitere Informationen finden Sie unter [Herunterladen von Berichten unter AWS Artifact](#) .

Ihre Compliance-Verantwortung bei der Verwendung von 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 Unterstützung der Compliance bereit:

- Für Organisationen, die ihre Amazon S3-Buckets verschlüsseln müssen, wird in den folgenden Themen beschrieben, wie die Verschlüsselung für AWS Private CA Komponenten konfiguriert wird:
 - [Verschlüsseln von Auditberichten](#)
 - [Verschlüsseln Ihrer CRLs](#)
- [Kurzanleitungen für Sicherheit und Compliance](#) – In diesen Bereitstellungsleitfäden werden Überlegungen zur Architektur erörtert und Schritte für die Bereitstellung von Sicherheits- und Compliance-orientierten Basisumgebungen in beschrieben AWS.
- [Whitepaper zur Erstellung einer Architektur mit HIPAA-konformer Sicherheit und Compliance](#) – In diesem Whitepaper wird beschrieben, wie Unternehmen mithilfe AWS von HIPAA-konforme Anwendungen erstellen können.
- [AWS Compliance-Ressourcen](#) – Diese Sammlung von Arbeitsmappen und Leitfäden könnte für Ihre Branche und Ihren Standort gelten.

- [Bewertung von Ressourcen mit Regeln](#) im -AWS Config Entwicklerhandbuch – Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- [AWS Security Hub](#) – Dieser AWS Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus in AWS , mit dem Sie Ihre Compliance mit den Sicherheitsstandards und bewährten Methoden der Branche überprüfen können.

Verwenden von Auditberichten 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 Ihrer Prüfungsberichte](#) .

Die Auditberichtsdatei hat den folgenden Pfad und Dateinamen. Der ARN für einen Amazon S3-Bucket ist der Wert für `bucket-name`. `CA_ID` ist die eindeutige Kennung einer ausstellenden CA. `UUID` ist die eindeutige Kennung eines Auditberichts.

```
bucket-name/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=#0a1b3c4d5e6f",
    "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=#0a1b3c4d5e6f",
    "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, füllt der private CA-Auditbericht das `requestedByServicePrincipal` Feld mit `ausacm.amazonaws.com`. Dies weist darauf hin, dass der AWS Certificate Manager Service die `-IssueCertificate` Aktion der AWS Private CA -API im Namen eines Kunden aufgerufen hat, um das Zertifikat zu erneuern.

Vorbereiten eines Amazon S3-Buckets für Prüfungsberichte

Um Ihre Auditberichte zu speichern, müssen Sie einen Amazon S3-Bucket vorbereiten. Weitere Informationen finden [Sie unter Wie erstelle ich einen S3-Bucket?](#)

Ihr S3-Bucket muss durch eine angehängte Berechtigungsrichtlinie gesichert sein. Autorisierte Benutzer und Service-Prinzipale benötigen die Put Berechtigung, AWS Private CA Objekte in den Bucket zu platzieren, und die Get Berechtigung, sie abzurufen. Wir empfehlen Ihnen, die unten gezeigte Richtlinie anzuwenden, die den Zugriff sowohl auf ein AWS Konto als auch auf den ARN einer privaten Zertifizierungsstelle einschränkt. Weitere Informationen finden Sie unter [Hinzufügen einer Bucket-Richtlinie mit der Amazon S3-Konsole](#).

Note

Während des Konsolenverfahrens zum Erstellen eines Auditberichts können Sie festlegen, dass einen neuen Bucket AWS Private CA erstellen und eine Standardberechtigungsrichtlinie anwenden soll. Die Standardrichtlinie wendet keine SourceArn Einschränkung auf die CA an und ist daher freizügiger als die empfohlene Richtlinie. Wenn Sie den Standard wählen, können Sie ihn jederzeit später [ändern](#).

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

```
    "StringEquals":{
      "aws:SourceAccount":"account",
      "aws:SourceArn":"arn:partition:acm-pca:region:account:certificate-
authority/CA_ID"
    }
  }
}
```

Erstellen eines Auditberichts

Sie können einen Auditbericht entweder über die Konsole oder die 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 unter <https://console.aws.amazon.com/acm-pca/home>.
2. Wählen Sie auf der Seite Private Zertifikatautoritäten Ihre private Zertifizierungsstelle aus der Liste aus.
3. Klicken Sie im Menü Aktionen auf Audit-Bericht generieren.
4. Wählen Sie unter Auditberichtsziel für Neuen S3-Bucket erstellen? die Option 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 erstellt und fügt die Standardrichtlinie an Ihren Bucket an. Wenn Sie Nein wählen, müssen Sie Ihrem Bucket eine Richtlinie anfügen, bevor Sie einen Auditbericht erstellen können. Verwenden Sie das unter beschriebene Richtlinienmuster [Vorbereiten eines Amazon S3-Buckets für Prüfungsberichte](#). Informationen zum Anfügen einer Richtlinie finden Sie unter [Hinzufügen einer Bucket-Richtlinie mit der Amazon S3-Konsole](#).

5. Wählen Sie unter Ausgabeformat die Option JSON für JavaScript Object Notation oder CSV für kommagetrennte Werte aus.
6. Wählen Sie Generate audit report (Auditbericht generieren).

So erstellen Sie einen Audit-Bericht (AWS CLI)

1. Wenn Sie noch keinen zu verwendenden S3-Bucket haben, [erstellen Sie einen](#) .

2. Fügen Sie Ihrem Bucket eine Richtlinie hinzu. Verwenden Sie das unter beschriebene Richtlinienmuster [Vorbereiten eines Amazon S3-Buckets für Prüfungsberichte](#). Weitere Informationen zum Anfügen einer Richtlinie finden Sie unter [Hinzufügen einer Bucket-Richtlinie mithilfe der Amazon 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 bucket_name \
--audit-report-response-format JSON
```

Abrufen eines Auditberichts

Um einen Auditbericht zur Überprüfung abzurufen, verwenden Sie die Amazon S3-Konsole, die API, die CLI oder das SDK. Weitere Informationen finden Sie unter [Herunterladen eines Objekts](#) im Benutzerhandbuch für Amazon Simple Storage Service.

Verschlüsseln Ihrer Prüfungsberichte

Sie können optional die Verschlüsselung für den Amazon S3-Bucket konfigurieren, der Ihre Prüfungsberichte enthält. AWS Private CA unterstützt zwei Verschlüsselungsmodi für Komponenten in S3:

- Automatische serverseitige Verschlüsselung mit von Amazon S3-managed AES-256-Schlüsseln.
- Vom Kunden verwaltete Verschlüsselung mit AWS Key Management Service und einem , das Ihren Spezifikationen AWS KMS key entspricht.

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-Konsole unter <https://console.aws.amazon.com/s3/>.
2. Wählen Sie in der Tabelle Buckets den Bucket aus, der Ihre AWS Private CA Assets enthält.
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 Amazon S3-Schlüssel (SSE-S3) aus.
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-Konsole unter <https://console.aws.amazon.com/s3/>.
2. Wählen Sie in der Tabelle Buckets den Bucket aus, der Ihre AWS Private CA Assets enthält.
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 Schlüssel (SSE-KMS) aus.
7. Wählen Sie entweder Aus Ihren AWS KMS Schlüsseln auswählen oder AWS KMS key ARN eingeben aus.
8. Wählen Sie Save Changes.
9. (Optional) Wenn Sie noch keinen -KMS-Schlüssel haben, erstellen Sie einen mit dem folgenden AWS CLI [create-key](#)-Befehl:

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

Die Ausgabe enthält die Schlüssel-ID und den Amazon-Ressourcennamen (ARN) des KMS-Schlüssels. Im Folgenden finden Sie eine Beispielausgabe:

```
{
```

```
"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 Service-Prinzipal die Berechtigung zur Verwendung des KMS-Schlüssels. Standardmäßig sind alle KMS-Schlüssel privat; nur der Ressourcenbesitzer 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 Service-Prinzipal muss sich in derselben Region befinden, in der der KMS-Schlüssel gespeichert ist.
 - a. Speichern Sie zunächst die Standardrichtlinie für Ihren KMS-Schlüssel als `policy.json` mit dem folgenden [get-key-policy](#) Befehl:

```
$ 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 Richtlinienanweisungen aus und fügen Sie sie der vorhandenen Richtlinie hinzu.

Wenn Ihr Amazon S3-Bucket-Schlüssel 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-Bucket-Schlüssel 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-policy](#) Befehl an:

```

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

```

Infrastruktursicherheit in AWS Private Certificate Authority

Als verwalteter Service AWS Private Certificate Authority ist durch die AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsservices und wie die Infrastruktur AWS schützt, finden Sie unter [AWS Cloud-Sicherheit](#). Informationen zum Entwerfen Ihrer AWS Umgebung mit den bewährten Methoden für die Infrastruktursicherheit finden Sie unter [Infrastrukturschutz](#) in Security Pillar AWS Well-Architected Framework.

Sie verwenden durch AWS veröffentlichte API-Aufrufe, um AWS Private CA über das Netzwerk auf zuzugreifen. 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.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit [AWS Security Token Service](#) (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

AWS Private CA VPC-Endpunkte (AWS PrivateLink)

Sie können eine private Verbindung zwischen Ihrer VPC und herstellen, AWS Private CA indem Sie einen Schnittstellen-VPC-Endpunkt konfigurieren. Schnittstellenendpunkte werden von unterstützt [AWS PrivateLink](#), einer Technologie für den privaten Zugriff auf AWS Private CA -API-Operationen. AWS PrivateLink leitet den gesamten Netzwerkverkehr zwischen Ihrer VPC und AWS Private CA über das Amazon-Netzwerk weiter, sodass keine Risiken im offenen Internet vermieden werden. Jeder VPC-Endpunkt wird durch eine oder mehrere [Elastic Network-Schnittstellen](#) mit privaten IP-Adressen in Ihren VPC-Subnetzen repräsentiert.

Der Schnittstellen-VPC-Endpunkt verbindet Ihre VPC direkt mit AWS Private CA ohne Internet-Gateway, NAT-Gerät, VPN-Verbindung oder - AWS Direct Connect Verbindung. Die Instances in Ihrer VPC benötigen für die Kommunikation mit der AWS Private CA API keine öffentlichen IP-Adressen.

Um AWS Private CA über Ihre VPC zu verwenden, müssen Sie eine Verbindung von einer Instance innerhalb der VPC herstellen. Alternativ können Sie Ihr privates Netzwerk mit Ihrer VPC verbinden,

indem Sie eine AWS Virtual Private Network (AWS VPN) oder verwenden AWS Direct Connect. Weitere Informationen zu AWS VPN finden Sie unter [VPN-Verbindungen](#) im Amazon-VPC-Benutzerhandbuch. Weitere Informationen zu AWS Direct Connect finden Sie unter [Erstellen einer Verbindung](#) im AWS Direct Connect -Benutzerhandbuch.

AWS Private CA erfordert nicht die Verwendung von AWS PrivateLink, aber wir empfehlen es als zusätzliche Sicherheitsebene. Weitere Informationen zu AWS PrivateLink und VPC-Endpunkten finden Sie unter [Zugriff auf -Services über AWS PrivateLink](#).

Überlegungen zu AWS Private CA VPC-Endpunkten

Beachten Sie die folgenden Überlegungen AWS Private CA, bevor Sie Schnittstellen-VPC-Endpunkte für einrichten:

- AWS Private CA unterstützt in einigen Availability Zones möglicherweise keine VPC-Endpunkte. Wenn Sie einen VPC-Endpunkt erstellen, überprüfen Sie zunächst den Support in der -Managementkonsole. Nicht unterstützte Availability Zones werden als „Service wird in dieser Availability Zone nicht unterstützt“ markiert.
- 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 Certificate Manager unterstützt keine VPC-Endpunkte.
- FIPS-Endpunkte (und ihre Regionen) unterstützen keine VPC-Endpunkte.

AWS Private CA Die -API unterstützt derzeit VPC-Endpunkte in den folgenden AWS-Regionen:

- US East (Ohio)
- USA Ost (Nord-Virginia)
- USA West (Nordkalifornien)
- USA West (Oregon)

- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asien-Pazifik (Singapur)
- Asien-Pazifik (Sydney)
- Asien-Pazifik (Tokio)
- Canada (Central)
- Europe (Frankfurt)
- Europa (Irland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- Europa (Milan)
- Israel (Tel Aviv)
- Naher Osten (Bahrain)
- Südamerika (São Paulo)

Erstellen der VPC-Endpunkte für AWS Private CA

Sie können einen VPC-Endpunkt für den AWS Private CA Service mithilfe der VPC-Konsole unter <https://console.aws.amazon.com/vpc/> oder der erstellen AWS Command Line Interface. Weitere Informationen finden Sie unter [Erstellen eines Schnittstellenendpunkts](#) im Amazon-VPC-Benutzerhandbuch. AWS Private CA unterstützt Aufrufe an alle API-Operationen innerhalb 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

Die Wert *region* stellt die Regionskennung für eine von unterstützte AWS Region dar 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 Endpunkte und Kontingente](#).

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 für 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 [Steuerung des Zugriffs auf Services mit VPC-Endpunkten](#) im Amazon VPC User Guide.

Beispiel – VPC-Endpunktrichtlinie für - AWS Private CA Aktionen

Wenn die folgende Richtlinie an einen Endpunkt angefügt wird, gewährt sie allen Prinzipalen Zugriff auf die AWS Private CA Aktionen `IssueCertificate`, `DescribeCertificateAuthority`, `GetCertificateAuthorityCertificate`, `GetCertificateListPermissions`, 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"
        ]
      }
    ]
  }

```

Protokollierung und Überwachung in AWS Private Certificate Authority

Die Überwachung ist wichtig, um die Zuverlässigkeit, Verfügbarkeit und Leistung von AWS Private Certificate Authority und Ihrer - AWS Lösungen aufrechtzuerhalten. Sie sollten Überwachungsdaten

von allen Teilen Ihrer - AWS Lösung sammeln, damit Sie Ausfälle an mehreren Punkten leichter debuggen können.

In den folgenden Themen werden Tools zur AWS Cloud-Überwachung beschrieben, die für die Verwendung mit verfügbar sind AWS Private CA.

Themen

- [Unterstützte CloudWatch Metriken](#)
- [Verwenden von - CloudWatch Ereignissen](#)
- [Verwenden von CloudTrail](#)

Unterstützte CloudWatch Metriken

Amazon CloudWatch ist ein Überwachungsservice für - AWS Ressourcen. Sie können verwenden, CloudWatch um Metriken zu erfassen und zu verfolgen, Alarme festzulegen und automatisch auf Änderungen in Ihren - AWS Ressourcen zu reagieren. CloudWatch -Metriken werden mindestens einmal veröffentlicht.

AWS Private CA unterstützt die folgenden CloudWatch Metriken.

Metrik	Beschreibung
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 Ausstellungsanforderung und dem Abschluss (oder Fehlschlag) der Ausstellung. Diese Metrik gilt nur für die -IssueCertificateOperation.

Metrik	Beschreibung
Success	Ein Zertifikat wurde erfolgreich ausgestellt. Diese Metrik gilt nur für die -IssueCertificateOperation.
Failure	Eine Operation ist fehlgeschlagen. Diese Metrik gilt nur für die -IssueCertificateOperation.

Weitere Informationen zu CloudWatch Metriken finden Sie in den folgenden Themen:

- [Verwenden von Amazon CloudWatch Metrics](#)
- [Erstellen von Amazon CloudWatch Alarmen](#)

Verwenden von - 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 von - AWS Services werden nahezu in Echtzeit an CloudWatch Ereignisse übermittelt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie von Interesse sind und welche automatisierten Aktionen ausgeführt werden sollen, wenn ein Ereignis mit einer Regel übereinstimmt. CloudWatch Ereignisse werden mindestens einmal veröffentlicht. Weitere Informationen finden Sie unter [Erstellen einer CloudWatch Ereignisregel, die bei einem Ereignis ausgelöst wird](#).

CloudWatch Ereignisse werden mit Amazon in Aktionen umgewandelt EventBridge. Mit können Sie Ereignisse verwenden EventBridge, um Ziele wie AWS Lambda Funktionen, AWS Batch Aufträge, Amazon SNS-Themen und viele andere auszulösen. Weitere Informationen finden Sie unter [Was ist Amazon EventBridge?](#)

Erfolg oder Fehlschlag beim Erstellen einer privaten Zertifizierungsstelle

Diese Ereignisse werden durch die -[CreateCertificateAuthority](#)Operation 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. Mit dem ARN können Sie aufrufen, [DescribeCertificateAuthority](#) um 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 Fehlschlag bei der Ausstellung eines Zertifikats

Diese Ereignisse werden durch die [-IssueCertificate](#) Operation ausgelöst.

Herzlichen Glückwunsch

Bei Erfolg gibt die Operation die 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 aufrufen, [GetCertificate](#) um den Grund für den Fehler anzuzeigen.

```
{
  "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 die [-RevokeCertificate](#) Operation ausgelöst.

Wenn das Widerrufen fehlschlägt oder wenn das Zertifikat bereits widerrufen wurde, wird kein Ereignis gesendet.

Herzlichen Glückwunsch

Bei Erfolg gibt die Operation die ARNs der Zertifizierungsstelle und des widerrufenen 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 Fehlschlag beim Generieren einer CRL

Diese Ereignisse werden durch die [-RevokeCertificate](#) Operation ausgelöst, was zur Erstellung einer Zertifikatsperrliste (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-Bucket-Berechtigungen, wenn 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 – CRL konnte AWS Private CA nicht 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 Fehlschlag beim Erstellen eines CA-Auditberichts

Diese Ereignisse werden durch die [CreateCertificateAuthorityAuditReport](#) Operation 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 Auditbericht kann fehlschlagen, wenn AWS Private CA keine PUT Berechtigungen für Ihren Amazon S3-Bucket vorhanden sind, 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"  
  }  
}
```

Verwenden von CloudTrail

Sie können verwenden [AWS CloudTrail](#), um API-Aufrufe aufzuzeichnen, die von getätigt werden AWS Private Certificate Authority. Weitere Informationen finden Sie unter den folgenden Themen.

Themen

- [Erstellen einer Richtlinie](#)
- [Abrufen einer Richtlinie](#)
- [Löschen einer Richtlinie](#)
- [Erstellen einer Zertifizierungsstelle](#)
- [GenerateCRL](#)
- [GenerateOCSPResponse](#)
- [Erstellen eines Auditberichts](#)
- [Löschen einer Zertifizierungsstelle](#)
- [Wiederherstellen einer Zertifizierungsstelle](#)
- [Beschreiben einer Zertifizierungsstelle](#)
- [Abrufen eines Zertifizierungsstellenzertifikats](#)
- [Abrufen der Signaturanforderung der Zertifizierungsstelle](#)
- [Abrufen eines Zertifikats](#)
- [Importieren eines Zertifizierungsstellenzertifikats](#)
- [Ausstellen eines Zertifikats](#)
- [Auflisten von Zertifizierungsstellen](#)
- [Auflisten von Tags](#)
- [Widerrufen eines Zertifikats](#)
- [Markieren privater Zertifizierungsstellen](#)

- [Entfernen von Tags aus einer privaten Zertifizierungsstelle](#)
- [Aktualisieren einer Zertifizierungsstelle](#)

Erstellen einer Richtlinie

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [PutPolicy](#) Operation.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    },
    "invokedBy": "agent"
  },
  "eventTime": "2021-02-26T21:25:36Z",
  "eventSource": "acm-pca.amazonaws.com",
  "eventName": "PutPolicy",
  "awsRegion": "region",
  "sourceIPAddress": "xx.xx.xx.xx",
  "userAgent": "agent",
  "requestParameters": {
    "resourceArn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
    "policy": "{ \"Version\": \"2012-10-17\", \"Statement\": [{ \"Sid\":
\\\"01234567-89ab-cdef-0123-456789abcdef4-external-principals\\\", \"Effect\": \"Allow
\\\", \"Principal\": { \"AWS\": \"account\" }, \"Action\": \"acm-pca:IssueCertificate
\\\", \"Resource\": \"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566\\\", \"Condition\": { \"StringEquals
\\\": { \"acm-pca:TemplateArn\": \"arn:aws:acm-pca::template/EndEntityCertificate/
V1\" } } }, { \"Sid\": \"01234567-89ab-cdef-0123-456789abcdef-external-principals
\\\", \"Effect\": \"Allow\\\", \"Principal\": { \"AWS\": \"account\" }, \"Action\":
[ \"acm-pca:DescribeCertificateAuthority\\\", \"acm-pca:GetCertificate\\\", \"acm-
pca:GetCertificateAuthorityCertificate\\\", \"acm-pca:ListPermissions\\\", \"acm-
pca:ListTags\" ], \"Resource\": \"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566\" } ] }"
  },
  "responseElements": null,
  "requestID": "01234567-89ab-cdef-0123-456789abcdef",
  "eventID": "01234567-89ab-cdef-0123-456789abcdef",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
}
```



```

"recipientAccountId": "account"
}

```

Abrufen einer Richtlinie

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [GetPolicy](#) Operation.

```

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

      },
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-02-26T20:49:51Z"
      }
    }
  },
  "eventTime": "2021-02-26T21:19:14Z",
  "eventSource": "acm-pca.amazonaws.com",
  "eventName": "GetPolicy",
  "awsRegion": "region",
  "sourceIPAddress": "IP_address",
  "userAgent": "agent",
  "errorCode": "ResourceNotFoundException",
  "errorMessage": "Could not find policy for resource arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566.",
  "requestParameters": {

```

```

    "resourceArn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
  },
  "responseElements": null,
  "requestID": "request_ID",
  "eventID": "event_ID",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "account"
}

```

Löschen einer Richtlinie

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [DeletePolicy](#) Operation.

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "account",
    "arn": "arn:aws:sts::account:assumed-role/role",
    "accountId": "account",
    "accessKeyId": "key_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "account",
        "arn": "arn:aws:iam::account:role/role",
        "accountId": "account",
        "userName": "name"
      }
    },
    "webIdFederationData": {}
  },
  "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2021-02-26T21:23:17Z"
  }
},
  "eventTime": "2021-02-26T21:23:31Z",

```

```

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

```

Erstellen einer Zertifizierungsstelle

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-CreateCertificateAuthority](#) Operation.

```

{
  "eventVersion":"1.05",
  "userIdentity":{
    "type":"IAMUser",
    "principalId":"account",
    "arn":"arn:aws:iam::account:user/name",
    "accountId":"account",
    "accessKeyId":"key_ID"
  },
  "eventTime":"2018-01-26T21:22:33Z",
  "eventSource":"acm-pca.amazonaws.com",
  "eventName":"CreateCertificateAuthority",
  "awsRegion":"region",
  "sourceIPAddress":"IP_address",
  "userAgent":"agent",
  "requestParameters":{
    "certificateAuthorityConfiguration":{
      "keyType":"RSA2048",
      "signingAlgorithm":"SHA256WITHRSA",

```

```

    "subject":{
      "country":"US",
      "organization":"Example Company",
      "organizationalUnit":"Corp",
      "state":"WA",
      "commonName":"www.example.com",
      "locality":"Seattle"
    }
  },
  "revocationConfiguration":{
    "crlConfiguration":{
      "enabled":true,
      "expirationInDays":3650,
      "customCname":"your-custom-name",
      "s3BucketName":"your-bucket-name"
    }
  },
  "certificateAuthorityType":"SUBORDINATE",
  "idempotencyToken":"98256344"
},
"responseElements":{
  "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
},
"requestID":"request_ID",
"eventID":"event_ID",
"eventType":"AwsApiCall",
"recipientAccountId":"account"
}

```

GenerateCRL

Das folgende CloudTrail Beispiel zeigt den Datensatz für ein [GenerateCRL](#)-Ereignis.

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "accountId": "account",
    "invokedBy": "acm-pca.amazonaws.com"
  },
  "eventTime": "2021-02-09T17:37:45Z",
  "eventSource": "acm-pca.amazonaws.com",
  "eventName": "GenerateCRL",

```

```

"awsRegion": "region",
"sourceIPAddress": "acm-pca.amazonaws.com",
"userAgent": "acm-pca.amazonaws.com",
"requestParameters": null,
"responseElements": null,
"eventID": "01234567-89ab-cdef-0123-456789abcdef",
"readOnly": false,
"resources": [
  {
    "type": "AWS::ACMPCA::CertificateAuthority",
    "ARN": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
  }
],
"eventType": "AwsServiceEvent",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "account"
}

```

GenerateOCSPResponse

Das folgende CloudTrail Beispiel zeigt den Datensatz für ein [GenerateOCSPResponse](#)-Ereignis.

```

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

```

```

    }
  ]
}

```

Erstellen eines Auditberichts

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-CreateCertificateAuthorityAuditReport](#) Operation.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "account",
    "arn": "arn:aws:iam:account:user/name",
    "accountId": "account",
    "accessKeyId": "key_ID"
  },
  "eventTime": "2018-01-26T21:56:00Z",
  "eventSource": "acm-pca.amazonaws.com",
  "eventName": "CreateCertificateAuthorityAuditReport",
  "awsRegion": "region",
  "sourceIPAddress": "IP_address",
  "userAgent": "agent",
  "requestParameters": {
    "certificateAuthorityArn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
    "s3BucketName": "bucket_name",
    "auditReportResponseFormat": "JSON"
  },
  "responseElements": {
    "auditReportId": "report_ID",
    "s3Key": "audit-report/CA_ID/audit_report_ID.json"
  },
  "requestID": "request_ID",
  "eventID": "event_ID",
  "eventType": "AwsApiCall",
  "recipientAccountId": "account"
}

```

Löschen einer Zertifizierungsstelle

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-DeleteCertificateAuthority](#) Operation. In diesem Beispiel kann die Zertifizierungsstelle nicht gelöscht werden, da sie sich im Status ACTIVE befindet.

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

Wiederherstellen einer Zertifizierungsstelle

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-RestoreCertificateAuthority](#) Operation. In diesem Beispiel kann die Zertifizierungsstelle nicht wiederhergestellt werden, da sie sich nicht im Status DELETED befindet.

```
{
  "eventVersion":"1.05",
```

```

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

```

Beschreiben einer Zertifizierungsstelle

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-DescribeCertificateAuthority](#) Operation.

```

{
  "eventVersion":"1.05",
  "userIdentity":{
    "type":"IAMUser",
    "principalId":"account",
    "arn":"arn:aws:iam::account:user/name",
    "accountId":"account",
    "accessKeyId":"key_ID"
  },
  "eventTime":"2018-01-26T21:58:18Z",
  "eventSource":"acm-pca.amazonaws.com",
  "eventName":"DescribeCertificateAuthority",

```



```

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

```

Abrufen eines Zertifizierungsstellenzertifikats

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-GetCertificateAuthorityCertificate](#) Operation.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "account",
    "arn": "arn:aws:iam::account:user/name",
    "accountId": "account",
    "accessKeyId": "key_ID"
  },
  "eventTime": "2018-01-26T22:03:52Z",
  "eventSource": "acm-pca.amazonaws.com",
  "eventName": "GetCertificateAuthorityCertificate",
  "awsRegion": "region",
  "sourceIPAddress": "IP_address",
  "userAgent": "agent",
  "requestParameters": {
    "certificateAuthorityArn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
  },
  "responseElements": null,
  "requestID": "request_ID",
  "eventID": "event_ID",
  "eventType": "AwsApiCall",
  "recipientAccountId": "account"
}

```

```
}
```

Abrufen der Signaturanforderung der Zertifizierungsstelle

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-GetCertificateAuthorityCsr](#) Operation.

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

Abrufen eines Zertifikats

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-GetCertificate](#) Operation.

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

```

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

```

Importieren eines Zertifizierungsstellenzertifikats

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-ImportCertificateAuthorityCertificate](#) Operation.

```

{
  "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"
}
```

Ausstellen eines Zertifikats

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [IssueCertificate](#) Operation.

```
{
  "eventVersion":"1.05",
  "userIdentity":{
    "type":"IAMUser",
    "principalId":"account",
    "arn":"arn:aws:iam::account:user/name",
    "accountId":"account",
    "accessKeyId":"key_ID"
  },
  "eventTime":"2018-01-26T22:18:43Z",
  "eventSource":"acm-pca.amazonaws.com",
  "eventName":"IssueCertificate",
  "awsRegion":"region",
  "sourceIPAddress":"xIP_address",
  "userAgent":"agent",
  "requestParameters":{
    "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
    "csr":{
      "hb":[
        45,
        45,
        ...10
      ],
      "offset":0,
      "isReadOnly":false,
      "bigEndian":true,
      "nativeByteOrder":false,
      "mark":-1,
      "position":1090,
      "limit":1090,
      "capacity":1090,
      "address":0
    },
    "signingAlgorithm":"SHA256WITHRSA",
    "validity":{
      "value":365,
      "type":"DAYS"
    }
  },
  "idempotencyToken":"1234"
```

```

},
"responseElements":{
  "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/certificate/certificate_ID"
},
"requestID":"request_ID",
"eventID":"event_ID",
"eventType":"AwsApiCall",
"recipientAccountId":"account"
}

```

Auflisten von Zertifizierungsstellen

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-ListCertificateAuthorities](#) Operation.

```

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

```

Auflisten von Tags

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [ListTags](#) Operation.

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

Widerrufen eines Zertifikats

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-RevokeCertificate](#) Operation.

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

Markieren privater Zertifizierungsstellen

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [-TagCertificateAuthority](#) Operation.

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

```



```

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

```

Entfernen von Tags aus einer privaten Zertifizierungsstelle

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [UntagCertificateAuthority](#) Operation.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "account",
    "arn": "arn:aws:iam::account:user/name",
    "accountId": "account",
    "accessKeyId": "key_ID"
  },
  "eventTime": "2018-02-02T00:21:50Z",
  "eventSource": "acm-pca.amazonaws.com",
  "eventName": "UntagCertificateAuthority",
  "awsRegion": "region",

```

```

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

```

Aktualisieren einer Zertifizierungsstelle

Das folgende CloudTrail Beispiel zeigt die Ergebnisse eines Aufrufs der [UpdateCertificateAuthority](#) Operation.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "account",
    "arn": "arn:aws:iam::account:user/name",
    "accountId": "account",
    "accessKeyId": "key_ID"
  },
  "eventTime": "2018-01-26T22:08:59Z",
  "eventSource": "acm-pca.amazonaws.com",
  "eventName": "UpdateCertificateAuthority",
  "awsRegion": "region",
  "sourceIPAddress": "IP_address",
  "userAgent": "agent",
  "requestParameters": {
    "certificateAuthorityArn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",

```

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

Planung Ihres AWS Private CA Einsatzes

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 Zertifizierungsstellen bis hin zu Endzertifizierungszertifikaten 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 vor Ort zu hosten als mit 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](#).

Themen

- [Einrichtung Ihres AWS Kontos und AWS CLI](#)
- [Entwerfen einer CA-Hierarchie](#)
- [Verwaltung des privaten CA-Lebenszyklus](#)
- [Einrichtung einer Methode zum Widerruf von Zertifikaten](#)
- [Modi der Zertifizierungsstelle](#)
- [Planung für Resilienz](#)

Einrichtung Ihres AWS Kontos und AWS CLI

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.

 Note

AWS Private CA ist im [AWS kostenlosen Kontingent](#) nicht verfügbar.

Themen

- [Melden Sie sich für eine an AWS-Konto](#)
- [Erstellen eines Administratorbenutzers](#)
- [Installieren Sie das AWS Command Line Interface](#)

Melden Sie sich für eine an 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/signup>.
2. Folgen Sie den Online-Anweisungen.

Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontos wird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem [Administratorbenutzer 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. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu <https://aws.amazon.com/> auf und klicken Sie auf Mein Konto.

Erstellen eines Administratorbenutzers

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 Management Console](#) als 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 Administratorbenutzers

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 Administratorzugriff.

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](#).

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.

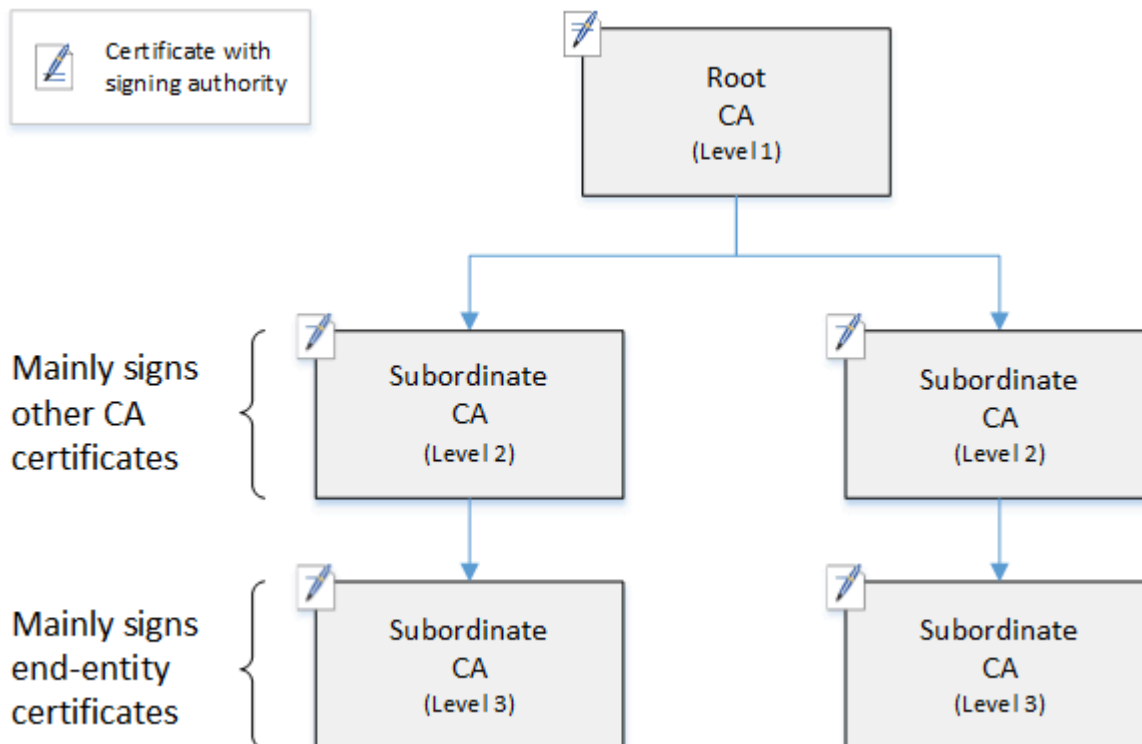
Entwerfen einer CA-Hierarchie

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 Stamm-CA kann bis zu vier Ebenen untergeordneter CAs in jedem Zweig aufweisen. 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 begrenzter, widerruflicher Vertrauensstellung 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 pen-and-paper Symbol) unterstützt. Das bedeutet, dass sie als CAs andere untergeordnete Zertifikate signieren können. 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 auf Ebene 3, die von PKI-Administratoren (Public Key-Infrastruktur) verwendet werden, die Endentitätszertifikate 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 Stamm-CA verankert die Vertrauensstellung für alle untergeordneten CAs und die darunter liegenden Endentitätszertifikate. Während lokaler Schaden durch die Kompromittierung eines Endentitätszertifikats entstehen kann, zerstört die Kompromittierung des Stammes die Vertrauensstellung in der gesamten PKI. Stammzertifizierungsstellen und untergeordnete Zertifizierungsstellen auf hoher Ebene 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

Stamm-CAs und andere CA auf hoher Ebene erfordern in hohem Maße sichere Betriebsprozesse und Zugriffssteuerungsprotokolle.

Themen

- [Validierung von Endzertifikaten](#)

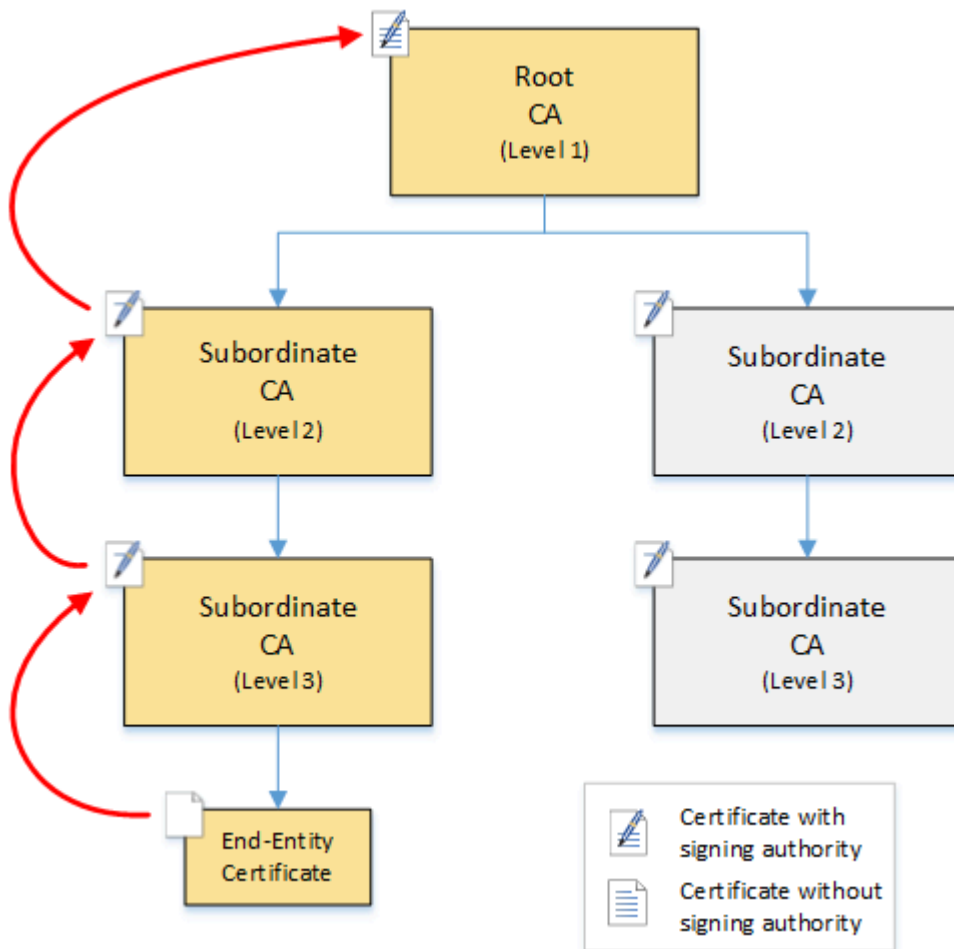
- [Planung der Struktur einer CA-Hierarchie](#)
- [Festlegung von Längenbeschränkungen für den Zertifizierungspfad](#)

Validierung von Endzertifikaten

Endentitätszertifikate leiten ihre Vertrauensstellung von einem Zertifizierungspfad ab, der durch die untergeordneten CAs zu einer Stamm-CA 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 Vertrauensspeicher ist eine Bibliothek vertrauenswürdiger CAs, die der Browser oder 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 die untergeordneten 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.

Planung der 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 Delegation 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 [pathLenConstraint](#) Parameter hat, eine unbegrenzte Anzahl untergeordneter Zertifizierungsstellen autorisieren. Eine untergeordnete Zertifizierungsstelle kann so viele untergeordnete Zertifizierungsstellen haben, 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 Möglichkeit, den Zugriff auf untergeordnete CAs zu delegieren
- Eine hierarchische Struktur, die CAs höherer Ebenen mit zusätzlichen Sicherheitskontrollen 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-Ebene wird nur verwendet, um untergeordnete CAs zu signieren, die die Ausstellung von Endentitätszertifikaten durchführen.

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 Delegation administrativer Aufgaben zu ermöglichen.

- Eine CA-Ebene: nur Stamm-CA

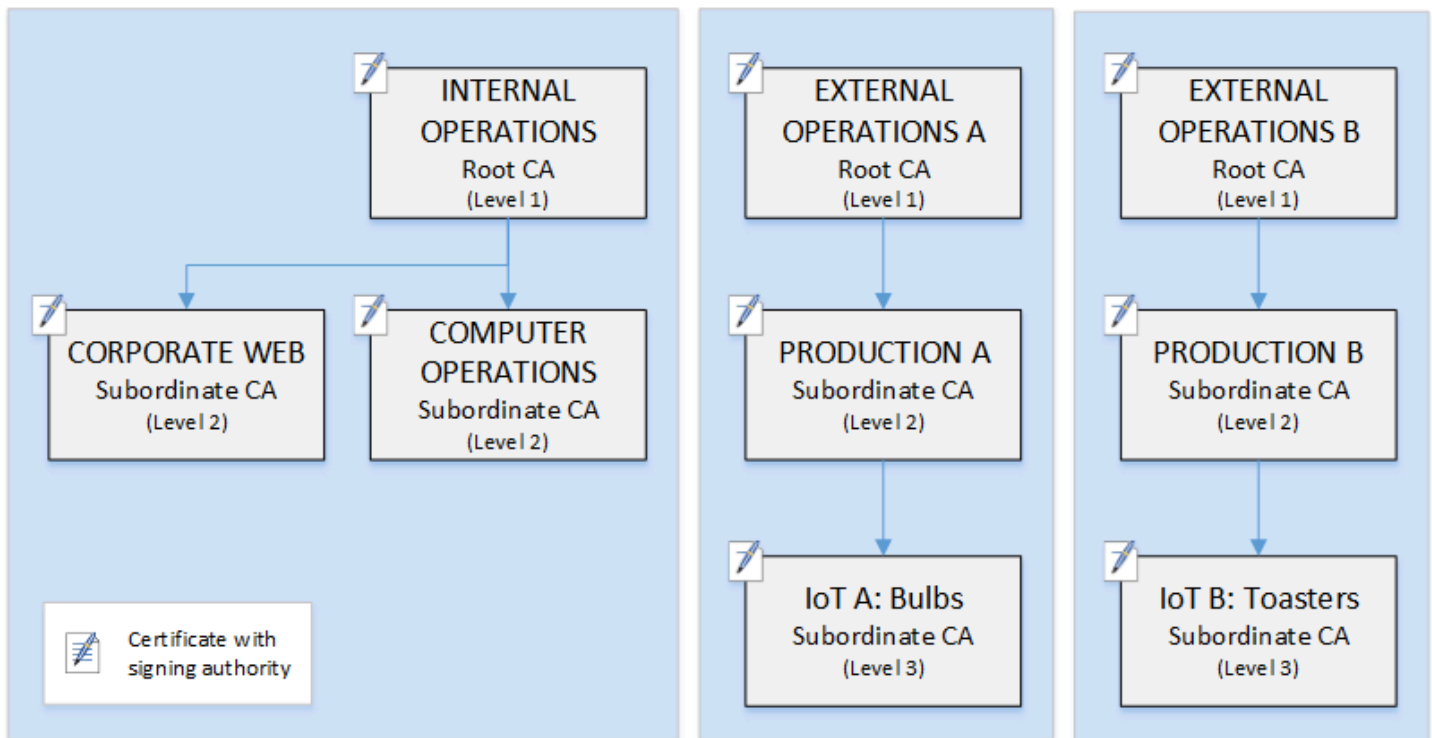
Diese Struktur wird häufig für die Entwicklung und das Durchführen von Tests verwendet, wenn keine vollständige Vertrauenskette erforderlich ist. Ihre Verwendung in der Produktion ist untypisch. Darüber hinaus verstößt sie gegen die bewährten Methoden zur Aufrechterhaltung separater Sicherheitsrichtlinien für die Stamm-CA und die CAs, die Endentitätszertifikate ausstellen.

Wenn Sie jedoch bereits Zertifikate direkt von einer Stammzertifizierungsstelle ausstellen, können Sie zu AWS Private CA dieser migrieren. 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 Stammzertifizierungsstellen und zusätzlicher untergeordneter Zertifizierungsstellen auf Seiten des externen Betriebs 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 mit zwei Ebenen untergeordneter CAs weisen die Stamm-CAs eine hohe Isolation von den CAs der Ebene 3 auf, die für die Massensignierung der Zertifikate für Tausende oder Millionen von hergestellten Artikeln verantwortlich sind.

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.

Festlegung von Längenbeschränkungen für den Zertifizierungspfad

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 Zertifizierungsstellen, die in einer gültigen Vertrauenskette existieren können. Das Zertifikat der Endeinheit wird nicht gezählt, da es sich nicht um ein CA-Zertifikat handelt.

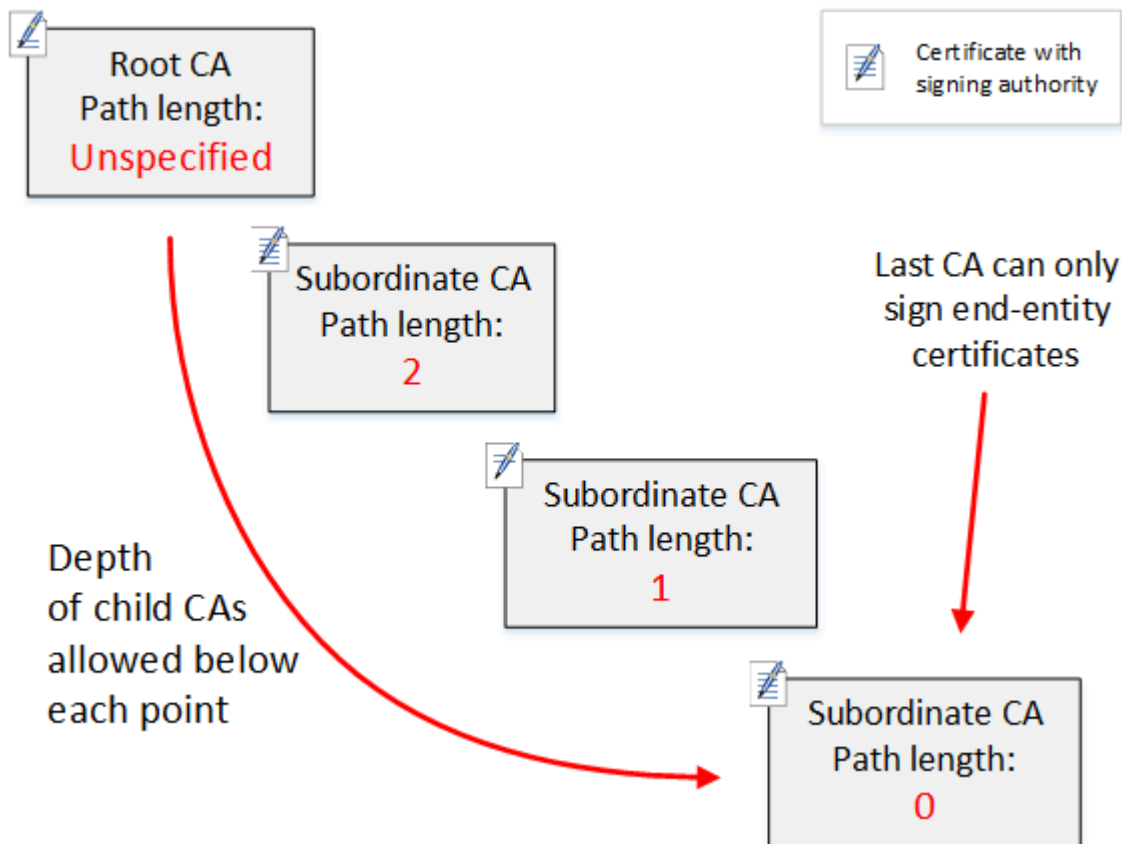
Ein Stamm-CA-Zertifikat erfordert maximale Flexibilität und enthält keine Pfadlängenbeschränkung. Dadurch kann der Stamm einen Zertifizierungspfad beliebiger Länge definieren.

Note

AWS Private CA begrenzt den Zertifizierungspfad auf fünf Stufen.

Untergeordnete CAs haben `pathLenConstraint`-Werte größer oder gleich Null, abhängig von der Position in der Hierarchie und den gewünschten Funktionen. Beispielsweise wird in einer Hierarchie mit drei CAs keine Pfadbeschränkung für die Stamm-CA angegeben. Die erste untergeordnete CA hat eine Pfadlänge von 1 und kann daher untergeordnete CAs signieren. Jede dieser untergeordneten CAs muss notwendigerweise einen `pathLenConstraint`-Wert von Null haben. Dies bedeutet, dass sie Endentitätszertifikate signieren, jedoch keine zusätzlichen CA-Zertifikate ausstellen können. Die Begrenzung der Berechtigung, neue CAs zu erstellen, 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). Aber die erste untergeordnete CA hat einen `pathLenConstraint`-Wert von 2, wodurch ihre untergeordneten CAs nicht mehr als zwei Ebenen tiefer gehen. 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.

Verwaltung der 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:

- RootCACertificate/V1
- Untergeordnetes CA-Zertifikat _ 0/V1 PathLen
- Untergeordnetes CA-Zertifikat _ 1/V1 PathLen

- Untergeordnetes CA-Zertifikat _ 2/V1 PathLen
- Untergeordnetes CA-Zertifikat _ 3/V1 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 [Grundlegendes zu Zertifikatsvorlagen](#).

Automatisieren der 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 AWS CloudFormation - Benutzerhandbuch.

Verwaltung des 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 untergeordneten CAs in der CA-Hierarchie ausgestellt wurden, ungültig. Sie können den Ablauf von CA-Zertifikaten vermeiden, indem Sie im Voraus planen.

Gültigkeitszeiträume wählen

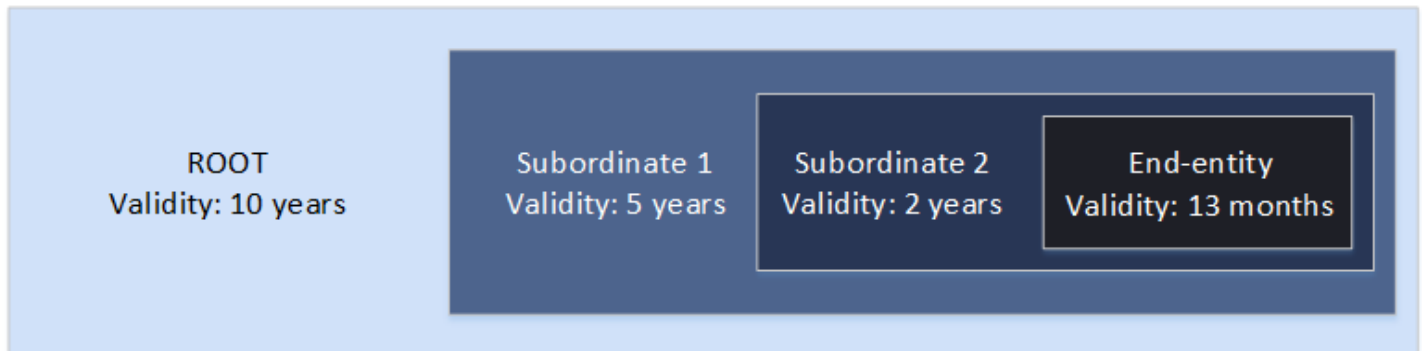
Der Gültigkeitszeitraum eines X.509-Zertifikats ist ein erforderliches grundlegendes Zertifikatsfeld. Er bestimmt den Zeitraum, in dem die ausstellende CA bescheinigt, dass das Zertifikat vertrauenswürdig ist, mit Ausnahme einer Sperre. (Ein Stammzertifikat, das selbst signiert wird, bescheinigt seinen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

AWS Private CA und AWS Certificate Manager helfen 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 können untergeordnete CAs und Endentitätszertifikate ihre übergeordneten Zertifikate nicht überleben. 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 ist das langlebigste Zertifikat Endentitätszertifikate sind relativ kurzlebig und untergeordnete CAs befinden sich im Bereich 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.

Verwaltung der CA-Nachfolge

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.

Eine alte CA ersetzen

Um eine alte CA zu ersetzen, erstellen Sie eine neue CA und verketteten 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 OCSP und/oder Zertifikatssperllisten (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 CA untergeordnete CAs hat, müssen Sie diese ebenfalls ersetzen, da untergeordnete CAs zur gleichen Zeit oder vor ihrer übergeordneten CA ablaufen. Ersetzen Sie zunächst die höchste CA in der Hierarchie, die ersetzt werden muss. Erstellen Sie dann neue untergeordnete Ersatz-CAs auf jeder nachfolgenden unteren Ebene.

AWS empfiehlt, dass Sie bei Bedarf eine Kennung für die Generierung einer Zertifizierungsstelle in die Namen der Zertifizierungsstellen aufnehmen. Nehmen wir beispielsweise an, dass Sie die Zertifizierungsstelle 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, Verwirrung zu vermeiden, wenn keine der beiden CAs abgelaufen ist.

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 Rotationsfrequenz sollte proportional zur Häufigkeit der Schlüsselverwendung sein: CAs, die mehr Zertifikate ausstellen, sollten häufiger rotiert werden.

Note

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

Neuausstellung einer alten Zertifizierungsstelle

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. [Erstellen und Installieren 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 einer CA

Sie widerrufen eine Zertifizierungsstelle, 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.

Einrichtung einer Methode zum Widerruf von 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 das Online Certificate Status Protocol (OCSP), Certificate Revocation Lists (CRLs) oder beides wählen.

Note

Wenn Sie Ihre Zertifizierungsstelle erstellen, ohne den Widerruf zu konfigurieren, können Sie sie jederzeit später konfigurieren. Weitere Informationen finden Sie unter [Aktualisieren Ihrer privaten Zertifizierungsstelle](#).

- 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 auf neuen oder vorhandenen Zertifizierungsstellen mit einem einzigen Vorgang über die AWS Private CA Konsole, die API, die CLI oder über AWS CloudFormation die Konsole aktivieren. Während CRLs auf dem Endpunkt

gespeichert und verarbeitet werden und veralten können, werden OCSP-Speicher- und Verarbeitungsanforderungen synchron im 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. In der AWS Private CA Konsole erstellte Zertifizierungsstellen verwenden standardmäßig den SHA256WITHRSA-Signaturalgorithmus. Andere unterstützte Algorithmen finden Sie in der API-Dokumentation. [CertificateAuthorityConfiguration](#)
- [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 CRL enthält eine Liste von gesperrten Zertifikaten. Wenn Sie eine Zertifizierungsstelle für die Generierung von CRLs konfigurieren, AWS Private CA schließt sie die Erweiterung CRL Distribution Points in jedes neu ausgestellte Zertifikat ein. Diese Erweiterung stellt die URL für die CRL bereit. Die Erweiterung ermöglicht es Clients wie Webbrowsern, die CRL abzufragen und festzustellen, ob einem Zertifikat der Endeinheit oder einer untergeordneten Zertifizierungsstelle vertraut werden kann.

Da ein Client CRLs herunterladen und lokal verarbeiten muss, ist ihre Verwendung speicherintensiver als bei OCSP. CRLs verbrauchen möglicherweise weniger Netzwerkbandbreite, da die Liste der CRLs heruntergeladen und zwischengespeichert wird, verglichen mit OCSP, das den Sperrstatus bei jedem neuen Verbindungsversuch überprüft.

Note

Sowohl bei OCSP als auch bei CRLs 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, die von Clients tagelang zwischengespeichert werden können, normalerweise nicht von Clients zwischengespeichert werden.
- Eine Zertifikatssperrliste wird in der Regel etwa 30 Minuten nach Widerruf eines Zertifikats aktualisiert. Falls eine CRL-Aktualisierung aus irgendeinem Grund fehlschlägt, werden alle 15 Minuten weitere Versuche unternommen AWS Private CA .

Allgemeine Anforderungen für Sperrkonfigurationen

Die folgenden Anforderungen gelten für alle Sperrkonfigurationen.

- Eine Konfiguration, die CRLs oder OCSP deaktiviert, darf nur den `Enabled=False`-Parameter enthalten und schlägt fehl, wenn andere Parameter wie `CustomCname` oder `ExpirationInDays` enthalten 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 [RFC7230-Beschränkungen](#) 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

- [Planung einer Zertifikatssperrliste \(CRL\)](#)
- [Konfiguration einer benutzerdefinierten URL für OCSP AWS Private CA](#)

Planung einer Zertifikatssperrliste (CRL)

Bevor Sie im Rahmen der [Erstellung einer Zertifizierungsstelle eine CRL konfigurieren können, sind möglicherweise](#) einige Einrichtungsschritte erforderlich. 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. [Optionen zum Widerrufen von Zertifikaten Konfiguration einer benutzerdefinierten URL für OCSP AWS Private CA](#)

Themen

- [CRL-Struktur](#)
- [Zugriffsrichtlinien für CRLs in Amazon S3](#)
- [Aktivieren von S3 Block Public Access \(BPA\) mit CloudFront](#)
- [Verschlüsseln Ihrer CRLs](#)
- [Ermitteln des CRL Distribution Point \(CDP\) -URI](#)

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
```

Zertifikatssperrlisten 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. Zuvor 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. Während des Konsolenvorgangs zum [Erstellen](#) einer CA können Sie festlegen, dass ein neuer Bucket AWS Private CA erstellt und eine Standardberechtigungsrichtlinie angewendet wird.

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](#).

Wenn Sie CRLs als Methode zum Widerruf von Zertifikaten konfigurieren, 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 . <i>Region</i> . amazonaws . com

PCA	S3	Dienstauftraggeber
Enabled	Disabled	Nicht unterstützt

Die Standardrichtlinie gilt `SourceArn` nicht für die CA. Wir empfehlen, dass Sie die unten dargestellte, weniger freizügige Richtlinie manuell anwenden, die den Zugriff sowohl auf ein bestimmtes AWS Konto als auch auf eine bestimmte private Zertifizierungsstelle einschränkt. Weitere Informationen finden Sie unter [Hinzufügen einer Bucket-Richtlinie mithilfe der Amazon S3 S3-Konsole](#).

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

Wenn Sie sich dafür entscheiden, die Standardrichtlinie zuzulassen, können Sie sie später jederzeit [ändern](#).

Aktivieren von S3 Block Public Access (BPA) mit CloudFront

Neue Amazon S3 S3-Buckets werden standardmäßig mit aktivierter Funktion Block Public Access (BPA) konfiguriert. BPA ist Teil der [Best Practices für Amazon S3 S3-Sicherheit](#) und umfasst eine Reihe von Zugriffskontrollen, mit denen Kunden den Zugriff auf Objekte in ihren S3-Buckets und auf die Buckets als Ganzes 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 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), konfigurieren, um CRLs bereitzustellen, 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.

Amazon S3 mit BPA einrichten

Erstellen Sie in S3 wie gewohnt einen neuen Bucket für Ihre CRL und aktivieren Sie dann BPA darin.

So konfigurieren Sie einen Amazon S3 S3-Bucket, der den öffentlichen Zugriff auf Ihre CRL blockiert

1. Erstellen Sie einen neuen S3-Bucket wie unter [Bucket erstellen beschrieben](#). Wählen Sie während des Vorgangs die Option **Allen öffentlichen Zugriff blockieren** aus.

Weitere Informationen finden Sie unter [Sperren des öffentlichen Zugriffs auf Ihren Amazon S3 S3-Speicher](#).

2. Wenn der Bucket erstellt wurde, wählen Sie seinen Namen aus der Liste aus, navigieren Sie zur Registerkarte „Berechtigungen“, wählen Sie im Bereich „Objekteigentümer“ die Option „Bearbeiten“ und wählen Sie „Bevorzugter Bucket-Besitzer“ aus.
3. Fügen Sie dem Bucket ebenfalls auf der Registerkarte „Berechtigungen“ eine IAM-Richtlinie hinzu, wie unter beschrieben [Zugriffsrichtlinien für CRLs in Amazon S3](#).

CloudFront Für BPA einrichten

Erstellen Sie eine CloudFront Distribution, die Zugriff auf Ihren privaten S3-Bucket hat und CRLs für nicht authentifizierte Clients bereitstellen 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 unter Leseberechtigungen für Bucket gewähren die Option Ja, Bucket-Richtlinie aktualisieren 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 Domännennamen 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 Ihrer CloudFront Distribution hinzu.

Um Ihre CA mit einem CNAME zu konfigurieren für CloudFront

- Erstellen Sie Ihre CA mit [Verfahren zum Erstellen einer Zertifizierungsstelle \(CLI\)](#).

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.

Verschlüsseln Ihrer CRLs

Sie können optional die Verschlüsselung für den Amazon S3 S3-Bucket konfigurieren, der Ihre CRLs enthält. AWS Private CA unterstützt zwei Verschlüsselungsmodi für Assets in Amazon 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-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 Amazon S3 S3-Schlüssel (SSE-S3).
7. Wählen Sie Save Changes.

So konfigurieren Sie die benutzerdefinierte Verschlüsselung

Gehen Sie wie folgt vor, um die Verschlüsselung mit einem benutzerdefinierten Schlüssel zu aktivieren.

1. Öffnen Sie die Amazon-S3-Konsole unter <https://console.aws.amazon.com/s3/>.
2. Wählen Sie in der Tabelle Buckets den Bucket aus, der Ihre AWS Private CA Assets 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-policy](#) Befehl:

```
$ 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-policy](#) Befehl an:

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

Ermitteln des CRL Distribution Point (CDP) -URI

Wenn Sie den S3-Bucket als CDP für Ihre CA verwenden, kann der CDP-URI in einem der folgenden Formate vorliegen.

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

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

Konfiguration einer benutzerdefinierten URL für OCSP AWS Private CA

Note

Dieses Thema richtet sich an Kunden, die die öffentliche URL des OCSP-Responder-Endpunkts 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

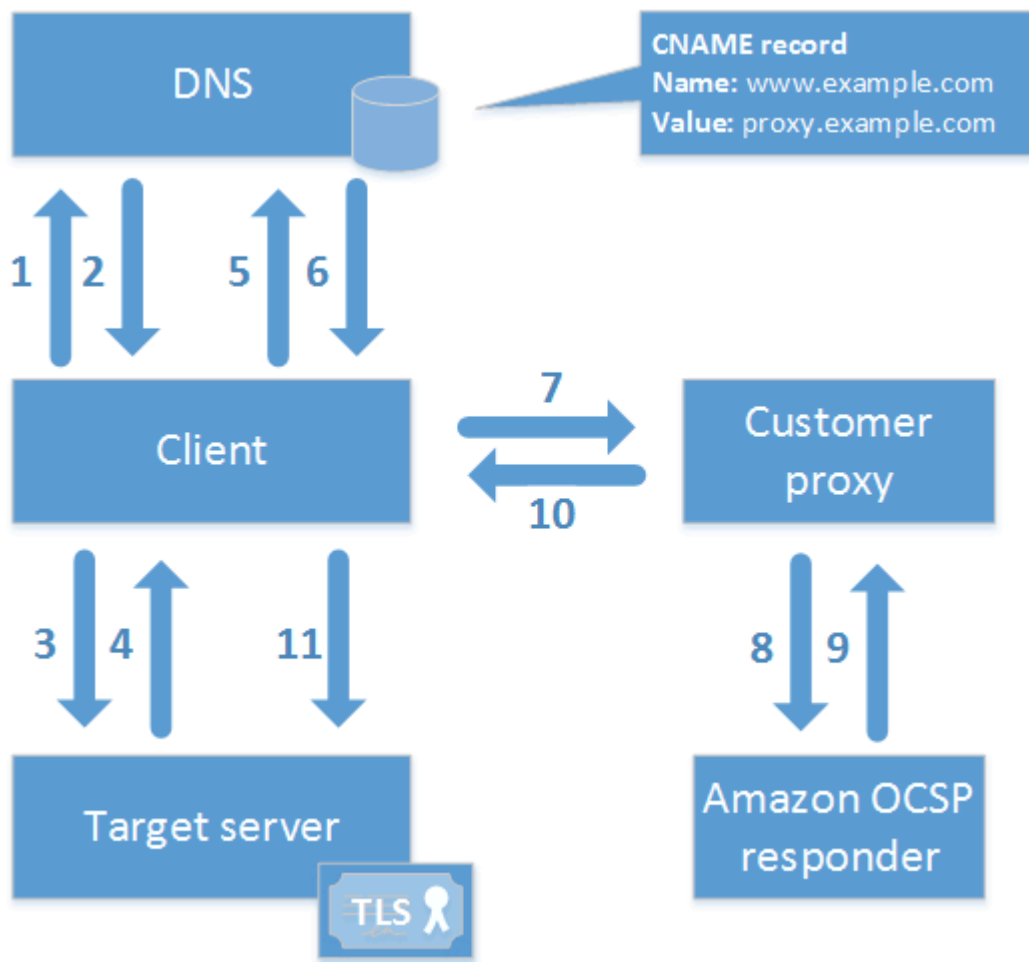
 Note

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 beschrieben [Beispiel 2: Erstellen einer Zertifizierungsstelle mit aktiviertem OCSP und einem benutzerdefinierten CNAME](#). [Verfahren zum Erstellen einer Zertifizierungsstelle \(CLI\)](#)
- 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 einer Zertifizierungsstelle mit aktiviertem OCSP und einem benutzerdefinierten CNAME](#) in [Verfahren zum Erstellen einer Zertifizierungsstelle \(CLI\)](#).
- 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.
 - Stellen Sie ocsf.acm-pca ein. <region>.amazonaws.com als Ursprung.
 - 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.

Modi der Zertifizierungsstelle

AWS Private CA unterstützt die Erstellung einer Zertifizierungsstelle in einem von zwei Modi. Die Modi GENERAL_PURPOSE und SHORT_LIVED_CERTIFICATE wirken sich auf die zulässige Gültigkeitsdauer der von der CA ausgestellten Zertifikate aus.

Note

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

GENERAL_PURPOSE (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.

SHORT_LIVED_CERTIFICATE

Dieser Modus definiert eine CA, 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.

Zertifizierungsstellen mit dem Modus SHORT_LIVED_CERTIFICATE kosten weniger als Zertifizierungsstellen für allgemeine Zwecke. [Weitere Informationen finden Sie unter Preise.AWS Private Certificate Authority](#)

Um eine Zertifizierungsstelle zu erstellen, die kurzlebige Zertifikate ausstellt, setzen Sie den UsageMode Parameter auf SHORT_LIVED_CERTIFICATE, indem Sie das Verfahren zum Erstellen einer Zertifizierungsstelle verwenden. [AWS CLI](#)

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](#)

Planung für Resilienz

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 redundante Zertifizierungsstellen in mehreren Regionen erstellen 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. Sie können aus Redundanz- und Notfallwiederherstellungsgründen zwei Stammzertifizierungsstellen in zwei verschiedenen AWS Regionen erstellen. 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 Stamm-CAs erhöht jedoch die betriebliche Komplexität: Sie müssen beide Stamm-CA-Zertifikate an die Vertrauensspeicher von Browsern und Betriebssystemen in Ihrer Umgebung verteilen.
2. Sie können auch redundante untergeordnete Zertifizierungsstellen für den Einsatz in jeder Ihrer AWS Regionen einrichten und diese mit derselben eindeutigen Stammzertifizierungsstelle in einer einzigen AWS Region verknüpfen. 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.

Bewährte Methoden für AWS Private CA

Bewährte Methoden sind Empfehlungen, die Ihnen helfen, AWS Private CA effektiv zu verwenden. Die folgenden bewährten Methoden basieren auf praktischen Erfahrungen aktueller AWS Certificate Manager- und AWS Private CA-Kunden.

Dokumentieren der CA-Struktur und -Richtlinien

AWS empfiehlt, alle Richtlinien und Verfahren für den Betrieb Ihrer Zertifizierungsstelle zu dokumentieren. Dazu kann Folgendes gehören:

- Schlussfolgerung für Ihre Entscheidungen bezüglich der CA-Struktur
- Ein Diagramm mit Ihren Zertifizierungsstellen und deren Beziehungen
- Richtlinien zu CA-Gültigkeitszeiträumen
- Planung der CA-Abfolge
- Richtlinien zur Pfadlänge
- Berechtigungskatalog
- 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](#).

Minimieren Sie die Verwendung der Stammzertifizierungsstelle, wenn möglich

Eine Stammzertifizierungsstelle sollte in der Regel nur zur Ausstellung von Zertifikaten für Zwischen-Zertifizierungsstellen verwendet werden. Dadurch kann die Stammzertifizierungsstelle gespeichert werden, ohne Schaden zu nehmen, während die Zwischenzertifizierungsstellen das tägliche Ausstellen von Endentitätszertifikaten übernehmen.

Wenn Ihre Organisation derzeit jedoch Endentitätszertifikate direkt über eine Stammzertifizierungsstelle ausstellt, kann AWS Private CA diesen Workflow unterstützen und gleichzeitig die Sicherheit sowie die Betriebssteuerung 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 legt Einschränkungen fest, 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 Stammzertifizierungsstelle eine eigene AWS-Konto

Das Erstellen einer Stammzertifizierungsstelle und einer untergeordneten Zertifizierungsstelle in zwei verschiedenen AWS Konten wird als bewährte Methode empfohlen. 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. Dieses Verfahren hat den Vorteil, dass Sie die Kontrolle über Ihre Zertifizierungsstellen nach Konto trennen können. Der Nachteil besteht darin, dass Sie den AWS Management Console Assistenten nicht verwenden können, um das Signieren des CA-Zertifikats einer untergeordneten CA von Ihrer Stammzertifizierungsstelle zu vereinfachen.

Important

Wir empfehlen dringend die Verwendung der Multi-Faktor-Authentifizierung (MFA), wenn Sie auf zugreifenAWS Private CA.

Separate Administrator- und Ausstellerrollen

Die CA-Administratorrolle sollte von Benutzern getrennt sein, die nur Endentitätszertifikate ausstellen müssen. Wenn sich Ihr CA-Administrator und Ihr Zertifikataussteller in derselben befindenAWS-Konto, können Sie die Ausstellerberechtigungen einschränken, indem Sie einen IAM-Benutzer speziell für diesen Zweck erstellen.

Implementieren des verwalteten Widerrufs von Zertifikaten

Der verwaltete Widerruf informiert Zertifikat-Clients automatisch, wenn ein Zertifikat widerrufen wurde. Möglicherweise müssen Sie ein Zertifikat widerrufen, wenn seine kryptografischen Informationen kompromittiert wurden oder wenn es falsch ausgestellt wurde. Clients lehnen in der Regel die Annahme widerrufenen Zertifikate ab. AWS Private CA bietet zwei Standardoptionen für den verwalteten Widerruf: Online Certificate Status Protocol (OCSP) und Zertifikatsperrlisten (CRLs). Weitere Informationen finden Sie unter [Einrichtung einer Methode zum Widerruf von Zertifikaten](#).

AWS CloudTrail aktivieren

Aktivieren Sie die CloudTrail Protokollierung, bevor Sie eine private Zertifizierungsstelle erstellen und mit dem Betrieb beginnen. Mit können CloudTrailSie einen Verlauf der AWS API-Aufrufe für Ihr Konto abrufen, um Ihre AWS Bereitstellungen zu überwachen. Dieser Verlauf umfasst API-Aufrufe, die von der AWS Management Console, den -AWSSDKs AWS Command Line Interface, der und AWS Services auf höherer Ebene ausgingen. SDKs 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 mithilfe der API CloudTrail in Anwendungen integrieren, die Trail-Erstellung für Ihre Organisation automatisieren, den Status Ihrer Trails überprüfen und steuern, wie Administratoren die CloudTrail Protokollierung aktivieren und deaktivieren. Weitere Informationen finden Sie unter [Erstellen eines Trails](#). Beispiel-Trails für AWS Private CA-Operationen finden Sie unter [Verwenden von CloudTrail](#).

Rotieren des privaten Schlüssels der Zertifizierungsstelle

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 ersetzen, beachten Sie, dass sich der ARN der CA ändert. Dies würde dazu führen, dass die Automatisierung, die sich auf einen hartcodierten ARN stützt, fehlschlägt.

Löschen ungenutzter 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 Zertifizierungsstelle](#) beschriebene Vorgehensweise.

Note

AWS stellt Ihnen eine Zertifizierungsstelle so lange in Rechnung, bis sie gelöscht wurde.

Den öffentlichen Zugriff auf Ihre CRLs blockieren

AWS Private CA empfiehlt die Verwendung der Funktion Amazon S3 Block Public Access (BPA) für Buckets, die CRLs enthalten. Dadurch werden unnötig Details Ihrer privaten PKI potenziellen Angreifern zugänglich gemacht. BPA ist eine [bewährte Methode](#) für S3 und ist standardmäßig für neue Buckets aktiviert. In einigen Fällen ist eine zusätzliche Einrichtung erforderlich. Weitere Informationen finden Sie unter [Aktivieren von S3 Block Public Access \(BPA\) mit CloudFront](#).

Bewährte Methoden für Amazon-EKS-Anwendungen

Wenn Sie verwenden, AWS Private CA um Amazon EKS mit X.509-Zertifikaten bereitzustellen, befolgen Sie die Empfehlungen zum Sichern von Umgebungen mit mehreren Mandanten in den [Amazon-EKS-Leitfadenn für bewährte Methoden](#). Allgemeine Informationen zur Integration AWS Private CA mit Kubernetes finden Sie unter [Sichern von Kubernetes mit AWS Private CA](#).

Verwaltung privater Zertifizierungsstellen

Mit können Sie eine vollständig AWS Private CA gehostete Hierarchie von Stamm- und untergeordneten Zertifizierungsstellen (CAs) zur internen Verwendung durch Ihre Organisation erstellen. Um den Zertifikatwiderruf zu verwalten, können Sie das Online Certificate Status Protocol (OCSP), Zertifikatssperrlisten (CRLs) oder beides aktivieren. AWS Private CA speichert und verwaltet Ihre CA-Zertifikate, CRLs und OCSP-Antworten. Die privaten Schlüssel für Ihre Root-Berechtigungen werden sicher von gespeichert AWS.

Note

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

Sie können AWS Private CA über die AWS Management Console, die AWS CLI und die AWS Private CA -API auf 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 der AWS Private CA-API \(Java-Beispiele\)](#).

Themen

- [Erstellen einer privaten Zertifizierungsstelle](#)
- [Erstellen und Installieren des CA-Zertifikats](#)
- [Steuern des Zugriffs auf eine private Zertifizierungsstelle](#)
- [Auflisten privater CAs](#)
- [Anzeigen einer privaten Zertifizierungsstelle](#)
- [Verwalten von Tags für Ihre private Zertifizierungsstelle](#)
- [Aktualisieren Ihrer privaten Zertifizierungsstelle](#)
- [Löschen Ihrer privaten Zertifizierungsstelle](#)
- [Wiederherstellen einer privaten Zertifizierungsstelle](#)

Erstellen einer privaten Zertifizierungsstelle

Sie können die Verfahren in diesem Abschnitt verwenden, um entweder CAs oder untergeordnete CAs zu erstellen, was zu einer überprüfbaren Hierarchie von Vertrauensbeziehungen führt, die Ihren organisatorischen Anforderungen entspricht. Sie können eine Zertifizierungsstelle mit der AWS Management Console, dem PCA-Teil der AWS CLI oder erstellen AWS CloudFormation.

Informationen zum Aktualisieren der Konfiguration einer bereits erstellten Zertifizierungsstelle finden Sie unter [Aktualisieren Ihrer privaten Zertifizierungsstelle](#).

Informationen zur Verwendung einer Zertifizierungsstelle zum Signieren von Endentitätszertifikaten für Ihre Benutzer, Geräte und Anwendungen finden Sie unter [Ausstellen privater Endentitätszertifikate](#).

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 den [AWS Preisrechner](#) auch verwenden, um die Kosten zu schätzen.

Themen

- [Verfahren zum Erstellen einer Zertifizierungsstelle \(Konsole\)](#)
- [Verfahren zum Erstellen einer Zertifizierungsstelle \(CLI\)](#)
- [Verwenden von AWS CloudFormation zum Erstellen einer Zertifizierungsstelle](#)

Verfahren zum Erstellen einer Zertifizierungsstelle (Konsole)

Führen Sie die folgenden Schritte aus, um eine private Zertifizierungsstelle mit der zu erstellen AWS Management Console.

So beginnen Sie mit der Verwendung der Konsole

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 Sie die Konsole in einer Region öffnen, in der Sie keine privaten CAs haben, wird die Einführungsseite angezeigt. Wählen Sie Private CA erstellen 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 CAs geöffnet. Wählen Sie CA erstellen aus.

Modusoptionen

Wählen Sie im Abschnitt Modusoptionen der Konsole den Ablaufmodus der Zertifikate aus, die Ihre CA ausstellt.

- Allgemeiner Zweck – 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 Widerrufsmechanismus ersetzen.

Optionen für den CA-Typ

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

- Wenn Sie Root auswählen, wird eine neue CA-Hierarchie eingerichtet. Diese CA wird durch ein selbstsigniertes Zertifikat gesichert. Sie dient als endgültige Signaturstelle für andere CAs und Endentitätszertifikate in der Hierarchie.
- Wenn Sie Unterordinate wählen, wird eine Zertifizierungsstelle erstellt, die von einer übergeordneten Zertifizierungsstelle darüber in der Hierarchie signiert werden muss. Untergeordnete CAs werden in der Regel verwendet, um andere untergeordnete CAs zu erstellen oder Endbenutzerzertifikate für Benutzer, Computer und Anwendungen auszustellen.

Note

AWS Private CA bietet einen automatisierten Signaturprozess, wenn die übergeordnete CA Ihrer untergeordneten Zertifizierungsstelle auch von gehostet wird AWS Private CA. Sie tun lediglich die zu verwendende übergeordnete CA.

Ihre untergeordnete Zertifizierungsstelle muss möglicherweise von einem externen Anbieter von Vertrauensdiensten signiert werden. In diesem Fall AWS Private CA stellt Ihnen eine Zertifikatsignierungsanforderung (CSR) bereit, die Sie herunterladen und

verwenden müssen, um ein signiertes CA-Zertifikat zu erhalten. Weitere Informationen finden Sie unter [Installieren eines untergeordneten CA-Zertifikats, das von einer externen übergeordneten CA signiert wurde](#).

Optionen für den definierten Namen des Betreffs

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

- Organisation (O) – Zum Beispiel ein Unternehmensname
- Organisationseinheit (OU) – Zum Beispiel eine Division innerhalb eines Unternehmens
- Ländername (C) – Eine zweistellige Landesvorwahl
- Name des Bundesstaats oder der Provinz – Vollständiger Name eines Bundesstaats oder einer Provinz
- Name des Standorts – Der Name einer Stadt
- Common Name (CN) – Eine für Menschen lesbare Zeichenfolge zur Identifizierung der CA.

Note

Sie können den Betreffnamen eines Zertifikats weiter anpassen, indem Sie zum Zeitpunkt der Ausstellung eine APIPassthrough-Vorlage anwenden. Weitere Informationen und ein detailliertes Beispiel finden Sie unter [Ein Zertifikat mit einem benutzerdefinierten Betreffnamen mithilfe einer APIPassthrough-Vorlage ausstellen](#).

Da das Sicherungszertifikat selbstsigniert ist, sind die Subjektinformationen, die Sie für eine private Zertifizierungsstelle bereitstellen, wahrscheinlich spärlicher als die, die eine öffentliche Zertifizierungsstelle enthalten würde. Weitere Informationen zu den einzelnen Werten, aus denen ein eindeutiger Subjektname besteht, finden Sie unter [RFC 5280](#).

Schlüsselalgorithmusoptionen

Wählen Sie unter Optionen für den Schlüsselalgorithmus den Schlüsselalgorithmus und die Bitgröße des Schlüssels aus. Der Standardwert ist ein RSA-Algorithmus mit einer Schlüssellänge von 2048 Bit. Sie können aus den folgenden Algorithmen wählen:

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

Optionen zum Widerrufen von Zertifikaten

Unter Optionen zum Widerrufen von Zertifikaten können Sie aus zwei Methoden zur Freigabe des Widerrufsstatus für Clients auswählen, die Ihre Zertifikate verwenden:

- Aktivieren der CRL-Verteilung
- Aktivieren von OCSP

Sie können entweder, keine oder beide dieser Widerrufsoptionen für Ihre CA konfigurieren. Obwohl optional, wird der verwaltete Widerruf als [bewährte Methode](#) empfohlen. Bevor Sie diesen Schritt ausführen, [Einrichtung einer Methode zum Widerruf von Zertifikaten](#) finden Sie unter Informationen zu den Vorteilen der einzelnen Methoden, zu der erforderlichen Vorabereinrichtung und zu zusätzlichen Widerrufsfunktionen.

Note

Wenn Sie Ihre CA erstellen, ohne den Widerruf zu konfigurieren, können Sie sie jederzeit später konfigurieren. Weitere Informationen finden Sie unter [Aktualisieren Ihrer privaten Zertifizierungsstelle](#).

So konfigurieren Sie eine CRL

1. Wählen Sie unter Optionen zum Widerrufen von Zertifikaten die Option CRL-Verteilung aktivieren aus.
2. Um einen Amazon S3-Bucket für Ihre CRL-Einträge zu erstellen, wählen Sie Neuen S3-Bucket erstellen und geben Sie einen eindeutigen Bucket-Namen ein. (Sie müssen den Pfad zum Bucket nicht einschließen.) Andernfalls wählen Sie unter S3-Bucket-URI einen vorhandenen Bucket aus der Liste aus.

Wenn Sie einen neuen Bucket über die Konsole erstellen, AWS Private CA versucht, die [erforderliche Zugriffsrichtlinie](#) an den Bucket anzuhängen und die S3-StandardEinstellung

Block Public Access (BPA) zu deaktivieren. Wenn Sie stattdessen einen vorhandenen Bucket angeben, müssen Sie sicherstellen, dass BPA für das Konto und für den Bucket deaktiviert ist. Andernfalls schlägt der Vorgang zum Erstellen der CA fehl. Wenn die CA erfolgreich erstellt wurde, müssen Sie ihr dennoch manuell eine Richtlinie anfügen, bevor Sie mit der Generierung von CRLs beginnen können. Verwenden Sie eines der unter beschriebenen Richtlinienmuster [Zugriffsrichtlinien für CRLs in Amazon S3](#). Weitere Informationen finden Sie unter [Hinzufügen einer Bucket-Richtlinie mit der Amazon S3-Konsole](#).

Important

Ein Versuch, eine CA mit der AWS Private CA Konsole zu erstellen, schlägt fehl, wenn alle der folgenden Bedingungen zutreffen:

- Sie richten eine CRL ein.
- Sie bitten AWS Private CA, automatisch einen S3-Bucket zu erstellen.
- Sie erzwingen BPA-Einstellungen in S3.

In diesem Fall erstellt die Konsole einen Bucket, versucht jedoch, ihn öffentlich zugänglich zu machen, und versucht nicht, ihn öffentlich zugänglich zu machen. Überprüfen Sie in diesem Fall Ihre Amazon S3-Einstellungen, deaktivieren Sie BPA nach Bedarf und wiederholen Sie dann das Verfahren zum Erstellen einer Zertifizierungsstelle. Weitere Informationen finden Sie unter [Blockieren des öffentlichen Zugriffs auf Ihren Amazon S3-Speicher](#).

3. Erweitern Sie CRL-Einstellungen für zusätzliche Konfigurationsoptionen.
 - Fügen Sie einen benutzerdefinierten CRL-Namen hinzu, um einen Alias für Ihren Amazon S3-Bucket zu erstellen. Dieser Name ist in Zertifikaten enthalten, die von der Zertifizierungsstelle in der Erweiterung „CRL Distribution Points“ ausgestellt wurden, die durch RFC 5280 definiert ist.
 - Geben Sie die Gültigkeit in Tagen ein, in denen Ihre CRL gültig bleibt. Der Standardwert lautet 7 Tage. Für Online-CRLs ist eine Gültigkeitsdauer von 2–7 Tagen üblich. AWS Private CA versucht, die CRL zum Mittelpunkt des angegebenen Zeitraums neu zu generieren.
4. Erweitern Sie S3-Einstellungen für die optionale Konfiguration von Bucket-Versioning und Bucket-Zugriffsprotokollierung.

So konfigurieren Sie OCSP

1. Wählen Sie unter Optionen zum Widerrufen von Zertifikaten die Option OCSP aktivieren aus.
2. Im Feld Benutzerdefinierter OCSP-Endpunkt – optional können Sie einen vollqualifizierten Domainnamen (FQDN) für einen Nicht-Amazon-OCSP-Endpunkt angeben.

Wenn Sie in diesem Feld einen FQDN angeben, AWS Private CA fügt den FQDN anstelle der Standard-URL für den AWS OCSP-Responder in die Erweiterung des Authority Information Access jedes ausgestellten Zertifikats ein. Wenn ein Endpunkt ein Zertifikat erhält, 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 Proxy-Server, um Datenverkehr, der bei Ihrem benutzerdefinierten FQDN ankommt, an den AWS OCSP-Responder weiterzuleiten.
- Fügen Sie Ihrer DNS-Datenbank einen entsprechenden CNAME-Datensatz hinzu.

Tip

Weitere Informationen zur Implementierung einer vollständigen OCSP-Lösung mit einem benutzerdefinierten CNAME finden Sie unter [Konfiguration einer benutzerdefinierten URL für OCSP AWS Private CA](#).

Hier ist beispielsweise ein CNAME-Datensatz für benutzerdefinierte OCSP, wie er in Amazon Route 53 angezeigt wird.

Datensatzname	Typ	Routing-Richtlinie	Unterscheidung	Bewerten/ Weiterleiten des Datenverkehrs an
alternati ve.example.com	CNAME	Einfach	-	proxy.exa mple.com

Note

Der Wert des CNAME darf kein Protokollpräfix wie „http://“ oder „https://“ enthalten.

Tags hinzufügen

Unter Tags hinzufügen können Sie Ihre CA optional markieren. 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 zu CAs nach der Erstellung finden Sie unter [Verwalten von Tags für Ihre private Zertifizierungsstelle](#).

Note

Um Tags während des Erstellungsprozesses an eine private Zertifizierungsstelle anzufügen, muss ein Zertifizierungsstellenadministrator zuerst eine Inline-IAM-Richtlinie mit der `CreateCertificateAuthority` Aktion verknüpfen und das Tagging explizit zulassen. Weitere Informationen finden Sie unter [Tag-on-create: Anhängen von Tags an eine Zertifizierungsstelle zum Zeitpunkt der Erstellung](#).

CA-Berechtigungsoptionen

Unter CA-Berechtigungsoptionen können Sie optional automatische Verlängerungsberechtigungen an den AWS Certificate Manager Service-Prinzipal delegieren. ACM kann private Endentitätszertifikate, die von dieser Zertifizierungsstelle generiert wurden, nur dann automatisch verlängern, wenn diese Berechtigung erteilt wurde. Sie können Erneuerungsberechtigungen jederzeit mit der AWS Private CA [CreatePermission](#) -API oder dem CLI-Befehl [create-permission](#) zuweisen.

Standardmäßig werden diese Berechtigungen aktiviert.

Note

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

Preisgestaltung

Vergewissern Sie sich unter Preise, dass Sie die Preise für eine private Zertifizierungsstelle verstehen.

Note

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

CA erstellen

Wählen Sie CA erstellen, nachdem Sie alle eingegebenen Informationen auf Genauigkeit überprüft haben. Die Detailseite für die Zertifizierungsstelle wird geöffnet und zeigt ihren Status als Ausstehendes Zertifikat an.

Note

Auf der Detailseite können Sie die Konfiguration Ihrer Zertifizierungsstelle abschließen, indem Sie Aktionen , CA-Zertifikat installieren auswählen, oder Sie können später zur Liste Private Zertifizierungsstellen zurückkehren und das Installationsverfahren abschließen, das in Ihrem Fall gilt:

- [Installieren eines Stammzertifizierungsstellenzertifikats](#)
- [Installieren eines untergeordneten CA-Zertifikats, das von gehostet wird AWS Private CA](#)
- [Installieren eines untergeordneten CA-Zertifikats, das von einer externen übergeordneten CA signiert wurde](#)

Verfahren zum Erstellen einer Zertifizierungsstelle (CLI)

Verwenden Sie den Befehl [create-certificate-authority](#), um eine private CA zu erstellen. Sie müssen die CA-Konfiguration (enthaltende Algorithmus- und Subjektnameninformationen), die Widerrufsconfiguration (wenn Sie OCSP und/oder eine CRL verwenden möchten) und den CA-Typ (Root oder untergeordneter Typ) angeben. Die Konfigurations- und Widerrufsconfigurationsdetails

sind in zwei Dateien enthalten, die Sie als Argumente für den Befehl angeben. Optional können Sie auch den CA-Nutzungsmodus konfigurieren (für die Ausstellung von Standard- oder kurzlebigen Zertifikaten), Tags anfügen und ein Idempotenz-Token bereitstellen.

Wenn Sie eine CRL konfigurieren, müssen Sie über einen gesicherten Amazon S3-Bucket verfügen, bevor Sie den `create-certificate-authority` Befehl ausführen. Weitere Informationen finden Sie unter [Zugriffsrichtlinien für CRLs in Amazon S3](#).

Die CA-Konfigurationsdatei gibt die folgenden Informationen an:

- 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 zum Signieren verwendet
- X.500-Themeninformationen

Die Widerrufskonfiguration 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 ist.

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

- Das `Enabled` Flag ist auf „true“ gesetzt.
- Der CRL-Ablaufzeitraum in Tagen (der Gültigkeitszeitraum der CRL).
- Der Amazon S3-Bucket, der die CRL enthält.
- (Optional) Ein [S3ObjectAcl](#)-Wert, der bestimmt, ob die CRL öffentlich zugänglich ist. Im hier gezeigten Beispiel ist der öffentliche Zugriff blockiert. Weitere Informationen finden Sie unter [Aktivieren von S3 Block Public Access \(BPA\) mit CloudFront](#).
- (Optional) Ein CNAME-Alias für den S3-Bucket, der in Zertifikaten enthalten ist, die von der CA ausgestellt wurden. Wenn die CRL nicht öffentlich zugänglich ist, verweist dies auf einen Verteilungsmechanismus wie Amazon CloudFront.
- (Optional) Ein `CrlDistributionPointExtensionConfiguration` Objekt mit den folgenden Informationen:

- Das `OmitExtension` Flag ist auf „true“ oder „false“ gesetzt. Dadurch wird gesteuert, ob der Standardwert für die CDP-Erweiterung in ein von der CA ausgestelltes Zertifikat geschrieben wird. Weitere Informationen zur CDP-Erweiterung finden Sie unter [Ermitteln des CRL Distribution Point \(CDP\) -URI](#). Ein `CustomCname` kann nicht festgelegt werden, wenn „true“ `OmitExtension` ist.

Note

Sie können beide Widerrufsmechanismen 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 für die Widerrufsvalidierung benötigen, finden Sie weitere Informationen unter [Aktualisieren einer CA \(CLI\)](#).

In den folgenden Beispielen wird davon ausgegangen, dass Sie Ihr `.aws` Konfigurationsverzeichnis mit einer gültigen Standardregion, einem Endpunkt und Anmeldeinformationen eingerichtet haben. Informationen zum Konfigurieren Ihrer AWS CLI Umgebung finden Sie unter [Einstellungen für Konfigurations- und Anmeldeinformationsdateien](#). Zur besseren Lesbarkeit stellen wir die CA-Konfiguration und die Widerrufseingabe in den Beispielbefehlen 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 einer CA mit aktiviertem OCSP

In diesem Beispiel aktiviert die Widerrufsdatei die standardmäßige OCSP-Unterstützung, die den AWS Private CA Responder verwendet, um den Zertifikatstatus 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 einer Zertifizierungsstelle mit aktiviertem OCSP und einem benutzerdefinierten CNAME

In diesem Beispiel ermöglicht die Widerrufsdatei eine angepasste OCSP-Unterstützung. Der `OcspCustomCname` Parameter verwendet einen vollqualifizierten Domainnamen (FQDN) als Wert.

Wenn Sie in diesem Feld einen FQDN angeben, AWS Private CA fügt den FQDN anstelle der Standard-URL für den AWS OCSP-Responder in die Erweiterung des Authority Information Access jedes ausgestellten Zertifikats ein. Wenn ein Endpunkt ein Zertifikat erhält, 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 Proxy-Server, um Datenverkehr, der bei Ihrem benutzerdefinierten FQDN ankommt, an den AWS OCSP-Responder weiterzuleiten.
- Fügen Sie Ihrer DNS-Datenbank einen entsprechenden CNAME-Datensatz hinzu.

i Tip

Weitere Informationen zur Implementierung einer vollständigen OCSP-Lösung mit einem benutzerdefinierten CNAME finden Sie unter [Konfiguration einer benutzerdefinierten URL für OCSP AWS Private CA](#).

Hier ist beispielsweise ein CNAME-Datensatz für benutzerdefinierte OCSP, wie er in Amazon Route 53 angezeigt wird.

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

i Note

Der Wert des CNAME darf kein Protokollpräfix wie „http://“ oder „https://“ enthalten.

Datei: revoke_config.txt für OCSP

```
{
  "OcsConfiguration":{
    "Enabled":true,
    "OcsCustomCname":"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 einer CA mit einer angehängten CRL

In diesem Beispiel definiert die Widerrufskonfiguration CRL-Parameter.

Datei: revoke_config.txt

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

Beispiel 4: Erstellen einer Zertifizierungsstelle mit einer angehängten CRL und einem aktivierten benutzerdefinierten CNAME

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

Datei: revoke_config.txt

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

```

Beispiel 5: Erstellen einer Zertifizierungsstelle und Angeben des Nutzungsmodus

In diesem Beispiel wird der CA-Nutzungsmodus beim Erstellen einer CA angegeben. Wenn nicht angegeben, ist der Verwendungsmodus-Parameter standardmäßig GENERAL_PURPOSE. In diesem Beispiel ist der Parameter auf SHORT_LIVED_CERTIFICATE gesetzt, was bedeutet, dass die Zertifizierungsstelle Zertifikate mit einer maximalen Gültigkeitsdauer von sieben Tagen ausstellt. In Situationen, in denen es schwierig ist, den Widerruf zu konfigurieren, läuft ein kurzlebiges Zertifikat, das kompromittiert wurde, im Rahmen des normalen Betriebs schnell ab. Folglich fehlt dieser Beispiel-CA ein Widerrufsmechanismus.

Note

AWS Private CA führt keine Gültigkeitsprüfungen für CA-Stammzertifikate 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-authority](#) Befehl in der , AWS CLI um Details zur resultierenden CA anzuzeigen, wie im folgenden Befehl gezeigt:

```

$ 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 einer CA für die Active-Directory-Anmeldung

Sie können eine private Zertifizierungsstelle erstellen, die für die Verwendung im Enterprise NTAAuth-Speicher von Microsoft Active Directory (AD) geeignet ist und wo sie Karten- oder Domaincontroller-Zertifikate ausstellen kann. Informationen zum Importieren eines CA-Zertifikats in AD finden Sie unter [Importieren von Zertifizierungsstellenzertifikaten \(CA\) von Drittanbietern in den Enterprise-NTAuth-Speicher](#).

Das Microsoft-[Certutil](#)-Tool kann verwendet werden, um CA-Zertifikate in AD zu veröffentlichen, indem die `-dspublish` Option aufgerufen wird. Ein in AD mit certutil veröffentlichtes Zertifikat ist in der gesamten Gesamtstruktur vertrauenswürdig. Mithilfe der Gruppenrichtlinie können Sie die Vertrauensstellung auch auf eine Teilmenge der gesamten Gesamtstruktur beschränken, z. B. auf eine einzelne Domain oder eine Gruppe von Computern in einer Domain. Damit die Anmeldung funktioniert, muss die ausstellende Zertifizierungsstelle auch im NTAAuth-Speicher veröffentlicht werden. Weitere Informationen finden Sie unter [Verteilen von Zertifikaten an Client-Computer mithilfe der Gruppenrichtlinie](#).

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 einer Boler-CA mit einer angehängten CRL und der CDP-Erweiterung, die in den ausgestellten Zertifikaten weggelassen wurde

Sie können eine private Zertifizierungsstelle erstellen, die für die Ausstellung von Zertifikaten für dener Smart Home-Standard geeignet ist. In diesem Beispiel `ca_config_PAA.txt` definiert die CA-Konfiguration in eine Boler Product Attestation Authority (PAA), wobei die Anbieter-ID (VID) auf FFF1 festgelegt ist.

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 Widerrufskonfiguration aktiviert CRLs und konfiguriert die CA so, dass die Standard-CDP-URL aus allen ausgestellten Zertifikaten weggelassen wird.

Datei: revoke_config.txt

```

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

Verwenden von AWS CloudFormation zum Erstellen einer Zertifizierungsstelle

Informationen zum Erstellen einer privaten Zertifizierungsstelle mit finden Sie AWS CloudFormation unter [AWS Private CA Ressourcentypreferenz](#) im AWS CloudFormation -Benutzerhandbuch.

Erstellen und Installieren 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:

- Installieren eines Zertifikats für eine Stammzertifizierungsstelle, die von gehostet wird 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 Konsolenverfahren beginnen auf der Konsolenseite Private CAs.

Kompatible Signaturalgorithmen

Die Unterstützung des Signaturalgorithmus für CA-Zertifikate hängt vom Signaturalgorithmus der übergeordneten CA und von der ab AWS-Region. Die folgenden Einschränkungen gelten sowohl für Konsolen- als auch für - AWS CLI Operationen.

- Eine übergeordnete Zertifizierungsstelle mit dem RSA-Signaturalgorithmus kann Zertifikate mit den folgenden Algorithmen ausstellen:
 - SHA256 RSA
 - SHA384 RSA
 - SHA512 RSA
- In einem Legacy-- kann AWS-Region eine übergeordnete Zertifizierungsstelle mit dem ECDSA-Signaturalgorithmus Zertifikate mit den folgenden Algorithmen ausstellen:
 - SHA256 ECDSA
 - SHA384 ECDSA
 - SHA512 ECDSA

Zu den Legacy-Versionen AWS-Regionen gehören:

Name der Region	Geografischer Standort
eu-north-1	Europa (Stockholm)
me-south-1	Naher Osten (Bahrain)
ap-south-1	Asien-Pazifik (Mumbai)

Name der Region	Geografischer Standort
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)
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 (USA-Ost)
us-gov-west-1	AWS GovCloud (USA West)
us-west-2	USA West (Oregon)

Name der Region	Geografischer Standort
us-west-1	USA West (Nordkalifornien)
ap-southeast-1	Asien-Pazifik (Singapur)
ap-southeast-2	Asien-Pazifik (Sydney)

- In einer nicht AWS-Region veralteten gelten die folgenden Regeln für EDCSA:
 - 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.

Installieren eines Stammzertifizierungsstellenzertifikats

Sie können ein Stammzertifizierungsstellenzertifikat über die AWS Management Console oder die installieren AWS CLI.

So erstellen und installieren Sie ein Zertifikat für Ihre private Stammzertifizierungsstelle (Konsole)

1. (Optional) Wenn Sie noch nicht auf der Detailseite der Zertifizierungsstelle sind, öffnen Sie die - AWS Private CA Konsole unter <https://console.aws.amazon.com/acm-pca/home>. 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 Stammzertifizierungsstellenzertifikat installieren zu öffnen.
3. Geben Sie unter Angeben der Parameter des CA-Stammzertifikats die folgenden Zertifikatparameter an:
 - Gültigkeit – Gibt das Ablaufdatum und die Ablaufzeit für das CA-Zertifikat an. Die AWS Private CA Standardgültigkeitsdauer für ein Stammzertifizierungsstellenzertifikat beträgt 10 Jahre.
 - Signaturalgorithmus – Gibt den Signaturalgorithmus an, der verwendet werden soll, wenn die Stammzertifizierungsstelle neue Zertifikate ausstellt. Die verfügbaren Optionen variieren je nach der AWS-Region , in der Sie die CA erstellen. Weitere Informationen finden Sie

unter [Kompatible Signaturalgorithmen](#), [Unterstützte kryptografische Algorithmen](#) und SigningAlgorithm in [CertificateAuthorityConfiguration](#).

- SHA256 RSA
- SHA384 RSA
- SHA512 RSA

Überprüfen Sie Ihre Einstellungen auf Richtigkeit, wählen Sie Bestätigen und installieren.

AWS Private CA exportiert eine CSR für Ihre CA, generiert ein Zertifikat mit einer [Stammzertifizierungsstellenzertifikatsvorlage](#) und signiert das Zertifikat selbst. importiert AWS Private CA dann das selbstsignierte Stammzertifizierungsstellenzertifikat.

4. Auf der Detailseite für die CA wird oben der Status der Installation (Erfolg oder Fehler) angezeigt. Wenn die Installation erfolgreich war, zeigt die neu abgeschlossene Stammzertifizierungsstelle im Bereich Allgemein den Status Aktiv an.

So erstellen und installieren Sie ein Zertifikat für Ihre private Stammzertifizierungsstelle (AWS CLI)

1. Generieren Sie eine Zertifikatsignierungsanforderung (Certificate Signing Request, 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 Datei `ca.csr`, eine PEM-Datei, die im base64-Format codiert ist, sieht wie folgt aus.

```
-----BEGIN CERTIFICATE REQUEST-----
MIIC1DCCAwwCAQAwTELMAkGA1UEBhMCVVMxFTATBgNVBAoMDEV4YW1wbGUgQ29y
cDE0MAwGA1UECwwFU2FsZXNxCzAJBgNVBAGMA1dBMRgwFgYDVQDDA93d3cuZXhh
bXBsZS5jb20xEDA0BgNVBACMB1NlYXR0bGUwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQQDD+7eQChWU02m6pHs1I7AVSFkVwbQofKIHvbvy7wm8V09/BuI7
LE/jrnd1jGoyI7jaMHKXPtEP3uNlCzv+oEza070jgjqPZVehtA6a3/3vdQ1qCoD2
rXpv6VIzCq2onx2X7m+Zixwn2oY111ELXP7I5g0GmUStymq+pY5VARPy3vTRMjgC
JEiz8w7VvC15uIsHFAWa2/NvKyndQMPaCNft238wesV5s2cX0US173jghISHg99o
ymf0TRUgvAGQMCXvsW07M1P5VDmBU7k/AZ9ExsUfMe20B++fhfQW1r2N7/1pC4+DP
qJTFXTEexLfRRTLeLuGEaJL+c6fMyG+Yk53tZAgMBAAGgIjAgBgkqhkiG9w0BCQ4x
EzARMA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEBA7xxLVI5s1B
```

```
qmXMMT44y1DZtQx3RDPanMNGLG01TmLtyqqnUH49T1a+2p7nr10tojUf/3PaZ52F
QN09SrFk8qtYSKnMGd5PZL0A+NFsNW+w4BAQNK1g9m617YEsnkztbfKRloaJNYoA
HZaRvbA01MQ/tU2PKZR2vnao444Ugm00/t3jx5rj817b31hQcHHQ01QuXV2kyTrM
ohWeLf2fL+K0xJ9ZgXD4KYnY0zarpreA5RBe05xs3Ms+oGwC13qQfMBx33vrrz2m
dw5iKjg71uuUUmtDV6ewwGa/V05hNinYAfogdu5aGuVbnTFT3n45B8WHz2+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 ähnlich der folgenden.

```
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 geben 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 Base64-encoded Daten korrekt AWS Private CA analysiert.

```

$ 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 PEM-Datei, die im base64-Format codiert ist, sieht wie folgt aus.

```
-----BEGIN CERTIFICATE-----
MIIDpzCCAo+gAwIBAgIRAIu0ar1QET1UQE0ZJGZYdIwDQYJKoZIhvcNAQELBQAw
bTElMAkGA1UEBhMCVVMxFTATBgNVBAoMDEV4YW1wbGUgQ29ycDE0MAwGA1UECwwF
U2FsZXNxCzAJBgNVBAGMA1dBMRGwFgYDVQDDA93d3cuZXhhbXBsZS5jb20xEDA0
BgNVBACMB1NlYXR0bGUwHhcNMjEwMzA4MTU0NjI3WWhcNMjEwMzA4MTY0NjI3WjBt
MQswCQYDVQQGEwJVUzEVMBMGA1UECgwMRXhhbXBsZS5SBDB3JwMQ4wDAYDVQQLEDAVT
YXlzcjE1MAkGA1UECAwCV0ExGDAWBgNVBAMMD3d3dy5leGFtcGx1LmNvbTEQMA4G
A1UEBwwHU2VhdHRsZTCCASiWdQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMP7
t5AKFZQ7abqkeyUjsBVIWRa9tCh8oge9u/LvCbxU738G4jssT+0ud3WMajIjuNow
cpc+0Q/e42UL0/6gTNRts60C0o91V6G0Dprf/e91DwoKgPatem/pUjNyraifHZfu
b5mLHCfahjWXUQtC/sjmDQaZRK3Kar6ljlUBE/Le9NEy0AikSLPzDtW8LXm4iwcU
BZrb828rKd1Aw9oI1+3bfzB6xXmzZxc5RLXve0CEhKGD32jKZ/RNFSC8AZAwJe+x
bTsys/1U0YFTuT8Bn0TGxR8x7Y4H75+F9BavY3v+WkLj4M+o1N9dMR7Et9FMt4u4
YRokv5zpz8zIb5iTne1kCAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4E
FgQUaW3+r328uTLokog2Tk1moBK+yt4wDgYDVR0PAQH/BAQDAgGGMA0GCSqGSIb3
DQEBcWUAA4IBAQAQjd/7UZ8RDE+PLWSDNGQdLem0BTcawF+tK+PzA4Ev1mn9VuNc
g+x3oZvVZSDQBANuz0b9oPeo54aE38dW1zQm2qfTab8822aqeWMLyJ1dMsAgqYX2
t9+u6w3NzRCw8Pvz18V69+dFE5AeXmNP0Z5/gdz8H/NSpctj1zopbScRZKCS1Pid
Rf3Z0Pm9QP92YpWyYdkfAU04xdDo1vR0MYjKPk14LjRqSU/tcCJnPmbJiwq+bWpX
2WJoEBXB/p15Kn6JxjI0ze2SnSI48JZ8it4fvxrh0o0VoLNIuCuNXJ0wU17Rd11W
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 ähnlich der folgenden.

```
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 Stammzertifizierungsstellenzertifikat, um es auf der Zertifizierungsstelle 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 Base64-encoded Daten korrekt AWS Private CA analysiert.

```
$ 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": "DOC-EXAMPLE-BUCKET1"
    },
    "OcspConfiguration": {
      "Enabled": false
    }
  }
}
}

```

Installieren eines untergeordneten CA-Zertifikats, das von gehostet wird AWS Private CA

Sie können die verwenden AWS Management Console , um ein Zertifikat für Ihre AWS Private CA gehostete untergeordnete CA zu erstellen und zu installieren.

So erstellen und installieren Sie ein Zertifikat für Ihre AWS Private CA gehostete untergeordnete CA

1. (Optional) Wenn Sie noch nicht auf der Detailseite der Zertifizierungsstelle sind, öffnen Sie die - AWS Private CA Konsole unter <https://console.aws.amazon.com/acm-pca/home>. 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 Untergeordnetes CA-Zertifikat installieren unter CA-Typ auswählen aus, ob Sie ein Zertifikat installieren AWS Private CA möchten, 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 gefiltert, um CAs anzuzeigen, die die folgenden Kriterien erfüllen:
 - Sie sind berechtigt, die CA zu verwenden.
 - Die Zertifizierungsstelle würde sich nicht selbst signieren.
 - Die CA befindet sich im Status ACTIVE.
 - Der CA-Modus ist GENERAL_PURPOSE.
5. Geben Sie unter Angeben der Parameter des untergeordneten CA-Zertifikats die folgenden Zertifikatparameter an:
 - Gültigkeit – Gibt das Ablaufdatum und die Ablaufzeit 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 mindestens Eins bedeutet, dass die untergeordnete CA Zertifikate ausstellt, um zusätzliche CAs zu erstellen, die ihr untergeordnet sind.

- 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 API-[IssueCertificate](#)Aktion erstellen, müssen Sie den ARN manuell angeben. Informationen zu verfügbaren CA-Zertifikatvorlagen finden Sie unter [Grundlegendes zu Zertifikatsvorlagen](#).
6. Überprüfen Sie Ihre Einstellungen auf Richtigkeit, wählen Sie Bestätigen und installieren. AWS Private CA exportiert eine CSR, generiert ein Zertifikat mit einer untergeordneten CA-[Zertifikatsvorlage](#) und signiert es mit der ausgewählten übergeordneten CA. importiert AWS Private CA dann das signierte untergeordnete CA-Zertifikat.
 7. Auf der Detailseite für die CA wird oben der Status der Installation (Erfolg oder Fehler) angezeigt. Wenn die Installation erfolgreich war, zeigt die neu abgeschlossene untergeordnete Zertifizierungsstelle im Bereich Allgemein den Status Aktiv an.

Installieren eines untergeordneten CA-Zertifikats, das von einer externen übergeordneten CA signiert wurde

Nachdem Sie eine untergeordnete private Zertifizierungsstelle wie unter [Verfahren zum Erstellen einer Zertifizierungsstelle \(Konsole\)](#) oder beschrieben erstellt haben [Verfahren zum Erstellen einer Zertifizierungsstelle \(CLI\)](#), haben Sie die Möglichkeit, sie zu aktivieren, indem Sie ein CA-Zertifikat installieren, das von einer externen Signaturstelle signiert wurde. Das Signieren Ihres untergeordneten CA-Zertifikats bei einer externen Zertifizierungsstelle erfordert, dass Sie zuerst einen externen Vertrauensdienstanbieter als Signaturautorität einrichten oder die Verwendung eines Drittanbieters anordnen.

Note

Verfahren zum Erstellen oder Abrufen 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. Fordern Sie eine Zertifikatsignierungsanforderung (CSR) von an AWS Private CA.
2. Senden Sie die CSR an Ihre externe Signaturstelle und fordern Sie ein signiertes CA-Zertifikat zusammen mit allen Kettenzertifikaten an.

3. Importieren Sie das CA-Zertifikat und die Kette in AWS Private CA , um Ihre untergeordnete CA zu aktivieren.

Die detaillierten Schritte finden Sie unter [Extern signierte private CA-Zertifikate](#) .

Steuern des Zugriffs auf eine private Zertifizierungsstelle

Jeder Benutzer mit den erforderlichen Berechtigungen für eine private Zertifizierungsstelle von AWS Private CA kann diese Zertifizierungsstelle verwenden, um andere Zertifikate zu signieren. Der CA-Besitzer 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-Konto. Ein Benutzer, der sich in einem anderen AWS Konto befindet, kann Zertifikate auch ausstellen, wenn er vom CA-Besitzer über eine [ressourcenbasierte Richtlinie](#) autorisiert wird.

Autorisierte Benutzer, ob Einzelkonto oder kontoübergreifend, können bei der Ausstellung von Zertifikaten - AWS Private CA oder - AWS Certificate Manager Ressourcen verwenden. Zertifikate, die von der AWS Private CA [IssueCertificate](#) API oder dem CLI-Befehl [issue-certificate](#) ausgestellt wurden, werden nicht verwaltet. Solche Zertifikate erfordern eine manuelle Installation auf Zielgeräten und eine manuelle Verlängerung, wenn sie ablaufen. Zertifikate, die von der ACM-Konsole, der ACM [RequestCertificate](#)-API oder dem CLI-Befehl [request-certificate](#) ausgestellt wurden, werden verwaltet. Solche Zertifikate können einfach in Services installiert werden, die in ACM integriert sind. Wenn der CA-Administrator dies zulässt und das Konto des Ausstellers über eine [serviceverknüpfte Rolle](#) für ACM verfügt, werden verwaltete Zertifikate automatisch verlängert, wenn sie ablaufen.

Themen

- [Erstellen von Einzelkontoberechtigungen für einen IAM-Benutzer](#)
- [Anfügen einer Richtlinie für den kontoübergreifenden Zugriff](#)

Erstellen von Einzelkontoberechtigungen für einen IAM-Benutzer

Wenn sich der CA-Administrator (d. h. der Eigentümer der CA) und der Zertifikataussteller in einem einzigen AWS Konto befinden, 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 Berechtigungen 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: Ausstellen eines nicht verwalteten Zertifikats

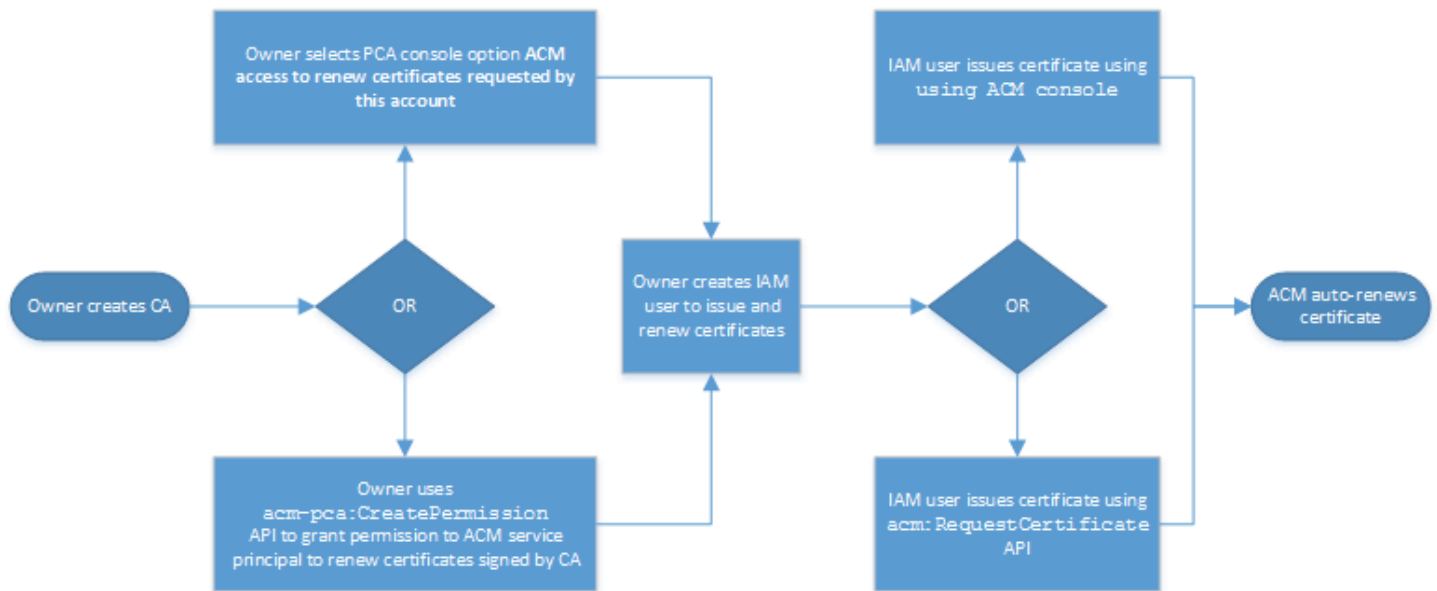
In diesem Fall erstellt der Kontoinhaber eine private Zertifizierungsstelle und erstellt dann einen IAM-Benutzer mit der Berechtigung zum Ausstellen von Zertifikaten, die von der privaten Zertifizierungsstelle signiert wurden. Der IAM-Benutzer gibt ein Zertifikat aus, indem er die AWS Private CA `IssueCertificate`-API 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 auch manuell erneuert werden, wenn sie ablaufen. Die Ausstellung eines Zertifikats mit dieser API erfordert eine Zertifikatsignierungsanforderung (CSR) und ein Schlüsselpaar, die außerhalb von AWS Private CA von [OpenSSL](https://docs.aws.amazon.com/openssl/) oder einem ähnlichen Programm generiert werden. Weitere Informationen finden Sie in der `IssueCertificate`-Dokumentation. https://docs.aws.amazon.com/privateca/latest/APIReference/API_IssueCertificate.html

Fall 2 für ein einzelnes Konto: Ausstellen eines verwalteten Zertifikats über ACM

Dieser zweite Fall umfasst API-Operationen von ACM und PCA. Der Kontoinhaber erstellt wie zuvor eine private Zertifizierungsstelle und einen IAM-Benutzer. Der Kontoinhaber [erteilt dann dem ACM-Service-Prinzipal die Berechtigung](#), automatisch alle Zertifikate zu erneuern, die von dieser Zertifizierungsstelle signiert wurden. Der IAM-Benutzer gibt das Zertifikat erneut aus, diesmal jedoch durch Aufrufen der `ACMRequestCertificate`-API, die CSR und Schlüsselgenerierung übernimmt. Wenn das Zertifikat abläuft, automatisiert ACM den Erneuerungsworkflow.



Der Kontoinhaber hat die Möglichkeit, während oder nach der Erstellung der CA oder mithilfe der PCA-CreatePermissionAPI über die Managementkonsole eine Verlängerungsberechtigung zu erteilen. Die aus diesem Workflow erstellten verwalteten Zertifikate stehen für die Verwendung auf mit - AWS Services zur Verfügung, die in ACM integriert sind.

Der folgende Abschnitt enthält Verfahren zum Erteilen von Erneuerungsberechtigungen.

Zuweisen von Zertifikatserneuerungsberechtigungen zu ACM

Mit der [verwalteten Erneuerung](#) 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 der ACM-Serviceprinzipal alle möglichen Berechtigungen von der Zertifizierungsstelle selbst erhalten. Wenn diese Erneuerungsberechtigungen für ACM nicht vorhanden sind, muss der Eigentümer der Zertifizierungsstelle (oder ein autorisierter Stellvertreter) jedes private Zertifikat manuell neu ausstellen, wenn es abläuft.

⚠ Wichtig

Diese Verfahren zum Zuweisen von Erneuerungsberechtigungen gelten nur, wenn sich der CA-Besitzer und der Zertifikataussteller im selben AWS Konto befinden. Kontoübergreifende Szenarien finden Sie unter [Anfügen einer Richtlinie für den kontoübergreifenden Zugriff](#).

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 private CA-Berechtigungen zu (Konsole)

1. 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>.
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, um von diesem Konto angeforderte Zertifikate zu verlängern.
5. Wählen Sie Speichern.

So verwalten Sie ACM-Berechtigungen in AWS Private CA (AWS CLI)

Verwenden Sie den Befehl [create-permission](#), um ACM Berechtigungen zuzuweisen. Sie müssen die erforderlichen Berechtigungen (IssueCertificate, und ListPermissions) zuweisenGetCertificate, 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 Berechtigungen zu widerrufen, die einer Zertifizierungsstelle einem - AWS Service-Prinzipal zugewiesen wurden.

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

Anfügen einer Richtlinie für den kontoübergreifenden Zugriff

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

Ein CA-Administrator kann Richtlinien wie folgt anfügen und verwalten:

- Verwenden Sie in der -Managementkonsole AWS Resource Access Manager (RAM), eine Standardmethode für die gemeinsame Nutzung von AWS Ressourcen über -Konten hinweg. Wenn Sie eine CA-Ressource in AWS RAM für einen Prinzipal in einem anderen Konto freigeben, wird die erforderliche ressourcenbasierte Richtlinie automatisch an die CA angehängt. Weitere Informationen zu RAM finden Sie im [AWS RAM -Benutzerhandbuch](#).

Note

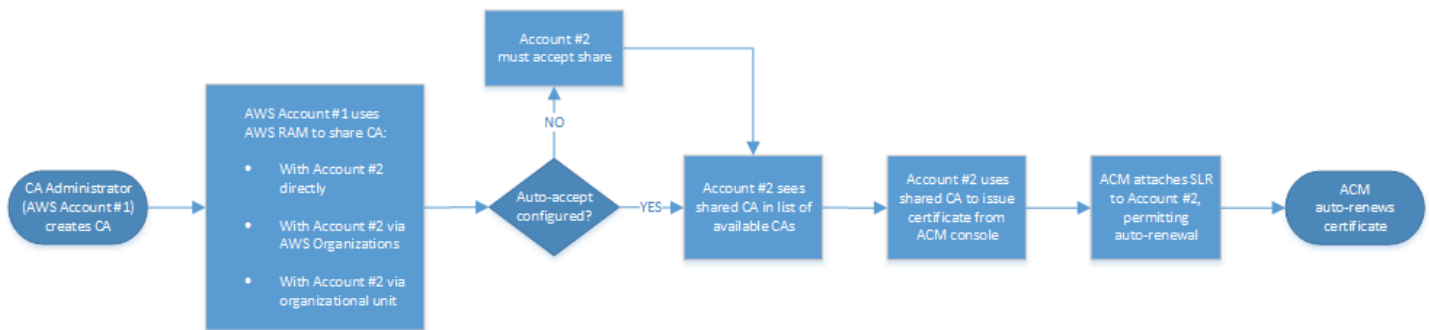
Sie können die RAM-Konsole einfach öffnen, indem Sie eine CA und dann Aktionen, Ressourcenfreigaben verwalten auswählen.

- Programmgesteuert unter Verwendung der PCA-APIs [PutPolicyGetPolicy](#), und [DeletePolicy](#).
- Manuelles Verwenden der PCA-Befehle [put-policy](#) , [get-policy](#) und [delete-policy](#) in der AWS CLI.

Nur die Konsolenmethode erfordert RAM-Zugriff.

Kontoübergreifender Fall 1: Ausstellen eines verwalteten Zertifikats über die Konsole

In diesem Fall verwendet der CA-Administrator AWS Resource Access Manager (AWS RAM), um den CA-Zugriff für ein anderes AWS Konto freizugeben, wodurch dieses Konto verwaltete ACM-Zertifikate ausstellen kann. Das Diagramm zeigt, dass die CA direkt mit dem Konto oder indirekt über eine AWS Organizations ID teilen AWS RAM kann, in der das Konto Mitglied ist.



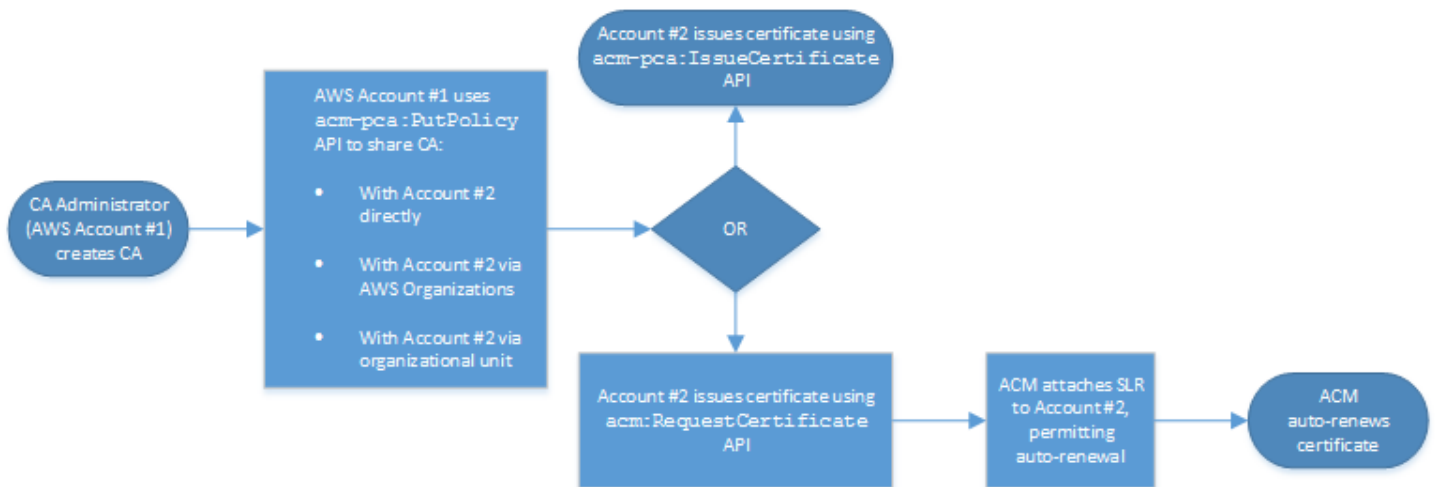
Nachdem RAM eine Ressource über freigegeben hat AWS Organizations, muss der Empfängerprinzipal die Ressource akzeptieren, damit sie wirksam wird. Der Empfänger kann so konfigurieren AWS Organizations , dass angebotene Freigaben automatisch akzeptiert werden.

Note

Das Empfängerkonto ist für die Konfiguration der Autorisierung in ACM verantwortlich. Wenn eine gemeinsam genutzte Zertifizierungsstelle zum ersten Mal verwendet wird, installiert ACM in der Regel eine serviceverknüpfte Rolle, die es ihr ermöglicht, unbeaufsichtigte Zertifikatsaufrufe auf durchzuführen AWS Private CA. Wenn dies fehlschlägt (in der Regel aufgrund einer fehlenden Berechtigung), werden Zertifikate von der CA nicht automatisch erneuert. Nur der ACM-Benutzer kann das Problem lösen, nicht der CA-Administrator. Weitere Informationen finden Sie unter [Verwenden einer serviceverknüpften Rolle \(SLR\) mit ACM](#).

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

Dieser zweite Fall zeigt die Freigabe- und Ausstellungsoptionen, die mit der AWS Certificate Manager und der AWS Private CA API möglich sind. Alle 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 zwei verschiedene API-Operationen zur Ausstellung eines Zertifikats. 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 [RequestCertificate](#) führt zu einem verwalteten Zertifikat, das einfach auf integrierten ACM-Services installiert werden kann und automatisch verlängert wird.

Note


Das Empfängerkonto ist für die Konfiguration der automatischen Verlängerung in ACM verantwortlich. Wenn eine gemeinsam genutzte Zertifizierungsstelle zum ersten Mal verwendet wird, installiert ACM in der Regel eine serviceverknüpfte Rolle, die es ihr ermöglicht, unbeaufsichtigte Zertifikatsaufrufe auf durchzuführen AWS Private CA. Wenn dies fehlschlägt (in der Regel aufgrund einer fehlenden Berechtigung), werden Zertifikate von der CA nicht automatisch verlängert, und nur der ACM-Benutzer kann das Problem lösen, nicht der CA-Administrator. Weitere Informationen finden Sie unter [Verwenden einer serviceverknüpften Rolle \(SLR\) mit ACM](#).

Auflisten privater CAs

Sie können die AWS Private CA Konsole oder verwenden AWS CLI , um private CAs aufzulisten, die Sie besitzen oder auf die Sie Zugriff haben.

So listen Sie verfügbare CAs mithilfe der Konsole auf

1. 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>.
2. Überprüfen Sie die Informationen in der Liste Private Zertifizierungsstellen. Sie können mithilfe der Seitenzahlen oben rechts durch mehrere Seiten von CAs navigieren. Jede Zertifizierungsstelle belegt eine Zeile mit einigen oder allen folgenden Spalten, die für jede Spalte angezeigt werden:
 - **Betreff** – Zusammenfassung der Informationen zu definierten Namen für die CA.
 - **ID** – Hexadezimale eindeutige 32-Byte-ID der CA.
 - **Status** – CA-Status. Mögliche Werte sind Erstellen, Ausstehendes Zertifikat, Aktiv, Gelöscht, Deaktiviert, Abgelaufen und Fehlgeschlagen.
 - **Typ** – Der Typ der Zertifizierungsstelle. Mögliche Werte sind Root und Subordinate .
 - **Modus** – Der Modus der Zertifizierungsstelle. Mögliche Werte sind Allzweck (ausgestellte Zertifikate, die mit einem beliebigen Ablaufdatum konfiguriert werden können) und Kurzlebige Zertifikat (ausgestellte Zertifikate mit einer maximalen Gültigkeitsdauer von sieben Tagen). Eine kurze Gültigkeitsdauer kann in einigen Fällen einen Widerrufsmechanismus ersetzen. Der Standardwert ist Allzweck .
 - **Eigentümer** – Das AWS Konto, dem die CA gehört. Dies kann Ihr Konto oder ein Konto sein, das über delegierte CA-Verwaltungsberechtigungen für Sie verfügt.
 - **Schlüsselalgorithmus** – Der von der CA unterstützte öffentliche Schlüsselalgorithmus. Mögliche Werte sind RSA_2048, RSA_4096, EC_prime256v1 und EC_secp384r1.
 - **Signieralgorithmus** – Der Algorithmus, den die Zertifizierungsstelle zum Signieren von Zertifikatsanforderungen verwendet. (Nicht zu verwechseln mit dem `SigningAlgorithm` Parameter, der zum Signieren von Zertifikaten verwendet wird, wenn sie ausgestellt werden.) Mögliche Werte sind SHA256WITHECDSA, SHA384WITHECDSA, SHA512WITHECDSA, SHA256WITHRSA , SHA384WITHRSA und SHA512WITHRSA .

 Note

Sie können die anzuzeigenden Spalten sowie andere Einstellungen anpassen, indem Sie das Einstellungssymbol in der oberen rechten Ecke der Konsole auswählen.

So listen Sie verfügbare CAs mithilfe der auf AWS CLI

Verwenden Sie den [list-certificate-authorities](#) Befehl , um verfügbare CAs aufzulisten, 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
        }
      }
    }
  ]
  ...
}
```

Anzeigen einer privaten Zertifizierungsstelle

Sie können die ACM-Konsole oder die verwenden AWS CLI , um detaillierte Metadaten zu einer privaten Zertifizierungsstelle anzuzeigen und mehrere der Werte nach Bedarf zu ändern.

Ausführliche Informationen zum Aktualisieren von CAs finden Sie unter [Aktualisieren Ihrer privaten Zertifizierungsstelle](#).

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 unter <https://console.aws.amazon.com/acm-pca/home>.
2. Überprüfen Sie die Liste Private Zertifizierungsstellen. Sie können mithilfe der Seitenzahlen oben rechts durch mehrere Seiten von CAs navigieren.
3. Um detaillierte Metadaten für eine aufgeführte Zertifizierungsstelle anzuzeigen, wählen Sie das Optionsfeld der Zertifizierungsstelle aus, die Sie überprüfen möchten. Dadurch wird ein Detailbereich mit den folgenden tabellarischen Ansichten geöffnet:
 - Registerkarte **Betreff** – Informationen über den definierten Namen für die CA. Weitere Informationen finden Sie unter [Optionen für den definierten Namen des Betreffs](#). Zu den angezeigten Feldern gehören:
 - **Betreff** – Zusammenfassung der Felder für bereitgestellte Namensinformationen
 - **Organisation (O)** – Zum Beispiel ein Unternehmensname
 - **Organisationseinheit (OU)** – Zum Beispiel eine Division innerhalb eines Unternehmens
 - **Ländername (C)** – Eine zweistellige Landesvorwahl
 - **Name des Bundesstaats oder der Provinz** – Vollständiger Name eines Bundesstaats oder einer Provinz
 - **Name des Standorts** – Der Name einer Stadt
 - **Common Name (CN)** – Eine für Menschen lesbare Zeichenfolge zur Identifizierung der CA.
 - Registerkarte **CA-Zertifikat** – Informationen über die Gültigkeit des CA-Zertifikats
 - **Gültig bis** – Das Datum und die Uhrzeit, bis das CA-Zertifikat gültig ist
 - **Läuft ab in** – Die Anzahl der Tage bis zum Ablauf
 - Registerkarte **„Konfiguration widerrufen“** – Ihre aktuelle Auswahl an Optionen zum Widerrufen von Zertifikaten. Wählen Sie **Zu aktualisierende Bearbeiten** aus.
 - **Verteilung der Zertifikatsperrliste (Certificate Revocation List, CRL)** – Status von **Aktiviert** oder **Deaktiviert**
 - **Online Certificate Status Protocol (OCSP)** – Status von **Aktiviert** oder **Deaktiviert**

- Registerkarte „Berechtigungen“ – Ihre aktuelle Auswahl von Zertifikaterneuerungsberechtigungen für diese CA über AWS Certificate Manager (ACM). Wählen Sie zum Aktualisieren Bearbeiten aus.
 - ACM-Autorisierung für Erneuerungen – Status von autorisiert oder nicht autorisiert
 - Registerkarte Tags – Ihre aktuelle Zuweisung anpassbarer Labels für diese Zertifizierungsstelle. Wählen Sie die zu aktualisierende Tags verwalten aus.
 - Registerkarte Ressourcenfreigaben – Ihre aktuelle Zuweisung von Ressourcenfreigaben für diese CA über 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. Die hexadezimale eindeutige 32-Byte-ID der CA wird oben angezeigt. Der Bereich enthält die folgenden zusätzlichen Informationen:
- Status – CA-Status. Mögliche Werte sind Erstellen , Ausstehendes Zertifikat , Aktiv , Gelöscht , Deaktiviert , Abgelaufen und Fehlgeschlagen .
 - ARN – Der [Amazon-Ressourcenname](#) für die CA.
 - Eigentümer – Das AWS Konto, dem die CA gehört. Dies kann Ihr -Konto (selbst) oder ein Konto sein, das über delegierte CA-Verwaltungsberechtigungen für Sie verfügt.
 - CA-Typ – Der Typ der CA. Mögliche Werte sind Root und Subordinate .
 - Erstellt am – Das Datum und die Uhrzeit, zu der die CA erstellt wurde.
 - Ablaufdatum – Das Datum und die Uhrzeit, zu der das CA-Zertifikat abläuft.
 - Modus – Der Modus der Zertifizierungsstelle. Mögliche Werte sind Allzweck (Zertifikate, die mit einem beliebigen Ablaufdatum konfiguriert werden können) und Kurzlebige Zertifikat (Zertifikate mit einer maximalen Gültigkeitsdauer von sieben Tagen). Eine kurze Gültigkeitsdauer kann in einigen Fällen einen Widerrufsmechanismus ersetzen. Der Standardwert ist Allzweck .
 - Schlüsselalgorithmus – Der von der CA unterstützte öffentliche Schlüsselalgorithmus. Mögliche Werte sind RSA 2048, RSA 4096, ECDSA P2567 und ECDSA P384.
 - Signieralgorithmus – Der Algorithmus, den die Zertifizierungsstelle zum Signieren von Zertifikatsanforderungen verwendet. (Nicht zu verwechseln mit dem SigningAlgorithm Parameter, der zum Signieren von Zertifikaten verwendet wird, wenn sie ausgestellt werden.)

Mögliche Werte sind SHA256 ECDSA , SHA384 ECDSA , SHA512 ECDSA , SHA256 RSA , SHA384 RSA und SHA512 RSA

- Sicherheitsstandard für Schlüsselspeicher – Grad der Einhaltung der Federal Information Processing Standards. Mögliche Werte sind FIPS 140-2 Level 3 oder höher und FIPS 140-2 Level 3 oder höher. Dieser Parameter variiert je nach AWS Region.

So zeigen Sie CA-Details mit der an und ändern sie AWS CLI

Verwenden Sie den [describe-certificate-authority](#) Befehl in der AWS CLI , um Details zu einer CA anzuzeigen, wie im folgenden Befehl gezeigt:

```
$ 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
      }
    }
  }
}
```

Informationen zum Aktualisieren einer privaten Zertifizierungsstelle über die Befehlszeile finden Sie unter [Aktualisieren einer CA \(CLI\)](#).

Verwalten von Tags für Ihre private Zertifizierungsstelle

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 Tags für private CAs hinzuzufügen, anzuzeigen oder zu entfernen.

Sie können jederzeit benutzerdefinierte Tags für Ihre private Zertifizierungsstelle hinzufügen oder entfernen. Sie können beispielsweise private CAs mit Schlüssel-Wert-Paaren wie `Environment=Prod` oder markieren, `Environment=Beta` um zu identifizieren, für welche Umgebung die Zertifizierungsstelle vorgesehen ist. Weitere Informationen finden Sie unter [Erstellen einer privaten Zertifizierungsstelle](#).

Note

Um Tags während des Erstellungsprozesses an eine private Zertifizierungsstelle anzufügen, muss ein Zertifizierungsstellenadministrator zuerst eine Inline-IAM-Richtlinie mit der `CreateCertificateAuthority` Aktion verknüpfen und das Tagging explizit zulassen. Weitere Informationen finden Sie unter [Tag-on-create: Anhängen von Tags an eine Zertifizierungsstelle zum Zeitpunkt der Erstellung](#).

Andere AWS Ressourcen unterstützen auch das Markieren. Sie können verschiedenen Ressourcen dasselbe Tag zuweisen, um anzugeben, dass diese Ressourcen verwandt sind. Sie können beispielsweise Ihrer `Website=example.com` CA, dem Elastic Load Balancing Load Balancer und anderen zugehörigen Ressourcen ein Tag wie zuweisen. Weitere Informationen zum Markieren von AWS Ressourcen finden Sie unter [Markieren Ihrer Amazon EC2-Ressourcen](#) im [Amazon EC2-Benutzerhandbuch für Linux-Instances](#).

Die folgenden grundlegenden Einschränkungen gelten für AWS Private CA Tags:

- 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 `aws:` beginnen und nicht auf Ihr tags-per-resource Kontingent angerechnet werden, werden `rds:` werden.
- Wenn Sie Ihr Markierungsschema für mehrere Services und Ressourcen verwenden möchten, denken Sie daran, dass andere Services möglicherweise unterschiedliche Einschränkungen für zulässige Zeichen haben. Weitere Informationen finden Sie in der Dokumentation des jeweiligen Services.
- AWS Private CA -Tags sind nicht für die Verwendung im [Ressourcengruppen- und Tag-Editor](#) im verfügbar AWS Management Console.

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 unter <https://console.aws.amazon.com/acm-pca/home>.
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-authority](#) Befehl , um Ihrer privaten Zertifizierungsstelle 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-authority](#) Befehl , um Tags aus einer privaten Zertifizierungsstelle 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
```

Aktualisieren Ihrer privaten Zertifizierungsstelle

Sie können den Status einer privaten Zertifizierungsstelle aktualisieren oder ihre [Widerrufskonfiguration](#) 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 des CA-Status

Der Status einer CA, die von verwaltet wird, AWS Private CA ergibt sich aus einer Benutzeraktion oder in einigen Fällen aus einer Service-Aktion. Beispielsweise ändert sich ein CA-Status, wenn er 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 jedem Status verfügbar sind.

Note

Für alle Statuswerte außer DELETED und wird FAILEDIhnen die CA in Rechnung gestellt.

Status	Ausstellen von Zertifikaten	Validieren von Zertifikaten mit OCSP	CRLs generieren	Generieren von Audits	Sie können das CA-Zertifikat aktualisieren	Zertifikate können widerrufen werden	Ihnen wird die CA in Rechnung gestellt
CREATING – Die CA wird 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 auf einen Netzwerkausfall, einen Backend-AWS Ausfall oder andere Fehler zurückzuführen sein. Eine fehlgeschlagene CA kann nicht wiederhergestellt werden. Löschen Sie die CA und erstellen Sie eine neue.						Nein
DELETED	Ihre CA befindet sich innerhalb des Wiederherstellungszeitraums, der eine Länge von 7–30 Tagen haben kann. Nach diesem Zeitraum wird sie endgültig gelöscht.						Nein

Status	Ausstellen von Zertifikaten	Validieren von Zertifikaten mit OCSP	CRLs generieren	Generieren von Audits	Sie können das CA-Zertifikat aktualisieren	Zertifikate können widerrufen werden	Ihnen wird die CA in Rechnung gestellt
--------	-----------------------------	--------------------------------------	-----------------	-----------------------	--	--------------------------------------	--

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 Zertifizierungsstelle](#).

* Um die Aktivierung abzuschließen, müssen Sie eine CSR generieren, ein signiertes CA-Zertifikat von einer CA abrufen und das Zertifikat importieren AWS Private CA. Die CSR kann entweder an Ihre neue Zertifizierungsstelle (für die Selbstsignierung) oder an einen On-Premises-Root oder eine untergeordnete Zertifizierungsstelle übermittelt werden. Weitere Informationen finden Sie unter [Erstellen und Installieren des CA-Zertifikats](#).

** Sie können den Status einer abgelaufenen Zertifizierungsstelle nicht direkt ändern. Wenn Sie ein neues Zertifikat für die CA importieren, setzt den Status auf AWS Private CA , ACTIVE es sei denn, es wurde auf gesetzt, DISABLED bevor das Zertifikat abgelaufen ist.

Zusätzliche Überlegungen zu abgelaufenen CA-Zertifikaten:

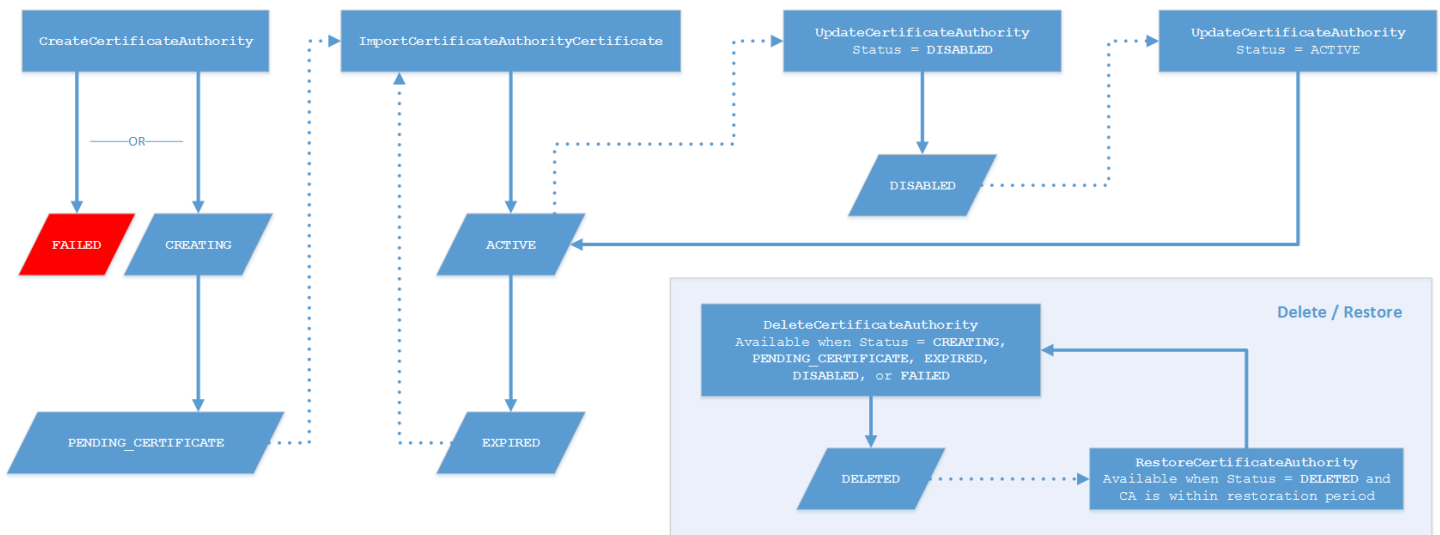
- CA-Zertifikate werden nicht automatisch verlängert. Informationen zur Automatisierung der Erneuerung über finden Sie AWS Certificate Manager unter [Zuweisen von Zertifikatserneuerungsberechtigungen zu ACM](#).
- Wenn Sie versuchen, ein neues Zertifikat mit einer abgelaufenen CA auszustellen, wird die `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 geben den Status EXPIRED zurück, wenn das CA-Zertifikat abgelaufen ist, unabhängig davon, ob der

Status der CA auf ACTIVE oder DISABLED festgelegt ist. Wenn jedoch die abgelaufene CA auf DELETED festgelegt wurde, ist der zurückgegebene Status DELETED.

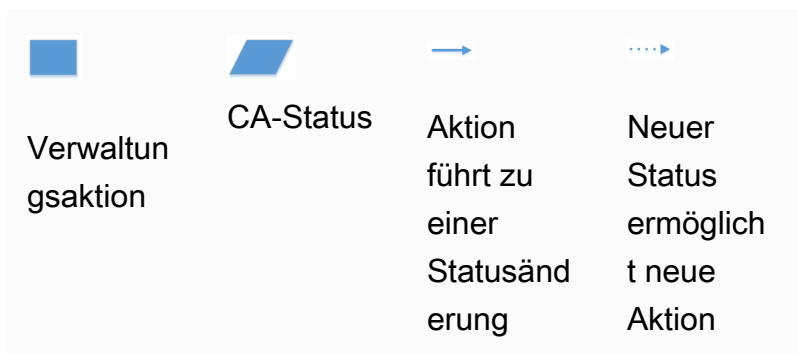
- Die UpdateCertificateAuthority-API kann den Status einer abgelaufenen CA nicht aktualisieren.
- Die RevokeCertificate API kann nicht verwendet werden, um abgelaufene Zertifikate, einschließlich eines CA-Zertifikats, zu widerrufen.

CA-Status und CA-Lebenszyklus

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



Diagrammschlüssel



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 einer Zertifizierungsstelle (Konsole)

Die folgenden Verfahren zeigen, wie Sie vorhandene CA-Konfigurationen mithilfe der aktualisieren AWS Management Console.

CA-Status aktualisieren (Konsole)

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

So aktualisieren Sie den Status einer Zertifizierungsstelle

1. 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>
2. Wählen Sie auf der Seite Private Zertifizierungsstellen eine private Zertifizierungsstelle aus, die derzeit aktiv ist.
3. Wählen Sie im Menü Aktionen die Option Deaktivieren aus, um die private CA zu deaktivieren.

Aktualisieren der Widerrufsconfiguration einer CA (Konsole)

Sie können die [Widerrufsconfiguration](#) für Ihre private Zertifizierungsstelle aktualisieren, indem Sie beispielsweise entweder OCSP- oder CRL-Unterstützung hinzufügen oder entfernen oder deren Einstellungen ändern.

Note

Änderungen an der Widerrufsconfiguration 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-Domännennamen (Fully Qualified Domain Name, FQDN).
- Ändern Sie den FQDN.

Für eine CRL können Sie eine der folgenden Einstellungen ändern:

- Gibt an, ob die private CA eine Zertifikatsperrliste (CRL) generiert
- Die Anzahl der Tage, bevor eine Zertifikatsperrliste abläuft. Beachten Sie, dass mit dem Versuch AWS Private CA beginnt, die CRL um die von Ihnen angegebene Anzahl von Tagen neu zu generieren.
- Der Name des Amazon S3-Buckets, in dem Ihre CRL gespeichert ist.
- Ein Alias zum Ausblenden des Namens Ihres Amazon S3-Buckets in der öffentlichen Ansicht.

Important

Das Ändern eines der vorhergehenden 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 Zertifizierungsstelle in Produktion 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.

So aktualisieren Sie die Widerrufseinstellungen

1. 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>.
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 Widerrufen von Zertifikaten werden zwei Optionen angezeigt:
 - Aktivieren der CRL-Verteilung
 - Aktivieren von OCSP

Sie können entweder, keinen oder beide dieser Widerrufsmechanismen für Ihre CA konfigurieren. Obwohl optional, wird der verwaltete Widerruf als [bewährte Methode](#) empfohlen. Bevor Sie diesen Schritt ausführen, [Einrichtung einer Methode zum Widerruf von Zertifikaten](#) finden Sie unter Informationen zu den Vorteilen der einzelnen Methoden, zur eventuell erforderlichen Vorabereinrichtung und zu zusätzlichen Widerrufsfunktionen.

So konfigurieren Sie eine CRL

1. Wählen Sie CRL-Verteilung aktivieren aus.
2. Um einen Amazon 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.) Lassen Sie andernfalls diese Option deaktiviert und wählen Sie einen vorhandenen Bucket aus der Liste S3-Bucket-Name aus.

Wenn Sie einen neuen Bucket erstellen, AWS Private CA erstellt und fügt ihm die [erforderliche Zugriffsrichtlinie](#) hinzu. Wenn Sie einen vorhandenen Bucket verwenden möchten, müssen Sie ihm eine Zugriffsrichtlinie anfügen, bevor Sie mit der Generierung von CRLs beginnen können. Verwenden Sie eines der unter beschriebenen Richtlinienmuster [Zugriffsrichtlinien für CRLs in Amazon S3](#). Informationen zum Anfügen einer Richtlinie finden Sie unter [Hinzufügen einer Bucket-Richtlinie mithilfe der Amazon S3-Konsole](#).

Note

Wenn Sie die AWS Private CA Konsole verwenden, schlägt ein Versuch, eine CA zu erstellen, fehl, wenn die beiden folgenden Bedingungen zutreffen:

- Sie erzwingen Block Public Access-Einstellungen für Ihren Amazon S3-Bucket oder Ihr Konto.
- Sie haben aufgefordert AWS Private CA, automatisch einen Amazon S3-Bucket zu erstellen.

In diesem Fall 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-Einstellungen. Weitere Informationen finden Sie unter [Blockieren des öffentlichen Zugriffs auf Ihren Amazon S3-Speicher](#).

3. Erweitern Sie Erweitert, um weitere Konfigurationsoptionen zu erhalten.
 - Fügen Sie einen benutzerdefinierten CRL-Namen hinzu, um einen Alias für Ihren Amazon S3-Bucket zu erstellen. Dieser Name ist in Zertifikaten enthalten, die von der Zertifizierungsstelle in der Erweiterung „CRL Distribution Points“ ausgestellt wurden, die durch RFC 5280 definiert ist.
 - Geben Sie die Anzahl der Tage an, die Ihre CRL gültig ist. Der Standardwert lautet 7 Tage. Bei Online-CRLs ist eine Gültigkeitsdauer von 2–7 Tagen üblich. AWS Private CA versucht, die CRL zum Zeitpunkt des angegebenen Zeitraums neu zu generieren.
4. Wählen Sie Änderungen speichern, wenn Sie fertig sind.

So konfigurieren Sie OCSP

1. Wählen Sie auf der Seite Zertifikatswiderruf die Option OCSP aktivieren aus.
2. (Optional) Geben Sie im Feld Benutzerdefinierter OCSP-Endpunkt einen vollqualifizierten Domännennamen (FQDN) für Ihren OCSP-Endpunkt an.

Wenn Sie in diesem Feld einen FQDN angeben, AWS Private CA fügt den FQDN anstelle der Standard-URL für den AWS OCSP-Responder in die Erweiterung des Authority Information Access jedes ausgestellten Zertifikats ein. Wenn ein Endpunkt ein Zertifikat erhält, 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 Proxy-Server, um Datenverkehr, der bei Ihrem benutzerdefinierten FQDN ankommt, an den AWS OCSP-Responder weiterzuleiten.
- Fügen Sie Ihrer DNS-Datenbank einen entsprechenden CNAME-Datensatz hinzu.

 Tip

Weitere Informationen zur Implementierung einer vollständigen OCSP-Lösung mit einem benutzerdefinierten CNAME finden Sie unter [Konfiguration einer benutzerdefinierten URL für OCSP AWS Private CA](#).

Hier ist beispielsweise ein CNAME-Datensatz für benutzerdefinierte OCSP, wie er in Amazon Route 53 angezeigt wird.

Datensatzname	Typ	Routing-Richtlinie	Unterscheidung	Bewerten/ Weiterleiten des Datenverkehrs an
alternati ve.example.com	CNAME	Einfach	-	proxy.exa mple.com


 Note

Der Wert des CNAME darf kein Protokollpräfix wie „http://“ oder „https://“ enthalten.

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

Aktualisieren einer CA (CLI)

Die folgenden Verfahren zeigen, wie Sie den Status und die [Widerrufskonfiguration](#) einer vorhandenen CA mithilfe der aktualisieren AWS CLI.

 Note

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

So aktualisieren Sie den Status Ihrer privaten Zertifizierungsstelle (AWS CLI)

Verwenden Sie den [update-certificate-authority](#)-Befehl.

Dies ist nützlich, wenn Sie über eine vorhandene Zertifizierungsstelle mit dem Status `DISABLED`, den Sie auf `ACTIVE` setzen möchten. Überprüfen 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 ähnlich der folgenden.

```
{
  "CertificateAuthority": {
    "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
    "CreatedAt": "2021-03-05T14:24:12.867000-08:00",
    "LastStateChangeAt": "2021-03-08T13:17:40.221000-08:00",
    "Type": "ROOT",
    "Serial": "serial_number",
    "Status": "DISABLED",
    "NotBefore": "2021-03-08T07:46:27-08:00",
    "NotAfter": "2022-03-08T08:46:27-08:00",
    "CertificateAuthorityConfiguration": {
      "KeyAlgorithm": "RSA_2048",
      "SigningAlgorithm": "SHA256WITHRSA",
      "Subject": {
        "Country": "US",
        "Organization": "Example Corp",
        "OrganizationalUnit": "Sales",
        "State": "WA",
        "CommonName": "www.example.com",
        "Locality": "Seattle"
      }
    },
    "RevocationConfiguration": {
      "CrlConfiguration": {
        "Enabled": true,
        "ExpirationInDays": 7,
        "CustomCname": "alternative.example.com",
        "S3BucketName": "DOC-EXAMPLE-BUCKET1"
      },
      "OcspConfiguration": {
        "Enabled": false
      }
    }
  }
}
```

Mit dem folgenden Befehl wird der Status der privaten Zertifizierungsstelle auf festgelegt **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 `ACTIVE` 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": "2021-03-08T13:23:09.352000-08:00",  
    "Type": "ROOT",  
    "Serial": "serial_number",  
    "Status": "ACTIVE",  
    "NotBefore": "2021-03-08T07:46:27-08:00",  
    "NotAfter": "2022-03-08T08:46:27-08:00",  
    "CertificateAuthorityConfiguration": {  
      "KeyAlgorithm": "RSA_2048",  
      "SigningAlgorithm": "SHA256WITHRSA",  
      "Subject": {  
        "Country": "US",  
        "Organization": "Example Corp",  
        "OrganizationalUnit": "Sales",  
        "State": "WA",  
        "CommonName": "www.example.com",  
        "Locality": "Seattle"  
      }  
    },  
    "RevocationConfiguration": {  
      "CrlConfiguration": {  
        "Enabled": true,  
        "ExpirationInDays": 7,  
        "CustomCname": "alternative.example.com",
```

```

        "S3BucketName": "DOC-EXAMPLE-BUCKET1"
    },
    "OcspConfiguration": {
        "Enabled": false
    }
}
}
}
}

```

In einigen Fällen haben Sie möglicherweise eine aktive Zertifizierungsstelle ohne konfigurierten Widerrufsmechanismus. Wenn Sie eine Zertifikatsperrliste (CRL) verwenden möchten, gehen Sie wie folgt vor.

So fügen Sie eine CRL zu einer vorhandenen CA 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 den Status hat, ACTIVE 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
    }
}
}
}
}

```

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

```

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

```

Note

Wenn Sie eine microSDer-Gerätebescheinigungs-Zertifizierungsstelle aktualisieren, um CRLs zu aktivieren, müssen Sie sie so konfigurieren, dass die CDP-Erweiterung aus den ausgestellten Zertifikaten weggelassen wird, um dem aktuellen Boler-Standard zu entsprechen. Definieren Sie dazu Ihre CRL-Konfigurationsparameter wie unten dargestellt:

```

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

```



```

        "OmitExtension": true
    }
}
}

```

3. Verwenden Sie den [update-certificate-authority](#) Befehl und die Widerrufskonfigurationsdatei, 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_numbner",
    "Status": "ACTIVE",
    "NotBefore": "2021-03-08T13:46:50-08:00",
    "NotAfter": "2022-03-08T14:46:50-08:00",
    "CertificateAuthorityConfiguration": {
      "KeyAlgorithm": "RSA_2048",
      "SigningAlgorithm": "SHA256WITHRSA",
      "Subject": {
        "Country": "US",
        "Organization": "Example Corp",
        "OrganizationalUnit": "Sales",
        "State": "WA",
        "CommonName": "www.example.com",

```

```
        "Locality": "Seattle"
      }
    },
    "RevocationConfiguration": {
      "CrlConfiguration": {
        "Enabled": true,
        "ExpirationInDays": 7,
        "S3BucketName": "DOC-EXAMPLE-BUCKET1",
      },
      "OcspConfiguration": {
        "Enabled": false
      }
    }
  }
}
```

In einigen Fällen möchten Sie möglicherweise OCSP-Widerrufsunterstützung hinzufügen, anstatt eine CRL wie im vorherigen Verfahren zu aktivieren. Führen Sie in diesem Fall die folgenden Schritte aus.

So fügen Sie OCSP-Unterstützung zu einer vorhandenen CA hinzu (AWS CLI)

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

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

2. Verwenden Sie den [update-certificate-authority](#) Befehl und die Widerrufskonfigurationsdatei, 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-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 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 CRL- als auch OCSP-Unterstützung auf einer CA konfigurieren.


Löschen Ihrer privaten Zertifizierungsstelle

Sie können eine private AWS CLI Zertifizierungsstelle dauerhaft aus der AWS Management Console oder 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 eine Löschanforderung verarbeitet wird. 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` Status `PENDING_CERTIFICATE` oder löschen, können Sie die Länge des Wiederherstellungszeitraums auf 7–30 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, wurde kein Wiederherstellungszeitraum zugewiesen und sie kann nicht wiederhergestellt werden. Weitere Informationen finden Sie unter [Wiederherstellen einer privaten Zertifizierungsstelle](#).

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](#).

So löschen Sie eine private CA (Konsole)

1. 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>.
2. Wählen Sie auf der Seite Private Zertifizierungsstellen Ihre private Zertifizierungsstelle aus der Liste aus.
3. Wenn sich Ihre CA im ACTIVE Status befindet, müssen Sie sie zunächst deaktivieren. Wählen Sie im Menü Aktionen die Option Deaktivieren. Wenn Sie dazu aufgefordert werden, wählen Sie Ich habe das Risiko verstanden, fahren Sie fort.
4. Wählen Sie für eine Zertifizierungsstelle, die sich nicht im ACTIVE Status befindet, Aktionen, Löschen aus.
5. Wenn sich Ihre CA im PENDING_CERTIFICATE Status DISABLED, EXPIRED oder befindet, können Sie auf der Seite CA löschen eine Wiederherstellungsdauer von 7 bis 30 Tagen angeben. Wenn sich Ihre private Zertifizierungsstelle nicht in einem dieser Zustände befindet, kann sie später nicht wiederhergestellt werden und der Löschvorgang 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. Um den Wiederherstellungszeitraum einer privaten Zertifizierungsstelle im DELETED Status zu überprüfen, rufen Sie die - [DescribeCertificateAuthority](#) oder [ListCertificateAuthorities](#)-API-Operation auf.

So löschen Sie eine private CA (AWS CLI)

Verwenden Sie den [delete-certificate-authority](#) Befehl , um eine private Zertifizierungsstelle 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
```

Wiederherstellen einer privaten Zertifizierungsstelle

Sie können eine gelöschte private Zertifizierungsstelle wiederherstellen, solange sich die Zertifizierungsstelle innerhalb des Wiederherstellungszeitraums befindet, den Sie beim Löschen angegeben haben. Der Wiederherstellungszeitraum liegt zwischen 7 und 30 Tagen. Nach Ablauf dieses Zeitraums wird die private Zertifizierungsstelle dauerhaft gelöscht. Weitere Informationen finden Sie unter [Löschen Ihrer privaten Zertifizierungsstelle](#). 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](#).

Wiederherstellen einer privaten Zertifizierungsstelle (Konsole)

Sie können die verwendete AWS Management Console , um eine private Zertifizierungsstelle 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 unter <https://console.aws.amazon.com/acm-pca/home>.
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 CA wiederherstellen erneut Wiederherstellen 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 in 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.

Wiederherstellen einer privaten Zertifizierungsstelle (AWS CLI)

Verwenden Sie den [restore-certificate-authority](#) Befehl , 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-authority](#) Befehl aus, um die private Zertifizierungsstelle zu löschen. Wenn der Status der privaten Zertifizierungsstelle DISABLED oder lautet PENDING_CERTIFICATE, können Sie den `--permanent-deletion-time-in-days` Parameter festlegen, um den Wiederherstellungszeitraum der privaten Zertifizierungsstelle von 7 bis 30 Tagen anzugeben. 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-authority](#) Befehl aus, um die private Zertifizierungsstelle 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-authority](#) Befehl aus, um den Status der privaten CA in zu ändernACTIVE.

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


Verwaltung von Zertifikaten

Nachdem Sie eine private Zertifizierungsstelle (CA) erstellt und aktiviert und den Zugriff darauf konfiguriert haben, können Sie oder Ihre autorisierten Benutzer die in diesem Abschnitt beschriebenen Aufgaben ausführen. Wenn Sie noch keine AWS Identity and Access Management (IAM)-Richtlinien für die CA eingerichtet haben, erfahren Sie mehr über deren Konfiguration im Abschnitt [Identity and Access Management](#) dieses Handbuchs. Informationen zur Konfiguration des CA-Zugriffs in konto- und kontoübergreifenden Szenarien finden Sie unter [Steuern des Zugriffs auf eine private Zertifizierungsstelle](#).

Themen

- [Ausstellen privater Endentitätszertifikate](#)
- [Abrufen eines privaten Zertifikats](#)
- [Auflisten privater Zertifikate](#)
- [Exportieren eines privaten Zertifikats und seines geheimen Schlüssels](#)
- [Widerrufen eines privaten Zertifikats](#)
- [Automatisieren des Exports eines erneuerten Zertifikats](#)
- [Grundlegendes zu Zertifikatsvorlagen](#)

Ausstellen privater Endentitätszertifikate

Wenn eine private Zertifizierungsstelle vorhanden ist, können Sie private Endentitätszertifikate entweder von AWS Certificate Manager (ACM) oder anfordern AWS Private CA. Die Funktionen beider Services werden in der folgenden Tabelle verglichen.

Funktion	ACM	AWS Private CA
Ausstellen von Endentitätszertifikaten	Bol (mit RequestCertificate oder der Konsole)	Bol (mit IssueCertificate)
Zuordnung zu Load Balancern und AWS Services, die mit dem Internet verbunden sind	✓	Nicht unterstützt

Funktion	ACM	AWS Private CA
Erneuerung verwalteter Zertifikate	✓	Indirekt über ACM unterstützt
Konsolenunterstützung	✓	Nicht unterstützt
API-Unterstützung	✓	✓
CLI-Unterstützung	✓	✓

Wenn ein Zertifikat AWS Private CA erstellt, folgt es einer Vorlage, die den Zertifikattyp und die Pfadlänge angibt. Wenn der API- oder CLI-Anweisung, die das Zertifikat erstellt, kein Vorlagen-ARN bereitgestellt wird, wird die [EndEntityCertificate/V1](#)-Vorlage standardmäßig angewendet. Weitere Informationen zu verfügbaren Zertifikatvorlagen finden Sie unter [Grundlegendes zu Zertifikatsvorlagen](#).

ACM-Zertifikate sind zwar auf öffentliche Vertrauensstellung ausgelegt, AWS Private CA erfüllt jedoch die Anforderungen Ihrer privaten PKI. Folglich können Sie Zertifikate mit der AWS Private CA API und CLI auf eine Weise konfigurieren, die von ACM nicht zugelassen ist. Diese umfassen u. a. folgende:

- Erstellen eines Zertifikats mit einem beliebigen Betreffnamen.
- Verwenden eines der [unterstützten privaten Schlüsselalgorithmen und Schlüssellängen](#) .
- Verwenden eines der [unterstützten Signaturalgorithmen](#) .
- Angabe eines Gültigkeitszeitraums für Ihre private [Zertifizierungsstelle](#) und private [Zertifikate](#).

Nachdem Sie ein privates TLS-Zertifikat mit erstellt haben AWS Private CA, können Sie es in ACM [import](#) ieren und mit einem unterstützten AWS Service verwenden.

Note

Zertifikate, die mit dem folgenden Verfahren, mit dem `-issue-certificate` Befehl oder mit der [IssueCertificate](#)-API-Aktion erstellt wurden, können nicht direkt zur Verwendung außerhalb von exportiert werden AWS. Sie können jedoch Ihre private Zertifizierungsstelle verwenden, um Zertifikate zu signieren, die über ACM ausgestellt wurden, 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.

Ausstellen eines Standardzertifikats (AWS CLI)

Sie können das AWS Private CA CLI-Befehl [issue-certificate](#) oder die API-Aktion [IssueCertificate](#), um ein Endentitätszertifikat anzufordern. 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 oder verwenden, AWS CLI um ein privates Zertifikat auszustellen, wird das Zertifikat nicht verwaltet, was bedeutet, dass Sie die ACM-Konsole, die ACM-CLI oder die ACM-API nicht verwenden können, um es anzuzeigen oder zu exportieren, und das Zertifikat wird nicht automatisch erneuert. Sie können jedoch den Befehl PCA [get-certificate](#) verwenden, um die Zertifikatdetails abzurufen. Wenn Sie Eigentümer der CA sind, können Sie einen [Auditbericht](#) erstellen.

Überlegungen beim Erstellen von Zertifikaten

- In Übereinstimmung mit [RFC 5280](#) darf die Länge des von Ihnen angegebenen Domännennamens (technisch gesehen der allgemeine Name) 64 Oktette (Zeichen), einschließlich Punkte, nicht überschreiten. Um einen längeren Domännennamen hinzuzufügen, geben Sie ihn im Feld Alternative Antragstellernamen an, das Namen mit einer Länge von bis zu 253 Oktette unterstützt.
- Wenn Sie AWS CLI Version 1.6.3 oder höher verwenden, verwenden Sie das Präfix `fileb://` wenn Sie base64-kodierte Eingabedateien wie CSRs angeben. Dadurch wird sichergestellt, dass die Daten korrekt AWS Private CA analysiert.

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 verkürzten 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

Der folgende Befehl erstellt ein Zertifikat. Da keine Vorlage angegeben ist, wird standardmäßig ein Basis-Endentitätszertifikat 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 der `issue-certificate` Befehl empfangen wird. Die Zertifikatsverarbeitung erfolgt jedoch asynchron und kann trotzdem fehlschlagen. In diesem Fall schlägt ein `get-certificate` Befehl, der den neuen ARN verwendet, ebenfalls fehl.

Ein Zertifikat mit einem benutzerdefinierten Betreffnamen mithilfe einer APIPassthrough-Vorlage ausstellen

In diesem Beispiel wird ein Zertifikat ausgestellt, das benutzerdefinierte Elemente des Betreffnamens enthält. Zusätzlich zur Bereitstellung einer CSR wie der in [übergeben](#) [Ausstellen eines Standardzertifikats \(AWS CLI\)](#) Sie zwei zusätzliche Argumente an den `issue-certificate` Befehl: den ARN einer APIPassthrough-Vorlage und eine JSON-Konfigurationsdatei, die die benutzerdefinierten Attribute und ihre Objektkennungen (OIDs) angibt. Sie können nicht `StandardAttributes` in Verbindung mit `verwendenCustomAttributes`. Sie können jedoch Standard-OIDs als Teil von `übergebenCustomAttributes`. Die OIDs des Standard-Betreffnamens sind in der folgenden Tabelle aufgeführt (Informationen aus [RFC 4519](#) und der [globalen OID-Referenzdatenbank](#)):

Betreff	Abkürzung	Objekt-ID
<code>countryName</code>	<code>c</code>	2.5.4.6
<code>commonName</code>	<code>cn</code>	2.5.4.3
<code>dnQualifier [Distinguished Name Qualifier]</code>		2.5.4.46
<code>generationQualifier</code>		2.5.4.44
<code>givenName</code>		2.5.4.42
<code>-Initialen</code>		2.5.4.43

Betreff	Abkürzung	Objekt-ID
Ort	l	2.5.4.7
organizationName	o	2.5.4.10
organizationalUnitName	ou	2.5.4.11
Trichter		2.5.4.65
serialNumber		2.5.4.5
st [Zustand]		2.5.4.8
Nachname	sn	2.5.4.4
Titel		2.5.4.12
domainComponent	dc	0.9.2342.19200300.100.1.25
userid		0.9.2342.19200300.100.1.1

Die Beispielkonfigurationsdatei `api_passthrough_config.txt` enthält den folgenden Code:

```
{
  "Subject": {
    "CustomAttributes": [
      {
        "ObjectIdentifier": "2.5.4.6",
        "Value": "US"
      },
      {
        "ObjectIdentifier": "1.3.6.1.4.1.37244.1.1",
        "Value": "BCDABCD12341234"
      },
      {
        "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 file://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 von ihr ausgestellte Zertifikat übergibt.

Ausstellen eines Zertifikats mit benutzerdefinierten Erweiterungen mithilfe einer APIPassthrough-Vorlage

In diesem Beispiel wird ein Zertifikat ausgestellt, das benutzerdefinierte Erweiterungen enthält. Dazu müssen Sie drei Argumente an den `issue-certificate` Befehl übergeben: den ARN einer APIPassthrough-Vorlage und eine JSON-Konfigurationsdatei, die die benutzerdefinierten Erweiterungen angibt, und eine CSR wie die in gezeigte [Ausstellen eines Standardzertifikats \(AWS CLI\)](#).

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 file://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
```

Abrufen eines privaten Zertifikats

Sie können die AWS Private CA-API und AWS CLI verwenden, um ein privates Zertifikat auszustellen. Wenn Sie dies tun, können Sie die AWS Private CA-API oder die AWS CLI verwenden, um dieses Zertifikat abzurufen. Wenn Sie ACM verwendet haben, um Ihre private Zertifizierungsstelle 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](#).

So rufen Sie ein Endentitätszertifikat ab

Verwenden Sie den AWS CLI Befehl [get-certificate](#), um ein privates Endentitätszertifikat abzurufen. Sie können auch die [GetCertificate](#)-API-Operation 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 von Auditberichten 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 Zertifikatkette 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-----
```

So rufen Sie ein CA-Zertifikat ab

Sie können die AWS Private CA-API und AWS CLI zum Abrufen des Zertifizierungsstellenzertifikats (CA-Zertifikats) für Ihre private CA verwenden. 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-----
```

Auflisten privater Zertifikate

Um Ihre privaten Zertifikate aufzulisten, generieren Sie einen Auditbericht, rufen Sie ihn aus seinem S3-Bucket ab und analysieren Sie den Berichtsinhalt nach Bedarf. Informationen zum Erstellen von

AWS Private CA Auditberichten finden Sie unter [Verwenden von Auditberichten mit Ihrer privaten Zertifizierungsstelle](#). Informationen zum Abrufen eines Objekts aus einem S3-Bucket finden Sie unter [Herunterladen eines Objekts](#) im Benutzerhandbuch für Amazon Simple Storage Service.

Die folgenden Beispiele veranschaulichen Ansätze zum Erstellen von Prüfungsberichten und zum Parsen nützlicher Daten. Die Ergebnisse sind in JSON formatiert und die Daten werden mit `jq` gefiltert, einem sed-ähnlichen Parser.

1. Erstellen Sie einen Auditbericht.

Der folgende Befehl generiert einen Auditbericht 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 Auditberichts zurück.

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

2. Rufen Sie einen Auditbericht ab und formatieren Sie ihn.

Dieser Befehl ruft einen Auditbericht ab, zeigt seinen Inhalt in der Standardausgabe an und filtert die Ergebnisse so, dass nur Zertifikate angezeigt werden, die am oder nach dem 2020-12-01 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 Elemente ähneln Folgendem:

```
{
```

```

    "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 Auditbericht lokal.

Wenn Sie mehrere Abfragen durchführen möchten, ist es praktisch, einen Auditbericht 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 ergibt 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. Abfragen innerhalb eines Datumsbereichs

Sie können Zertifikate, die innerhalb eines Datumsbereichs ausgestellt wurden, wie folgt abfragen:

```

$ 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 nach einer angegebenen Vorlage.

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")'
```

Die Ausgabe zeigt übereinstimmende Zertifikatsdatensätze an:

```
{
  "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. Filtern 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 mit einem regulären Ausdruck.

Der folgende Befehl sucht nach Subjektnamen, die die Zeichenfolge „leaf“ enthalten:

```
$ cat my_local_audit_report.json | jq '.[] | select(.subject|test("leaf"))'
```

Abgleichende 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 eines privaten Zertifikats und seines geheimen Schlüssels

AWS Private CA kann ein privates Zertifikat, das es signiert und ausgestellt hat, nicht direkt exportieren. Sie können 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 bereitgestellt werden. Weitere Informationen finden Sie unter [Exportieren eines privaten Zertifikats](#) im AWS Certificate Manager-Benutzerhandbuch.

Als zusätzlicher Vorteil AWS Certificate Manager bietet eine verwaltete Erneuerung für private Zertifikate, die über die ACM-Konsole, die `-RequestCertificate`Aktion der ACM-API oder den `-request-certificate`Befehl im ACM-Abschnitt der `awscli` ausgestellt wurden. Weitere Informationen zu Erneuerungen finden Sie unter [Erneuern von Zertifikaten in einer privaten PKI](#).

Widerrufen eines privaten Zertifikats

Sie können ein `-AWS Private CA`Zertifikat mit dem AWS CLI Befehl [revoke-certificate](#) oder der [-RevokeCertificate](#)API-Aktion widerrufen. Ein Zertifikat muss möglicherweise vor seinem geplanten Ablauf widerrufen werden, wenn beispielsweise der geheime Schlüssel kompromittiert wurde oder die zugehörige Domain ungültig wird. Damit der Widerruf wirksam wird, benötigt der Client, der das Zertifikat verwendet, eine Möglichkeit, den Widerrufsstatus zu überprüfen, wenn er versucht, eine sichere Netzwerkverbindung herzustellen.

AWS Private CA bietet zwei vollständig verwaltete Mechanismen zur Unterstützung der Überprüfung des Widerrufsstatus: Online Certificate Status Protocol (OCSP) und Zertifikatsperrlisten (CRLs). Mit OCSP fragt der Client eine autoritative Widerrufsdatenbank ab, die einen Status in Echtzeit zurückgibt. Bei einer CRL prüft der Client das Zertifikat anhand einer Liste widerrufenen Zertifikate, die er regelmäßig herunterlädt und speichert. Clients verweigern die Annahme von Zertifikaten, die widerrufen wurden.

Sowohl OCSP als auch CRLs hängen von Validierungsinformationen ab, die in Zertifikaten eingebettet sind. Aus diesem Grund muss eine ausstellende Zertifizierungsstelle so konfiguriert werden, dass sie vor der Ausstellung einen oder beide dieser Mechanismen unterstützt. Informationen zur Auswahl und Implementierung des verwalteten Widerrufs finden Sie in [AWS Private CA](#) unter [Einrichtung einer Methode zum Widerruf von Zertifikaten](#).

Widerrufene Zertifikate werden immer in AWS Private CA Prüfungsberichten aufgezeichnet.

Note

[Kontoubergreifende](#) Zertifikataussteller benötigen zusätzliche Berechtigungen, um die von ihnen ausgestellten Zertifikate zu widerrufen. Andernfalls muss der CA-Besitzer den Widerruf

durchführen. Um den Widerruf durch kontoübergreifende Aussteller zu aktivieren, muss der CA-Administrator zwei RAM-Freigaben erstellen, die beide auf dieselbe CA verweisen:

1. Eine Freigabe mit der `-AWSRAMRevokeCertificateCertificateAuthorityBerechtigung`.
2. Eine Freigabe mit der `-AWSRAMDefaultPermissionCertificateAuthorityBerechtigung`.

So widerrufen Sie ein Zertifikat

Verwenden Sie die [-RevokeCertificate](#) API-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

OCSP-Antworten können bis zu 60 Minuten dauern, bis der neue Status angezeigt wird, wenn Sie ein Zertifikat widerrufen. Im Allgemeinen unterstützt OCSP in der Regel eine schnellere Verteilung von Widerrufsinformationen, da im Gegensatz zu CRLs, die von Clients tagelang zwischengespeichert werden können, OCSP-Antworten 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. Wenn eine CRL-Aktualisierung aus irgendeinem Grund fehlschlägt, AWS Private CA unternimmt alle 15 Minuten weitere Versuche.

Mit Amazon können CloudWatch Sie Alarmer für die Metriken `CRLGenerated` und `MisconfiguredCRLBucket` erstellen. Weitere Informationen finden Sie unter [Unterstützte CloudWatch Metriken](#). Weitere Informationen zum Erstellen und Konfigurieren von Zertifikatssperrlisten finden Sie unter [Planung einer Zertifikatssperrliste \(CRL\)](#).

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 von Auditberichten 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.example2.com,OU=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 des Exports eines erneuerten Zertifikats

Wenn Sie verwenden, AWS Private CA um eine Zertifizierungsstelle zu erstellen, können Sie diese Zertifizierungsstelle in importieren AWS Certificate Manager und ACM die Ausstellung und Erneuerung von Zertifikaten verwalten lassen. Wenn ein Zertifikat, das erneuert wird, einem [integrierten Service](#) zugeordnet ist, wendet der Service das neue Zertifikat nahtlos an. Wenn das Zertifikat jedoch ursprünglich zur Verwendung an anderer Stelle in Ihrer PKI-Umgebung [exportiert](#) wurde (z. B. auf einem On-Premises-Server oder einer On-Premises-Appliance), müssen Sie es nach der Verlängerung erneut exportieren.

Eine Beispiellösung, die den ACM-Exportprozess mit Amazon EventBridge und AWS Lambda automatisiert, finden Sie unter [Automatisieren des Exports erneuerter Zertifikate](#).

Grundlegendes zu Zertifikatsvorlagen

AWS Private CA verwendet Konfigurationsvorlagen, um sowohl CA-Zertifikate als auch Endentitätszertifikate auszustellen. Wenn Sie ein CA-Zertifikat über die PCA-Konsole ausstellen, wird die entsprechende Stamm- oder untergeordnete CA-Zertifikatsvorlage automatisch angewendet.

Wenn Sie die CLI oder API verwenden, um ein Zertifikat auszustellen, können Sie der `IssueCertificate` Aktion einen Vorlagen-ARN als Parameter bereitstellen. Wenn Sie keinen ARN angeben, wird die `EndEntityCertificate/V1` Vorlage standardmäßig angewendet. Weitere Informationen finden Sie in der Dokumentation zur [IssueCertificate](#) API und zum Befehl [issue-certificate](#).

Note

AWS Certificate Manager (ACM)-Benutzer mit kontoübergreifendem 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 Endentitätszertifikate:

- [EndEntityCertificate/V1](#)
- [EndEntityClientAuthCertificate/V1](#)
- [EndEntityServerAuthCertificate/V1](#)
- [BlankEndEntityCertificate_APIPassthrough /V1](#)
- [BlankEndEntityCertificate_APICSRPassthrough /V1](#)
- [SubordinateCACertificate_PathLen0/V1](#)

Weitere Informationen finden Sie unter [Ressourcenbasierte Richtlinien](#).

Themen

- [Vorlagenvarianten](#)
- [Reihenfolge der Operationen in Vorlagen](#)

- [Vorlagendefinitionen](#)

Vorlagenvarianten

AWS Private CA unterstützt vier Varianten von Vorlagen.

- Basisvorlagen

Vordefinierte Vorlagen, in denen keine Pass-Through-Parameter zulässig sind.

- CSRPassthrough-Vorlagen

Vorlagen, die ihre entsprechenden Basisvorlagenversionen erweitern, indem sie CSR-Passthrough zulassen. Erweiterungen in der CSR, die zum Ausstellen des Zertifikats verwendet wird, werden in 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 unter [Reihenfolge der Operationen in Vorlagen](#).

- APIPassthrough-Vorlagen

Vorlagen, die ihre entsprechenden Basisvorlagenversionen erweitern, indem sie API-Passthrough zulassen. Dynamische Werte, die dem Administrator oder anderen Zwischensystemen bekannt sind, sind möglicherweise von der Entität, die das Zertifikat anfordert, 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 abrufen, z. B. aus einem Active Directory, um die Anforderung abzuschließen. Wenn ein Computer beispielsweise nicht weiß, zu welcher Organisationseinheit er gehört, kann der Administrator die Informationen in Active Directory nachschlagen und sie der Zertifikatanforderung hinzufügen, indem er die Informationen in eine JSON-Struktur einfügt.

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 unter [Reihenfolge der Operationen in Vorlagen](#).

- APICSRPassthrough-Vorlagen

Vorlagen, die ihre entsprechenden Basisvorlagenversionen erweitern, indem sie sowohl API- als auch CSR-Passthrough zulassen. Erweiterungen in der CSR, die zum Ausstellen des Zertifikats verwendet werden, werden in das ausgegebene Zertifikat kopiert, und Werte im

-ApiPassthroughParameter der -IssueCertificateAktion werden ebenfalls über kopiert. In Fällen, in denen die Vorlagendefinition, API-Pass-Through-Werte und CSR-Pass-Through-Erweiterungen einen Konflikt aufweisen, hat die Vorlagendefinition höchste Priorität, gefolgt von den API-Pass-Through-Werten, gefolgt von den CSR-Pass-Through-Erweiterungen. Weitere Informationen zur Priorität finden Sie unter [Reihenfolge der Operationen in Vorlagen](#).

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 unter [AWS Private Certificate Authority](#) im 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/EndEntityServerAuthCertificate/V1	Endentität

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
OCSPSigningCertificate/V1	arn:aws:acm-pca:::template/OCSPSigningCertificate/V1	OCSP Signing
RootCACertificate/V1	arn:aws:acm-pca:::template/RootCACertificate/V1	CA
SubordinateCACertificate_PathLen0/V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1	CA
SubordinateCACertificate_PathLen1/V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1/V1	CA
SubordinateCACertificate_PathLen2/V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2/V1	CA
SubordinateCACertificate_PathLen3/V1	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	Endentität

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
	entityCertificate_CSRPassthrough/V1	
BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassthrough /V1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassthrough/V1	Endentität
BlankSubordinateCACertificate_PathLen0_CSRPassthrough /V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen1_CSRPassthrough /V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen2_CSRPassthrough /V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen3_CSRPassthrough /V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
CodeSigningCertificate_CSRPassthrough/V1	arn:aws:acm-pca:::template/CodeSigningCertificate_CSRPassthrough/V1	Codesignatur
EndEntityCertificate_CSRPassthrough/V1	arn:aws:acm-pca:::template/EndEntityCertificate_CSRPassthrough/V1	Endentität
EndEntityClientAuthCertificate_CSRPassthrough/V1	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_CSRPassthrough/V1	Endentität
EndEntityServerAuthCertificate_CSRPassthrough/V1	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_CSRPassthrough/V1	Endentität
OCSP SigningCertificate_CSRPassthrough/V1	arn:aws:acm-pca:::template/OCSPSigningCertificate_CSRPassthrough/V1	OCSP Signing
SubordinateCACertificate_PathLen0_CSRPassthrough/V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
SubordinateCACertificate_PathLen1_CSRPassthrough/V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
SubordinateCACertificate_PathLen2_CSRPassthrough /V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
SubordinateCACertificate_PathLen3_CSRPassthrough /V1	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_APIPassthrough /V1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough/V1	Endentität
CodeSigningCertificate_APIPassthrough /V1	arn:aws:acm-pca:::template/CodeSigningCertificate_APIPassthrough/V1	Codesignatur
EndEntityCertificate_APIPassthrough /V1	arn:aws:acm-pca:::template/EndEntity	Endentität

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
	Certificate_APIPassthrough/V1	
EndEntityClientAuthCertificate_APIPassthrough/V1	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_APIPassthrough/V1	Endentität
EndEntityServerAuthCertificate_APIPassthrough/V1	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APIPassthrough/V1	Endentität
OCSP SigningCertificate_APIPassthrough /V1	arn:aws:acm-pca:::template/OCSPSigningCertificate_APIPassthrough/V1	OCSP Signing
RootCACertificate_APIPassthrough /V1	arn:aws:acm-pca:::template/RootCACertificate_APIPassthrough/V1	CA
BlankRootCACertificate_APIPassthrough /V1	arn:aws:acm-pca:::template/BlankRootCACertificate_APIPassthrough/V1	CA
BlankRootCACertificate_PathLen0_APIPassthrough /V1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen0_APIPassthrough/V1	CA

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
BlankRootCACertificate_PathLen1_APIPassthrough /V1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen1_APIPassthrough/V1	CA
BlankRootCACertificate_PathLen2_APIPassthrough /V1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen2_APIPassthrough/V1	CA
BlankRootCACertificate_PathLen3_APIPassthrough /V1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen3_APIPassthrough/V1	CA
SubordinateCACertificate_PathLen0_APIPassthrough /V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APIPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen0_APIPassthrough /V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1	CA
SubordinateCACertificate_PathLen1_APIPassthrough /V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APIPassthrough/V1	CA

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
BlankSubordinateCACertificate_PathLen1_APIPassthrough /V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
SubordinateCACertificate_PathLen2_APIPassthrough /V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen2_APIPassthrough /V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
SubordinateCACertificate_PathLen3_APIPassthrough /V1	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_A PICSRPassthrough /V1	arn:aws:acm-pca::: template/BlankEndE ntityCertificate_A PICSRPassthrough/V1	Endentität
BlankEndEntityCertificate_C riticalBasicConstraints_API CSRPassthrough /V1	arn:aws:acm-pca::: template/BlankEndE ntityCertificate_C riticalBasicConstr aints_APICSRPassth rough/V1	Endentität
CodeSigningCertificate_API SRPassthrough /V1	arn:aws:acm-pca::: template/CodeSigni ngCertificate_API SRPassthrough/V1	Codesignatur
EndEntityCertificate_APICSR Passthrough /V1	arn:aws:acm-pca::: template/EndEntity Certificate_APICSR Passthrough/V1	Endentität
EndEntityClientAuthCertific ate_APICSRPassthrough /V1	arn:aws:acm-pca::: template/EndEntity ClientAuthCertific ate_APICSRPassthrough/ V1	Endentität
EndEntityServerAuthCertific ate_APICSRPassthrough /V1	arn:aws:acm-pca::: template/EndEntity ServerAuthCertific	Endentität

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
	ate_APICSRPassthrough/ V1	
OCSP SigningCertificate _APICSRPassthrough /V1	arn:aws:acm-pca::: template/OCSPSigni ngCertificate_APIC SRPassthrough/V1	OCSP Signing
SubordinateCACertificate _PathLen0_APICSRPassthrough / V1	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen0_APICSRPasst hrough/V1	CA
BlankSubordinateCACertificate _PathLen0_APICSRPassthrough / V1	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen0_APICSR Passthrough/V1	CA
SubordinateCACertificate _PathLen1_APICSRPassthrough / V1	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen1_APICSRPasst hrough/V1	CA
BlankSubordinateCACertificate _PathLen1_APICSRPassthrough / V1	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen1_APICSR Passthrough/V1	CA

Name der Vorlage	Vorlagen-ARN	Zertifikatstyp
SubordinateCACertificate_PathLen2_APICSRPassthrough_PathLen3_APIPassthroughV1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APICSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen2_APICSRPassthroughV1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1	CA
SubordinateCACertificate_PathLen3_APICSRPassthroughV1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APICSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen3_APICSRPassthroughV1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1	CA

Reihenfolge der Operationen in Vorlagen

Informationen, die in einem ausgestellten Zertifikat enthalten sind, können aus vier Quellen stammen: der Vorlagendefinition, dem API-Passthrough, dem CSR-Passthrough und der CA-Konfiguration.

API-Pass-Through-Werte werden nur eingehalten, wenn Sie eine API-Pass-Through- oder APICSR-Pass-Through-Vorlage verwenden. CSR-Pass-Through wird nur eingehalten, wenn Sie eine CSR-Passthrough- oder APICSR-Pass-Through-Vorlage verwenden. Wenn diese Informationsquellen in Konflikt stehen, gilt normalerweise eine allgemeine Regel: Für jeden Erweiterungswert hat die Vorlagendefinition die höchste Priorität, gefolgt von API-Pass-Through-Werten, gefolgt von CSR-Pass-Through-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-Pass-Through-Variante handelt.

Note

Die Vorlagendefinition kopiert dennoch über andere Werte aus der CSR, 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-Pass-Through-Variante handelt, da die Vorlagendefinition immer die höchste Priorität hat.

2. Die Vorlagendefinition für [EndEntityClientAuthCertificate_APICSRPassthrough](#) definiert die Subject Alternative Name (SAN)-Erweiterung als aus der API oder CSR kopiert. Wenn die SAN-Erweiterung in der CSR definiert und im -IssueCertificate ApiPassthroughParameter bereitgestellt wird, hat der API-Pass-Through-Wert Priorität, da API-Pass-Through-Werte Vorrang vor CSR-Pass-Through-Werten haben.

Vorlagendefinitionen

Die folgenden Abschnitte enthalten Konfigurationsdetails zu unterstützten AWS Private CA Zertifikatsvorlagen.

BlankEndEntityCertificate_APIPassthrough/V1-Definition

Mit leeren Zertifikatsvorlagen für Endentitäten können Sie Endentitätszertifikate mit nur X.509 Basic-Einschränkungen ausstellen. Dies ist das einfachste Endentitätszertifikat, das ausstellen AWS Private CA kann, aber es kann mithilfe der API-Struktur angepasst werden. Die Erweiterung Grundlegende Einschränkungen definiert, ob es sich bei dem Zertifikat um ein CA-Zertifikat handelt oder nicht. Eine leere Vorlage für ein Endentitätszertifikat erzwingt den Wert FALSE für Grundlegende Einschränkungen, um sicherzustellen, dass ein Endentitätszertifikat ausgestellt wird und kein CA-Zertifikat.

Sie können leere Pass-Through-Vorlagen verwenden, um Smartcard-Zertifikate auszustellen, die bestimmte Werte für die Schlüsselnutzung (KU) und die erweiterte Schlüsselnutzung (EKU) erfordern. Beispielsweise kann die erweiterte Schlüsselnutzung die Client-Authentifizierung und Smartcard-Anmeldung erfordern, und die Schlüsselnutzung kann die digitale Signatur, Nicht-Ablehnung und das Schlüsselprinzipal erfordern. Im Gegensatz zu anderen Pass-Through-Vorlagen ermöglichen leere Endentitätszertifikatsvorlagen die Konfiguration von KU- und EKU-Erweiterungen, wobei KU jeder der neun unterstützten Werte sein kann (digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly und decipherOnly) und EKU jeder der unterstützten Werte sein kann (serverAuth, clientAuth, codesigning, emailProtection, timestamping und OCSPSigning) sowie benutzerdefinierte Erweiterungen.

BlankEndEntityCertificate_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankEndEntityCertificate_APICSRPassthrough/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter [BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankEndEntityCertificate_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]

X509v3-Parameter	Wert
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankEndEntityCertificate_CriticalBasicConstraints_APICSRPassthrough/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter

[BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankEndEntityCertificate_CriticalBasicConstraints_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	Kritisch, CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Pass-Through von 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_APIPassthrough/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter [BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	Kritisch, CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Pass-Through von 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_CSRPassthrough/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter [BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	Kritisch, CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]

X509v3-Parameter	Wert
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankEndEntityCertificate_CSRPassthrough/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter [BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankEndEntityCertificate_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Passthrough von 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 /V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter [BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen0_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 0
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Pass-Through von 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 /V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter [BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen0_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 0
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Pass-Through von 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/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter

[BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen0_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 0
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter

[BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen1_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 1
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

X509v3-Parameter	Wert
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter [BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen1_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 1
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Pass-Through von 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 /V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter [BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen1_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 1
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Pass-Through von 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/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter [BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen2_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 2
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen2_CSRPassthrough /V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter

[BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen2_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 2
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Pass-Through von 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 /V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter

[BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen2_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 2

X509v3-Parameter	Wert
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Pass-Through von 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/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter

[BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen3_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 3
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

BlankSubordinateCACertificate_PathLen3_CSRPassthrough /V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter

[BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen3_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 3
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Pass-Through von 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/V1-Definition

Allgemeine Informationen zu leeren Vorlagen finden Sie unter [BlankEndEntityCertificate_APIPassthrough/V1-Definition](#).

BlankSubordinateCACertificate_PathLen3_APICSRPassthrough

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 3
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
CRL-Verteilungspunkte*	[Pass-Through von 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 Codesignatur-Zertifikate aus AWS Private CA mit jeder Codesignaturlösung verwenden, die auf einer privaten CA-Infrastruktur basiert. Kunden, die die Codesignatur für AWS IoT verwenden, können beispielsweise ein Codesignatur-Zertifikat mit AWS Private CA generieren und in AWS Certificate Manager importieren. Weitere Informationen finden Sie unter [Was ist Code Signing für AWS IoT?](#) und [Abrufen und Importieren eines Code-Signaturzertifikats](#).

CodeSigningCertificate/V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA : FALSE
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Schlüsselnutzung	Kritisch, Codesignatur
CRL-Verteilungspunkte*	[Passthrough von 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 API- und CSR-Pass-Through-Werte zu unterstützen.

CodeSigningCertificate_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA : FALSE
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Schlüsselnutzung	Kritisch, Codesignatur
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

CodeSigningCertificate_APIPassthrough/V1-Definition

Diese Vorlage ist identisch mit der CodeSigningCertificate Vorlage mit einem Unterschied: In dieser Vorlage AWS Private CA übergibt zusätzliche Erweiterungen über die API an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. In der Vorlage angegebene Erweiterungen überschreiben immer Erweiterungen in der API.

CodeSigningCertificate_APIPassthrough/V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA : FALSE
Autorisierungsschlüssel-ID	[SKI von CA Certificate]

X509v3-Parameter	Wert
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Schlüsselnutzung	Kritisch, Codesignatur
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

CodeSigningCertificate_CSRPassthrough/V1-Definition

Diese Vorlage ist identisch mit der Vorlage `CodeSigningCertificate`, bis auf einen Unterschied: AWS Private CA übergibt in dieser Vorlage zusätzliche Erweiterungen aus der Zertifikatsignieranforderung (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.

CodeSigningCertificate_CSRPassthrough/V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA: FALSE
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Schlüsselnutzung	Kritisch, Codesignatur
CRL-Verteilungspunkte*	[Passthrough von 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 Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webserver-Authentifizierung, TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von 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 API- und CSR-Pass-Through-Werte zu unterstützen.

EndEntityCertificate_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webserver-Authentifizierung, TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityCertificate_APIPassthrough/V1-Definition

Diese Vorlage ist identisch mit der EndEntityCertificate Vorlage mit einem Unterschied: In dieser Vorlage AWS Private CA übergibt zusätzliche Erweiterungen über die API an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. In der Vorlage angegebene Erweiterungen überschreiben immer Erweiterungen in der API.

EndEntityCertificate_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE

X509v3-Parameter	Wert
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselveschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webserver-Authentifizierung, TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityCertificate_CSRPassthrough/V1-Definition

Diese Vorlage ist identisch mit der Vorlage `EndEntityCertificate`, bis auf einen Unterschied: AWS Private CA übergibt in dieser Vorlage zusätzliche Erweiterungen aus der Zertifikatsignieranforderung (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.

EndEntityCertificate_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

X509v3-Parameter	Wert
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webserver-Authentifizierung, TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von 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 vom `EndEntityCertificate` nur im erweiterten Schlüsselnutzungswert, der sie auf die TLS-Webclient-Authentifizierung beschränkt.

EndEntityClientAuthCertificate/V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

*Zertifikatssperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatssperrlisten-Generierung konfiguriert ist.

EndEntityClientAuthCertificate_APICSRPassthrough/V1-Definition

Diese Vorlage erweitert EndEntityClientAuthCertificate/V1, um API- und CSR-Pass-Through-Werte zu unterstützen.

EndEntityClientAuthCertificate_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityClientAuthCertificate_APIPassthrough/V1-Definition

Diese Vorlage ist identisch mit der Vorlage EndEntityClientAuthCertificate, mit einem Unterschied: In dieser Vorlage AWS Private CA übergibt zusätzliche Erweiterungen über die API an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. In der Vorlage angegebene Erweiterungen überschreiben immer Erweiterungen in der API.

EndEntityClientAuthCertificate_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityClientAuthCertificate_CSRPassthrough/V1-Definition

Diese Vorlage ist identisch mit der Vorlage EndEntityClientAuthCertificate, mit einem Unterschied: AWS Private CA übergibt in dieser Vorlage zusätzliche Erweiterungen von der Zertifikatsignieranforderung (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_CSRPassthrough/V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA:FALSE

X509v3-Parameter	Wert
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webclient-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von 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 vom `EndEntityCertificate` nur im erweiterten Schlüsselnutzungswert, der sie auf die TLS-Webserverauthentifizierung beschränkt.

EndEntityServerAuthCertificate/V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webserver-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

*Zertifikatssperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatssperrlisten-Generierung konfiguriert ist.

EndEntityServerAuthCertificate_APICSRPassthrough/V1-Definition

Diese Vorlage erweitert EndEntityServerAuthCertificate/V1, um API- und CSR-Pass-Through-Werte zu unterstützen.

EndEntityServerAuthCertificate_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webserver-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityServerAuthCertificate_APIPassthrough/V1-Definition

Diese Vorlage ist identisch mit der Vorlage EndEntityServerAuthCertificate, mit einem Unterschied: In dieser Vorlage AWS Private CA übergibt zusätzliche Erweiterungen über die API an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. In der Vorlage angegebene Erweiterungen überschreiben immer Erweiterungen in der API.

EndEntityServerAuthCertificate_APIPassthrough/V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA:FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webserver-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

EndEntityServerAuthCertificate_CSRPassthrough/V1-Definition

Diese Vorlage ist identisch mit der Vorlage `EndEntityServerAuthCertificate`, mit einem Unterschied: AWS Private CA übergibt in dieser Vorlage zusätzliche Erweiterungen von der Zertifikatsignieranforderung (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_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA:FALSE

X509v3-Parameter	Wert
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, Schlüsselverschlüsselung
Erweiterte Schlüsselnutzung	TLS-Webserver-Authentifizierung
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

OCSPSigningCertificate/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 Schlüsselnutzung die OCSP-Signatur anstelle der Codesignatur angibt.

OCSPSigningCertificate/V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA : FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Schlüsselnutzung	Kritisch, OCSP-Signatur
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

*Zertifikatssperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatssperrlisten-Generierung konfiguriert ist.

OCSP SigningCertificate_APICSRPassthrough /V1-Definition

Diese Vorlage erweitert OCSP SigningCertificate/V1, um API- und CSR-Pass-Through-Werte zu unterstützen.

OCSP SigningCertificate_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA : FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Schlüsselnutzung	Kritisch, OCSP-Signatur
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

OCSP SigningCertificate_APIPassthrough /V1-Definition

Diese Vorlage ist identisch mit der Vorlage OCSPSigningCertificate, mit einem Unterschied: In dieser Vorlage AWS Private CA übergibt zusätzliche Erweiterungen über die API an das Zertifikat, wenn die Erweiterungen nicht in der Vorlage angegeben sind. In der Vorlage angegebene Erweiterungen überschreiben immer Erweiterungen in der API.

OCSP SigningCertificate_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	CA: FALSE
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Schlüsselnutzung	Kritisch, OCSP-Signatur
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

OCSP SigningCertificate_CSRPassthrough /V1-Definition

Diese Vorlage ist identisch mit der Vorlage OCSPSigningCertificate, mit einem Unterschied: AWS Private CA übergibt in dieser Vorlage zusätzliche Erweiterungen von der Zertifikatsignieranforderung (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.

OCSP SigningCertificate_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	CA: FALSE

X509v3-Parameter	Wert
Autorisierungsschlüssel-ID	[SKI von CA-Zertifikat]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur
Erweiterte Schlüsselnutzung	Kritisch, OCSP-Signatur
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

*Zertifikatsperrlisten-Verteilungspunkte werden nur dann in die Vorlage aufgenommen, wenn die Zertifizierungsstelle mit aktivierter Zertifikatsperrlisten-Generierung konfiguriert ist.

RootCACertificate/V1-Definition

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 die zukünftige Erweiterung der Hierarchie beeinträchtigen 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.

RootCACertificate/V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA : TRUE
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur keyCertSign, CRL-Zeichen

X509v3-Parameter	Wert
CRL-Verteilungspunkte	N/A

RootCACertificate _APIPassthrough/V1-Definition

Diese Vorlage erweitert RootCACertificate /V1, um API-Pass-Through-Werte zu unterstützen.

RootCACertificate _APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA : TRUE
Autorisierungsschlüssel-ID	[Pass-Through von API]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur keyCertSign, CRL-Zeichen
CRL-Verteilungspunkte*	N/A

BlankRootCACertificate _APIPassthrough /V1-Definition

Mit leeren Stammzertifikatsvorlagen können Sie Stammzertifikate mit nur X.509 grundlegenden Einschränkungen ausstellen. Dies ist das einfachste Stammzertifikat, das ausstellen AWS Private CA kann, aber es kann mithilfe der API-Struktur angepasst werden. Die Erweiterung der grundlegenden Einschränkungen definiert, ob es sich bei dem Zertifikat um ein CA-Zertifikat handelt oder nicht. Eine leere Stammzertifikatsvorlage erzwingt den Wert TRUE für grundlegende Einschränkungen, um sicherzustellen, dass ein Stammzertifizierungszertifikat ausgestellt wird.

Sie können leere Pass-Through-Stammvorlagen verwenden, um Stammzertifikate auszustellen, die bestimmte Werte für die Schlüsselnutzung (KU) erfordern. Beispielsweise könnte die Schlüsselnutzung keyCertSign und erfoderncRLSign, aber nicht digitalSignature.

Im Gegensatz zu der anderen nicht leeren Stamm-Pass-Through-Zertifikatvorlage erlauben leere Stammzertifikatvorlagen die Konfiguration der KU-Erweiterung, wobei KU einer der neun unterstützten Werte sein kann (`digitalSignature`, `nonRepudiation`, `keyEnciphermentdataEncipherment`, `keyAgreement`, `keyCertSign`, `cRLSign`, `encipherOnly`, und `decipherOnly`).

BlankRootCACertificate _APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA: TRUE
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

BlankRootCACertificate _PathLen0_APIPassthrough /V1-Definition

Allgemeine Informationen zu leeren Root-CA-Vorlagen finden Sie unter [???](#).

BlankRootCACertificate _PathLen0_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA: TRUE, pathLen: 0
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

BlankRootCACertificate _PathLen1_APIPassthrough/V1-Definition

Allgemeine Informationen zu leeren Root-CA-Vorlagen finden Sie unter [???](#).

BlankRootCACertificate_PathLen1_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 1
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

BlankRootCACertificate_PathLen2_APIPassthrough/V1-Definition

Allgemeine Informationen zu leeren Root-CA-Vorlagen finden Sie unter [???](#).

BlankRootCACertificate_PathLen2_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 2
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

BlankRootCACertificate_PathLen3_APIPassthrough/V1-Definition

Allgemeine Informationen zu leeren Root-CA-Vorlagen finden Sie unter [???](#).

BlankRootCACertificate_PathLen3_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 3

X509v3-Parameter	Wert
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

SubordinateCACertificate _PathLen0/V1-Definition

Diese Vorlage wird verwendet, um untergeordnete CA-Zertifikate mit einer Pfadlänge von auszustellen \emptyset . 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](#).

SubordinateCACertificate _PathLen0/V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: \emptyset
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

*CRL-Verteilungspunkte sind nur dann in Zertifikaten enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

SubordinateCACertificate_PathLen0_APICSRPassthrough /V1-Definition

Diese Vorlage erweitert SubordinateCACertificate_PathLen0/V1, um API- und CSR-Pass-Through-Werte zu unterstützen.

SubordinateCACertificate_PathLen0_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 0
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

SubordinateCACertificate_PathLen0_APIPassthrough /V1-Definition

Diese Vorlage erweitert SubordinateCACertificate_PathLen0/V1, um API-Pass-Through-Werte zu unterstützen.

SubordinateCACertificate_PathLen0_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 0

X509v3-Parameter	Wert
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

SubordinateCACertificate_PathLen0_CSRPassthrough /V1-Definition

Diese Vorlage ist identisch mit der Vorlage `SubordinateCACertificate_PathLen0`, bis auf einen Unterschied: AWS Private CA übergibt in dieser Vorlage zusätzliche Erweiterungen aus der Zertifikatsignieranforderung (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.

Note

Eine CSR, die benutzerdefinierte zusätzliche Erweiterungen enthält, muss außerhalb von `aws-private-ca` erstellt werden.

SubordinateCACertificate_PathLen0_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 0
Autorisierungsschlüssel-ID	[SKI von CA Certificate]

X509v3-Parameter	Wert
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatssperrlistengenerierung konfiguriert ist.

SubordinateCACertificate _PathLen1/V1-Definition

Diese Vorlage wird verwendet, um untergeordnete CA-Zertifikate mit einer Pfadlänge von auszustellen¹. CA-Zertifikate enthalten eine kritische Erweiterung der grundlegenden Einschränkungen, wobei das CA-Feld auf gesetzt ist, TRUE um festzulegen, dass das Zertifikat 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](#).

SubordinateCACertificate _PathLen1/V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 1
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

X509v3-Parameter	Wert
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatssperrlistengenerierung konfiguriert ist.

SubordinateCACertificate_PathLen1_APICSRPassthrough /V1-Definition

Diese Vorlage erweitert SubordinateCACertificate_PathLen1/V1, um API- und CSR-Pass-Through-Werte zu unterstützen.

SubordinateCACertificate_PathLen1_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 1
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

SubordinateCACertificate_PathLen1_APIPassthrough /V1-Definition

Diese Vorlage erweitert SubordinateCACertificate_PathLen0/V1, um API-Pass-Through-Werte zu unterstützen.

SubordinateCACertificate_PathLen1_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 1
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

SubordinateCACertificate_PathLen1_CSRPassthrough /V1-Definition

Diese Vorlage ist identisch mit der Vorlage SubordinateCACertificate_PathLen1, bis auf einen Unterschied: AWS Private CA übergibt in dieser Vorlage zusätzliche Erweiterungen aus der Zertifikatsignieranforderung (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.

Note

Eine CSR, die benutzerdefinierte zusätzliche Erweiterungen enthält, muss außerhalb von erstellt werden AWS Private CA.

SubordinateCACertificate_PathLen1_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 1
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Pass-Through von der CA-Konfiguration oder CSR]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatssperrlistengenerierung konfiguriert ist.

SubordinateCACertificate_PathLen2/V1-Definition

Diese Vorlage wird verwendet, um untergeordnete CA-Zertifikate mit einer Pfadlänge von 2 auszustellen. CA-Zertifikate enthalten eine kritische Erweiterung der grundlegenden Einschränkungen, wobei das CA-Feld auf gesetzt ist, TRUE um anzugeben, dass das Zertifikat 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](#).

SubordinateCACertificate_PathLen2/V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 2
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatssperrlistengenerierung konfiguriert ist.

SubordinateCACertificate_PathLen2_APICSRPassthrough /V1-Definition

Diese Vorlage erweitert SubordinateCACertificate_PathLen2/V1, um API- und CSR-Pass-Through-Werte zu unterstützen.

SubordinateCACertificate_PathLen2_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 2
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

X509v3-Parameter	Wert
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Pass-Through von der CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

SubordinateCACertificate_PathLen2_APIPassthrough/V1-Definition

Diese Vorlage erweitert SubordinateCACertificate_PathLen2/V1, um API-Pass-Through-Werte zu unterstützen.


SubordinateCACertificate_PathLen2_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 2
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

SubordinateCACertificate_PathLen2_CSRPassthrough /V1-Definition

Diese Vorlage ist identisch mit der Vorlage SubordinateCACertificate_PathLen2, bis auf einen Unterschied: AWS Private CA übergibt in dieser Vorlage zusätzliche Erweiterungen aus der Zertifikatsignieranforderung (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.

 Note

Eine CSR, die benutzerdefinierte zusätzliche Erweiterungen enthält, muss außerhalb von erstellt werden AWS Private CA.

SubordinateCACertificate_PathLen2_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektnamen	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA: TRUE, pathlen: 2
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Pass-Through von der CA-Konfiguration oder CSR]

*Zertifikatssperlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatssperlistengenerierung konfiguriert ist.

SubordinateCACertificate_PathLen3/V1-Definition

Diese Vorlage wird verwendet, um untergeordnete CA-Zertifikate mit einer Pfadlänge von 3 auszustellen. CA-Zertifikate enthalten eine kritische Erweiterung der grundlegenden Einschränkungen, wobei das CA-Feld auf gesetzt ist, TRUE um anzugeben, dass das Zertifikat 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](#).

SubordinateCACertificate_PathLen3/V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathLen: 3
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatssperrlistengenerierung konfiguriert ist.

SubordinateCACertificate_PathLen3_APICSRPassthrough /V1-Definition

Diese Vorlage erweitert SubordinateCACertificate_PathLen3/V1, um API- und CSR-Pass-Through-Werte zu unterstützen.

SubordinateCACertificate_PathLen3_APICSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 3
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

SubordinateCACertificate_PathLen3_APIPassthrough /V1-Definition

Diese Vorlage erweitert SubordinateCACertificate_PathLen3/V1, um API-Pass-Through-Werte zu unterstützen.

SubordinateCACertificate_PathLen3_APIPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von API oder CSR]
Betreff	[Pass-Through von API oder CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 3
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

X509v3-Parameter	Wert
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration]

* CRL-Verteilungspunkte sind nur dann in der Vorlage enthalten, wenn die CA mit aktivierter CRL-Generierung konfiguriert ist.

SubordinateCACertificate_PathLen3_CSRPassthrough /V1-Definition

Diese Vorlage ist identisch mit der Vorlage SubordinateCACertificate_PathLen3, bis auf einen Unterschied: AWS Private CA übergibt in dieser Vorlage zusätzliche Erweiterungen aus der Zertifikatsignieranforderung (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.

Note

Eine CSR, die benutzerdefinierte zusätzliche Erweiterungen enthält, muss außerhalb von erstellt werdenAWS Private CA.

SubordinateCACertificate_PathLen3_CSRPassthrough /V1

X509v3-Parameter	Wert
Alternativer Subjektname	[Pass-Through von CSR]
Betreff	[Pass-Through von CSR]
Grundlegende Einschränkungen	kritisch, CA:TRUE, pathlen: 3
Autorisierungsschlüssel-ID	[SKI von CA Certificate]
Kennung des Subjektschlüssels	[Abgeleitet aus CSR]

X509v3-Parameter	Wert
Schlüsselnutzung	Kritische, digitale Signatur, keyCertSign , CRL-Zeichen
CRL-Verteilungspunkte*	[Passthrough von CA-Konfiguration oder CSR]

*Zertifikatssperrlisten-Verteilungspunkte sind nur dann in Zertifikaten enthalten, die mit dieser Vorlage ausgestellt wurden, wenn die Zertifizierungsstelle mit aktivierter Zertifikatssperrlistengenerierung konfiguriert ist.

Verwenden der AWS Private CA-API (Java-Beispiele)

Sie können die AWS Private Certificate Authority API für die programmgesteuerte Interaktion mit dem Service durch Senden von HTTP-Anforderungen verwenden. Der Service gibt HTTP-Antworten zurück. Weitere Informationen finden Sie in der [AWS Private Certificate Authority API-Referenz zu](#) .

Zusätzlich zu der HTTP-API können Sie die AWS-SDKs und Befehlszeilen-Tools für die Interaktion mit AWS Private CA verwenden. 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 [DescribeCertificateAuthorityAuditReport](#) Operationen [GetCertificateAuthorityCsr](#), [GetCertificate](#) und unterstützen Waiter. 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 [Waiters im AWS SDK for Java](#) im [AWS -Entwickler-Blog](#).

Themen

- [Programmgesteuertes Erstellen und Aktivieren einer Root-CA](#)
- [Programmgesteuertes Erstellen und Aktivieren einer untergeordneten Zertifizierungsstelle](#)
- [CreateCertificateAuthority](#)
- [Verwenden von CreateCertificateAuthority zur Unterstützung von Active Directory](#)
- [CreateCertificateAuthorityAuditReport](#)
- [CreatePermission](#)
- [DeleteCertificateAuthority](#)
- [DeletePermission](#)
- [DeletePolicy](#)
- [DescribeCertificateAuthority](#)
- [DescribeCertificateAuthorityAuditReport](#)
- [GetCertificate](#)
- [GetCertificateAuthorityCertificate](#)
- [GetCertificateAuthorityCsr](#)
- [GetPolicy](#)
- [ImportCertificateAuthorityCertificate](#)

- [IssueCertificate](#)
- [ListCertificateAuthorities](#)
- [ListPermissions](#)
- [ListTags](#)
- [PutPolicy](#)
- [RestoreCertificateAuthority](#)
- [RevokeCertificate](#)
- [TagCertificateAuthorities](#)
- [UntagCertificateAuthority](#)
- [UpdateCertificateAuthority](#)
- [Erstellen von Zertifizierungsstellen und Zertifikaten mit benutzerdefinierten Betreffnamen](#)
- [Erstellen Sie Zertifikate mit benutzerdefinierten Erweiterungen](#)

Programmgesteuertes Erstellen und Aktivieren einer Root-CA

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

- [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.setEnabled(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 createCARquest = new
CreateCertificateAuthorityRequest();
    createCARquest.withCertificateAuthorityConfiguration(configCA);
```

```
createCARRequest.withRevocationConfiguration(revokeConfig);
createCARRequest.withIdempotencyToken("123987");
createCARRequest.withCertificateAuthorityType(CAtype);

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

// Retrieve the ARN of the private CA.
String rootCAArn = createCARResult.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);
}
}
```

Programmgesteuertes Erstellen und Aktivieren einer untergeordneten Zertifizierungsstelle

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.setEnabled(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 createCARRequest = new
CreateCertificateAuthorityRequest();
    createCARRequest.withCertificateAuthorityConfiguration(configCA);
    createCARRequest.withRevocationConfiguration(revokeConfig);
    createCARRequest.withIdempotencyToken("123987");
    createCARRequest.withCertificateAuthorityType(CAtype);

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

// Retrieve the ARN of the private CA.
String subordinateCAArn = createCARResult.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 [-CreateCertificateAuthority](#) Operation 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 zum Signieren verwendet
- X.500-Themeninformationen

Die CRL-Konfiguration legt Folgendes fest:

- Die Ablaufzeit der Zertifikatsperrliste in Tagen (die Gültigkeitsdauer der CRL)
- Der Amazon S3-Bucket, der die CRL enthält
- 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.setEnabled(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
```

Verwenden von CreateCertificateAuthority zur Unterstützung von Active Directory

Das folgende Java-Beispiel zeigt, wie Sie die [-CreateCertificateAuthority](#) Operation verwenden, um eine CA zu erstellen, die im Enterprise NTAAuth-Speicher von Microsoft Active Directory (AD) installiert werden kann.

Die -Operation erstellt eine private Stammzertifizierungsstelle (CA) unter Verwendung von benutzerdefinierten Objektkennungen (OIDs). Weitere Informationen und ein AWS CLI Beispiel für eine gleichwertige Operation finden Sie unter [Erstellen einer CA 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 **
AWSACMPClient 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 AWSACMPClient 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(
format.",",
"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\\.aws\\credentials), and is in valid
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 createCARRequest = new
CreateCertificateAuthorityRequest();
    createCARRequest.withCertificateAuthorityConfiguration(configCA);
    createCARRequest.withIdempotencyToken("123987");
    createCARRequest.withCertificateAuthorityType(CAtype);

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

    // Retrieve the ARN of the private CA.
    String rootCAArn = createCARResult.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 [CreateCertificateAuthorityAuditReport](#) Operation 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-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 [CreatePermission](#) Operation verwendet wird.

Die Operation weist einem bestimmten AWS-Service-Prinzipal 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 ACMIssueCertificate-Serviceprinzipal (`ListPermissions`) alle möglichen Berechtigungen (`GetCertificate`, und) von der CA zuweisen `acm.amazonaws.com`. Sie finden den ARN einer CA, indem Sie die [ListCertificateAuthorities](#) Funktion aufrufen.

Sobald eine Berechtigung erstellt wurde, können Sie sie mit der [ListPermissions](#) Funktion überprüfen oder mit der [DeletePermission](#) Funktion 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 [-DeleteCertificateAuthority](#) Operation verwendet wird.

Diese Operation löscht die private Zertifizierungsstelle (CA), die Sie mit der [CreateCertificateAuthority](#) Operation erstellt haben. Für die Operation `DeleteCertificateAuthority` müssen Sie einen ARN angeben, damit die Zertifizierungsstelle gelöscht werden kann. Sie finden den ARN, indem Sie die [-ListCertificateAuthorities](#) Operation 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 [-UpdateCertificateAuthority](#) Operation aufrufen und den Parameter `Status` auf `setzenDISABLED` setzen. 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 [Wiederherstellen einer Zertifizierungsstelle](#).

Ein Java-Beispiel, das Ihnen zeigt, wie Sie die [-RestoreCertificateAuthority](#) Operation verwenden, finden Sie unter [RestoreCertificateAuthority](#).

```
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 request 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 [DeletePermission](#) Operation verwendet wird.

Die -Operation löscht Berechtigungen, die eine private Zertifizierungsstelle mithilfe der [CreatePermissions](#) Operation an einen -AWSService-Prinzipal delegiert hat. Sie finden den ARN einer CA, indem Sie die [ListCertificateAuthorities](#) Funktion aufrufen. Sie können die Berechtigungen überprüfen, die eine Zertifizierungsstelle erteilt hat, indem Sie die [ListPermissions](#) Funktion 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.
        AWSStaticCredentialsProvider 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 [-DeletePolicy](#) Operation verwendet wird.

Die -Operation löscht die ressourcenbasierte Richtlinie, die einer privaten Zertifizierungsstelle zugeordnet ist. Eine ressourcenbasierte Richtlinie wird verwendet, um die kontoübergreifende gemeinsame Nutzung von Zertifizierungsstellen zu ermöglichen. Sie finden den ARN einer privaten Zertifizierungsstelle, indem Sie die [ListCertificateAuthorities](#) Aktion aufrufen.

Zu den zugehörigen API-Aktionen gehören [PutPolicy](#) und [GetPolicy](#).

```
package com.amazonaws.samples;

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

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

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

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

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

import java.util.ArrayList;

public class CreatePermission {

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

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

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

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



```
        .build();

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

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

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

req.setActions(permissions);

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

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

DescribeCertificateAuthority

Das folgende Java-Beispiel zeigt, wie die [DescribeCertificateAuthority](#) Operation 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 Zertifizierungsstelle ist aktiv.
- **DISABLED** – Ihre private Zertifizierungsstelle wurde deaktiviert.
- **EXPIRED** – Ihr privates CA-Zertifikat ist abgelaufen.
- **FAILED** – Ihre private Zertifizierungsstelle kann nicht erstellt werden.
- **DELETED** – Ihre private Zertifizierungsstelle befindet sich innerhalb des Wiederherstellungszeitraums, nach dem sie dauerhaft gelöscht wird.

```
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 [DescribeCertificateAuthorityAuditReport](#) Operation verwendet wird.

Die Operation listet Informationen zu einem bestimmten Auditbericht auf, den Sie durch Aufrufen der [CreateCertificateAuthorityAuditReport](#) Operation 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 [-GetCertificate](#) Operation verwendet wird.

Mit dieser Funktion wird ein Zertifikat von Ihrer privaten Zertifizierungsstelle abgerufen. Der ARN des Zertifikats wird zurückgegeben, wenn Sie den [IssueCertificate](#) Vorgang 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 die [-CreateCertificateAuthorityAuditReport](#) Operation aufrufen, um einen Bericht zu erstellen, der Informationen zu allen Zertifikaten enthält, die von Ihrer privaten Zertifizierungsstelle ausgestellt und widerrufen 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 [GetCertificateAuthorityCertificate](#) Operation 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 Zeichenfolgen 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 [-GetCertificateAuthorityCsr](#) Operation verwendet wird.

Diese Operation ruft die Zertifikatsignierungsanforderung (CSR) für Ihre private CA ab. Die CSR wird erstellt, wenn Sie den [CreateCertificateAuthority](#) Vorgang 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 die [-ImportCertificateAuthorityCertificate](#) Operation aufrufen. 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 [-GetPolicy](#) Operation verwendet wird.

Die [-Operation](#) ruft die ressourcenbasierte Richtlinie ab, die einer privaten Zertifizierungsstelle zugeordnet ist. Eine ressourcenbasierte Richtlinie wird verwendet, um die kontoübergreifende gemeinsame Nutzung von Zertifizierungsstellen zu ermöglichen. Sie finden den ARN einer privaten Zertifizierungsstelle, indem Sie die [ListCertificateAuthorities](#) Aktion aufrufen.

Zu den zugehörigen API-Aktionen gehören [PutPolicy](#) und [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 [ImportCertificateAuthorityCertificate](#) Operation verwendet wird.

Mit dieser Operation wird Ihr signiertes privates CA-Zertifikat in AWS Private CA importiert. Bevor Sie diesen Vorgang aufrufen können, müssen Sie die private Zertifizierungsstelle erstellen, indem Sie den [CreateCertificateAuthority](#) Vorgang aufrufen. Sie müssen dann eine Zertifikatsignierungsanforderung (Certificate Signing Request, CSR) generieren, indem Sie den [GetCertificateAuthorityCsr](#) Vorgang 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 Zertifikatskette 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 [IssueCertificate](#) Operation verwendet wird.

Dieser Vorgang verwendet Ihre private Zertifizierungsstelle (CA), um ein Endentitätszertifikat auszustellen. Diese Operation gibt den Amazon-Ressourcennamen (ARN) des Zertifikats zurück. Sie können das Zertifikat abrufen, indem Sie die aufrufen [GetCertificate](#) und den ARN angeben.

Note

Für den [IssueCertificate](#) Vorgang müssen Sie eine Zertifikatsvorlage angeben. In diesem Beispiel wird die EndEntityCertificate/V1 Vorlage verwendet. Informationen zu allen verfügbaren Vorlagen finden Sie unter [Grundlegendes zu Zertifikatsvorlagen](#).

```
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 AWSSStaticCredentialsProvider(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

Im folgenden Java-Beispiel wird gezeigt, wie Sie [ListCertificateAuthorities](#) Verwendungsvorgang.

Bei diesem Vorgang werden die privaten Zertifizierungsstellen (CAs) aufgelistet, die Sie mit [CreateCertificateAuthority](#) Verwendungsvorgang.

```
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.
        AWSStaticCredentialsProvider 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.setMaxResults(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 [ListPermissions](#) Operation verwendet wird.

In dieser Operation werden ggf. die Berechtigungen aufgelistet, die Ihrer privaten Zertifizierungsstelle zugewiesen wurden. Berechtigungen, einschließlich GetCertificate, und ListPermissions, können einem IssueCertificate-AWSService-Prinzipal mit der [CreatePermission](#) Operation zugewiesen und mit der [DeletePermissions](#) Operation 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,
  Principal: acm.amazonaws.com,
  Permissions: {
    ISSUE_CERTIFICATE,
    GET_CERTIFICATE,
    DELETE, CERTIFICATE
  },
  SourceAccount: account
]]
```

ListTags

Das folgende Java-Beispiel zeigt, wie die [ListTags](#) Operation verwendet wird.

Diese Operation listet die Tags auf, sofern vorhanden, die zu Ihrer privaten Zertifizierungsstelle gehören. Tags sind Kennzeichnungen, die Sie verwenden können, um Ihre CAs zu identifizieren und zu organisieren. Jedes Tag besteht aus einem Schlüssel und einem optionalen Wert. Rufen Sie die [TagCertificateAuthority](#) Operation auf, um Ihrer CA ein oder mehrere Tags hinzuzufügen. Rufen Sie die Operation [UntagCertificateAuthority](#) auf, um Tags zu entfernen.

```
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 AWSSStaticCredentialsProvider(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 [-PutPolicy](#) Operation verwendet wird.

Die `-` Operation fügt eine ressourcenbasierte Richtlinie an eine private Zertifizierungsstelle an und ermöglicht so die kontoübergreifende Freigabe. Wenn ein Prinzipal, der sich in einem anderen AWS Konto befindet, durch eine Richtlinie autorisiert wird, kann er private Endentitätszertifikate mit einer privaten Zertifizierungsstelle ausstellen und erneuern, die ihm nicht gehört. Sie finden den ARN einer privaten Zertifizierungsstelle, indem Sie die [ListCertificateAuthorities](#) Aktion aufrufen. Beispiele für - Richtlinien finden Sie in der AWS Private CA Anleitung zu ressourcenbasierten Richtlinien. <https://docs.aws.amazon.com/privateca/latest/userguide/pca-rbp.html>

Sobald eine Richtlinie an eine Zertifizierungsstelle angefügt ist, können Sie sie mit der [-GetPolicy](#) Aktion überprüfen oder mit der [-DeletePolicy](#) Aktion 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 = AWSACMPAClientBuilder.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 [RestoreCertificateAuthority](#) Operation 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 [Wiederherstellen 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 [-RevokeCertificate](#) Operation verwendet wird.

Dieser Vorgang widerruft ein Zertifikat, das Sie durch Aufrufen des [IssueCertificate](#) Vorgangs ausgestellt haben. Wenn Sie beim Erstellen oder Aktualisieren Ihrer privaten Zertifizierungsstelle eine Zertifikatsperrliste (CRL) aktiviert haben, sind Informationen zu den widerrufenen Zertifikaten in der CRL enthalten. AWS Private CA schreibt die CRL in einen von Ihnen angegebenen Amazon S3-Bucket. Weitere Informationen finden Sie in der [-CrlConfiguration](#) Struktur.

```
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 [-TagCertificateAuthority](#) Operation verwendet wird.

Diese Operation fügt Ihrer privaten CA einen oder mehrere Tags hinzu. Tags sind Kennzeichnungen, die Sie verwenden können, um Ihre AWS-Ressourcen zu identifizieren und zu organisieren. 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 CAs zu filtern, können Sie dasselbe Tag auch auf mehrere private CAs anwenden. Um ein oder mehrere Tags zu entfernen, verwenden Sie die [UntagCertificateAuthority](#) Operation. Rufen Sie die Operation auf, [ListTags](#) 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 = AWSACMPClientBuilder.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 [-UntagCertificateAuthority](#) Operation 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. Um einer privaten Zertifizierungsstelle Tags hinzuzufügen, verwenden Sie die [-TagCertificateAuthority](#) Operation. Rufen Sie die Operation auf, [ListTags](#) 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 [UpdateCertificateAuthority](#) Operation 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.setEnabled(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 von Zertifizierungsstellen und Zertifikaten mit benutzerdefinierten Betreffnamen

Die [CustomAttribute](#) object ermöglicht es Administratoren, benutzerdefinierte Objektkennungen (OIDs) an private Zertifizierungsstellen und Zertifikate zu übergeben. Benutzerdefinierte OIDs können verwendet werden, um spezielle Subjektnamenhierarchien zu erstellen, die die Struktur und die Bedürfnisse Ihrer Organisation widerspiegeln. Angepasste Zertifikate müssen mit einem der [ApiPassthrough](#) Vorlagen. Weitere Informationen zu Vorlagen finden Sie unter [Vorlagenvarianten](#). Weitere Informationen zur Verwendung von finden Sie unter [Ausstellen privater Endentitätszertifikate](#) und [Verfahren zum Erstellen einer Zertifizierungsstelle \(CLI\)](#).

Du kannst nicht benutzen `StandardAttributes` in Verbindung mit `CustomAttributes`. Sie können jedoch Standard-OIDs als Teil eines `CustomAttributes`. Die OIDs mit Standardnamen werden in der folgenden Tabelle aufgeführt:

Betreff	Objekt-ID
Land	2.5.4.6
CommonName	2.5.4.3
DistinguishedNameQualifier	2.5.4.46
GenerationQualifier	2.5.4.4
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
Status	2.5.4.8
Nachname	2.5.4.4
Titel	2.5.4.12

Themen

- [CA erstellen mit CustomAttribute](#)
- [Zertifikat ausstellen 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("ABCDEFGG"),
        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);
}
}
```

Zertifikat ausstellen 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("ABCDEFGH"),
    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);
}
}
```

Erstellen Sie Zertifikate mit benutzerdefinierten Erweiterungen

Die [CustomExtension](#) object ermöglicht es Administratoren, benutzerdefinierte X.509-Erweiterungen in privaten Zertifikaten festzulegen. Angepasste Zertifikate müssen mit einem der ApiPassthrough vorlagen. Weitere Informationen zu Vorlagen finden Sie unter [Vorlagenvarianten](#). Weitere Informationen zur Verwendung von benutzerdefinierten Erweiterungen finden Sie unter [Ausstellen privater Endentitätszertifikate](#).

Themen

- [Aktivieren Sie eine untergeordnete CA mit dem NameConstraints Ausdehnung](#)
- [Ausstellen eines Zertifikats mit der Erweiterung der QC-Anweisung](#)

Aktivieren Sie eine untergeordnete CA mit dem NameConstraints Ausdehnung

```
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 createCARRequest = new
CreateCertificateAuthorityRequest();
    createCARRequest.withCertificateAuthorityConfiguration(configCA);
    createCARRequest.withRevocationConfiguration(revokeConfig);
    createCARRequest.withIdempotencyToken("1234");
    createCARRequest.withCertificateAuthorityType(caType);

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

// Retrieve the ARN of the private CA.
String subordinateCAArn = createCARResult.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);
    }
}
```

Ausstellen eines Zertifikats mit der Erweiterung der QC-Anweisung

```
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(bspAIVector);

        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(bspASVector);

        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(bspPIVector);

        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(bspICVector);

    ASN1EncodableVector pspSeqVector = new ASN1EncodableVector(4);
    pspSeqVector.add(bspPISeq);
    pspSeqVector.add(bspICSeq);
    pspSeqVector.add(bspASSeq);
    pspSeqVector.add(bspAISEq);
    DERSequence pspSeq = new DERSequence(bspSeqVector);

    ASN1EncodableVector pspVector = new ASN1EncodableVector(3);
    pspVector.add(bspSeq);
    pspVector.add(new DERUTF8String("Your Financial Authority"));
    pspVector.add(new DERUTF8String("AB-CD"));
    DERSequence psp = new DERSequence(bspVector);
    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 = AWSACMPClientBuilder.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);
}
}
```

Verwenden der AWS Private CA API zur Implementierung des Boler-Standards (Java-Beispiele)

Sie können die AWS Private Certificate Authority-API verwenden, um Zertifikate zu erstellen, die dem [Boler-Konnektivitätsstandard](#) entsprechen. Der Steuerer gibt Zertifikatkonfigurationen an, die die Sicherheit und Konsistenz von Internet-of-Dings (IoT)-Geräten auf mehreren Engineering-Plattformen verbessern. Weitere Informationen zu Boler finden Sie unter buildwithmatter.com.

Die Java-Beispiele in diesem Abschnitt interagieren mit dem Service, indem HTTP-Anfragen gesendet werden. Der Service gibt HTTP-Antworten zurück. Weitere Informationen finden Sie in der [AWS Private Certificate Authority API-Referenz zu](#) .

Zusätzlich zu der HTTP-API können Sie die AWS-SDKs und Befehlszeilen-Tools für die Interaktion mit AWS Private CA verwenden. 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 [DescribeCertificateAuthorityAuditReport](#) Operationen [GetCertificateAuthorityCsr](#), [GetCertificate](#) und unterstützen Waiter. 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 [Waiters im AWS SDK for Java](#) im [AWS -Entwickler-Blog](#).

Die im Oktober 2023 veröffentlichte Version 1.2 unterstützt den DAC-Widerruf mithilfe von Certificate Revocation Lists (CRLs). Um Sie bei der Einhaltung des aktuellen Boler-Standards zu unterstützen, legen Sie beim Aktivieren des CRL-Widerrufs für CAs, die Boler-Zertifikate ausstellen, im `-CrlConfigurationObjekt` in der `-CrlDistributionPointExtensionConfigurationStruktur` `OmitExtension` auf `festtrue`.

In der Regel betten CAs den CRL Distribution Point (CDP) in die von ihnen ausgestellten Zertifikate ein, sodass die vertrauenden Parteien, die die Zertifikatskettenvalidierung durchführen, die CRL abrufen und den Zertifikatstatus überprüfen können. In In Inser wird der CDP-URI nicht in Zertifikate geschrieben. Stattdessen rufen Benutzer CDPs aus dem Boler Distributed Compliance Ledger (DCL), dem vertrauenswürdigen Boler-Datenspeicher, ab. Sie müssen den CDP-URI in die Boler DCL hochladen, damit er bei der Validierung von DACs erkannt werden kann. Weitere Informationen zur Bestimmung der CDP-URI finden Sie unter [Ermitteln des CRL Distribution Point \(CDP\) -URI](#) . Weitere Informationen zu Boler finden Sie in der [DCL-Dokumentation zu Boler](#) .

Themen

- [Aktivieren einer Product Attestation Authority \(PAA\)](#)
- [Aktivieren eines Product Attestation Intermediate \(PAI\)](#)
- [Erstellen eines Gerätebescheinigungszertifikats \(DAC\)](#)
- [Aktivieren Sie eine Root CA für Node Operational Certificates \(NOC\).](#)
- [Aktivieren einer untergeordneten Zertifizierungsstelle für Betriebszertifikate für Knoten \(NOC\)](#)
- [Erstellen eines Node Operational Certificate \(NOC\)](#)

Aktivieren einer Product Attestation Authority (PAA)

Dieses Java-Beispiel zeigt, wie Sie mit der [RootCACertificate _APIPassthrough/V1-Definition](#) Vorlage [einPAA](#)-Zertifikat (Root CA) für die Produktbescheinigung erstellen und installieren. Die AuthorityKeyIdentifier (AKI)-Erweiterung ist für PAAs optional. Um eine AKI festzulegen, müssen Sie einen Base64-encoded AKI-Wert generieren und ihn durch eine übergeben CustomExtension.

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

- [CreateCertificateAuthority](#)
- [GetCertificateAuthorityCsr](#)
- [IssueCertificate](#)
- [GetCertificate](#)
- [ImportCertificateAuthorityCertificate](#)

Wenn Probleme auftreten, finden Sie weitere Informationen unter [Verwenden Sie den Matter-Standard](#) 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 createCARrequest = new
CreateCertificateAuthorityRequest();
    createCARrequest.withCertificateAuthorityConfiguration(configCA);
    createCARrequest.withIdempotencyToken("123987");
    createCARrequest.withCertificateAuthorityType(CAtype);
    createCARrequest.withRevocationConfiguration(revokeConfig);

    // Create the private CA.
    CreateCertificateAuthorityResult createCARresult = null;
    try {
        createCARresult = client.createCertificateAuthority(createCARrequest);
    } 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 eines Product Attestation Intermediate (PAI)

Dieses Java-Beispiel zeigt, wie Sie die [BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1-Definition](#) Vorlage verwenden, um ein [PAI-Zertifikat](#) (Subordinate CA) für die Produktbescheinigung zu erstellen und zu installieren. Sie müssen einen Base64-encoded KeyUsage Wert generieren und ihn durch eine übergeben CustomExtension.

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

- [CreateCertificateAuthority](#)
- [GetCertificateAuthorityCsr](#)
- [IssueCertificate](#)
- [GetCertificate](#)
- [ImportCertificateAuthorityCertificate](#)

- [GetCertificateAuthorityCertificate](#)

Wenn Probleme auftreten, finden Sie weitere Informationen unter [Verwenden Sie den Matter-Standard](#) 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 **
AWSACMPClient client = ClientBuilder(endpointRegion);
String rootCertificate = GetCertificateAuthorityCertificate(paaArn, client);
```

```
String subordinateCAArn = CreateCertificateAuthority(configCA, crtConfigure,
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 createCARrequest = new
CreateCertificateAuthorityRequest();
    createCARrequest.withCertificateAuthorityConfiguration(configCA);
    createCARrequest.withIdempotencyToken("123987");
    createCARrequest.withCertificateAuthorityType(CAtype);
    createCARrequest.withRevocationConfiguration(revokeConfig);

    // Create the private CA.
    CreateCertificateAuthorityResult createCARresult = 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 eines Gerätebescheinigungszertifikats (DAC)

Dieses Java-Beispiel zeigt, wie Sie die Vorlage

[BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough /V1](#) verwenden, um ein [Boler](#) Device Attestation Certificate zu erstellen. Sie müssen einen Base64-encoded KeyUsage Wert generieren und ihn durch eine übergeben CustomExtension.

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

- [IssueCertificate](#)

Wenn Probleme auftreten, finden Sie weitere Informationen unter [Verwenden Sie den Matter-Standard](#) 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 AWSSStaticCredentialsProvider(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 [RootCACertificate_APIPassthrough/V1-Definition](#) Vorlage verwenden, um ein [Boler](#)-Root-CA-Zertifikat zu erstellen und zu installieren, um NOCs auszustellen. Die AuthorityKeyIdentifier (AKI)-Erweiterung ist für NOC-Root-CA-Zertifikate optional. Um eine AKI

festzulegen, müssen Sie einen Base64-encoded AKI-Wert generieren und ihn durch eine übergeben CustomExtension.

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

- [CreateCertificateAuthority](#)
- [GetCertificateAuthorityCsr](#)
- [IssueCertificate](#)
- [GetCertificate](#)
- [ImportCertificateAuthorityCertificate](#)

Wenn Probleme auftreten, finden Sie weitere Informationen unter [Verwenden Sie den Matter-Standard](#) 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 createCARRequest = new
CreateCertificateAuthorityRequest();
    createCARRequest.withCertificateAuthorityConfiguration(configCA);
    createCARRequest.withIdempotencyToken("123987");
    createCARRequest.withCertificateAuthorityType(CAtype);

    // Create the private CA.
    CreateCertificateAuthorityResult createCARResult = null;
    try {
        createCARResult = client.createCertificateAuthority(createCARRequest);
    } 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 einer untergeordneten Zertifizierungsstelle für Betriebszertifikate für Knoten (NOC)

Dieses Java-Beispiel zeigt, wie Sie die [BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1-Definition](#) Vorlage verwenden, um ein [microSDer](#)-Subordinate-CA-Zertifikat auszustellen und zu installieren, um NOCs auszustellen. Sie müssen einen Base64-encoded KeyUsage Wert generieren und ihn durch eine übergeben CustomExtension.

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

- [CreateCertificateAuthority](#)
- [GetCertificateAuthorityCsr](#)
- [IssueCertificate](#)
- [GetCertificate](#)
- [ImportCertificateAuthorityCertificate](#)

- [GetCertificateAuthorityCertificate](#)

Wenn Probleme auftreten, finden Sie weitere Informationen unter [Verwenden Sie den Matter-Standard](#) 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 createCARRequest = new
CreateCertificateAuthorityRequest();
    createCARRequest.withCertificateAuthorityConfiguration(configCA);
    createCARRequest.withIdempotencyToken("123987");
    createCARRequest.withCertificateAuthorityType(CAtype);

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

    // Retrieve the ARN of the private CA.
    String subordinateCAArn = createCARResult.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 eines Node Operational Certificate (NOC)

Dieses Java-Beispiel zeigt, wie Sie die Vorlage

[BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough /V1](#) verwenden, um ein Betriebszertifikat für [Boler](#)-Knoten zu erstellen. Sie müssen einen Base64-encoded KeyUsage Wert generieren und ihn durch eine übergeben CustomExtension.

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

- [IssueCertificate](#)

Wenn Probleme auftreten, finden Sie weitere Informationen unter [Verwenden Sie den Matter-Standard](#) 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 IssueNodeOperationalCertificate {
    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_APIassthrough/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);
}
}
```

Verwenden der AWS Private CA API zur Implementierung des mDL-Standards (Mobile Drive License) (Java-Beispiele)

Sie können die AWS Private Certificate Authority API verwenden, um Zertifikate zu erstellen, die dem [ISO/IEC-Standard für mobile Führerscheine \(mDL\)](#) entsprechen. Dieser Standard legt Schnittstellenspezifikationen für die Implementierung eines Führerscheins in Verbindung mit einem Mobilgerät fest, einschließlich Zertifikatkonfigurationen.

Die Java-Beispiele in diesem Abschnitt interagieren mit dem Service, indem HTTP-Anfragen gesendet werden. Der Service gibt HTTP-Antworten zurück. Weitere Informationen finden Sie in der [AWS Private Certificate Authority-API-Referenz](#).

Zusätzlich zur HTTP-API können Sie auch die AWS -SDKs und -AWS CLITools verwenden, um zu verwalten AWS Private CA. Wir empfehlen die Verwendung des SDK oder AWS CLI über die HTTP-API. 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 [DescribeCertificateAuthorityAuditReport](#) Operationen [GetCertificateAuthorityCsr](#), [GetCertificate](#) und unterstützen Waiter. 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 und unter [Waiters im AWS -SDK für Java](#) im [AWS -Entwickler-Blog](#).

Themen

- [Aktivieren eines IAM-Zertifikats \(ausstellende Zertifizierungsstelle\)](#)
- [Erstellen eines Dokumentsigniererzertifikats](#)

Aktivieren eines IAM-Zertifikats (ausstellende Zertifizierungsstelle)

Dieses Java-Beispiel zeigt, wie Sie die [BlankRootCACertificate_PathLen0_APIPassthrough / V1-Definition](#) Vorlage verwenden, um ein [ISO/IEC mDL standard](#) -konformes Zertifizierungsstellenzertifikat (IACA) zu erstellen und zu installieren. Sie müssen base64-kodierte Werte für `KeyUsage`, `IssuerAlternativeName` und generieren `CRLDistributionPoint` und diese durch `übergebenCustomExtensions`.

Note

Das IACA-Link-Zertifikat richtet einen Vertrauenspfad vom alten IACA-Stammzertifikat zum neuen IACA-Stammzertifikat ein. Die ausstellende Stelle kann während des IACA-Neuschlüsselprozesses ein IACA-Link-Zertifikat generieren und verteilen. Sie können ein IACA-Link-Zertifikat nicht mithilfe eines IACA-Stammzertifikats mit `pathLen=0` festgelegt ausstellen.

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 createCARrequest = new
CreateCertificateAuthorityRequest()
        .withCertificateAuthorityConfiguration(configCA)
        .withIdempotencyToken("123987")
        .withCertificateAuthorityType(CAtype);

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

    // Get the ARN of the private CA.
    String rootCAArn = createCARresult.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
    }

    // Add custom extension to api-passthrough
    Extensions extensions = new Extensions()
        .withCustomExtensions(Arrays.asList(keyUsageCustomExtension,
            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 eines Dokumentsigniererzertifikats

Dieses Java-Beispiel zeigt, wie Sie die Vorlage [BlankEndEntityCertificate_APIPassthrough_V1](#) verwenden, um ein [ISO/IEC mDL standard](#) -konformes Dokumentsigniererzertifikat zu erstellen. Sie müssen base64-kodierte Werte für KeyUsage, IssuerAlternativeName und generieren CRLDistributionPoint und diese durch übergeben CustomExtensions.

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

- [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 = AWSACMPAClientBuilder.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);
}
}
```

Extern signierte private CA-Zertifikate

Wenn die Stammvertrauensstellung Ihrer privaten CA-Hierarchie eine CA außerhalb von AWS Private CA sein muss, können Sie eine eigene Stamm-CA 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 erworbene Zertifizierungsstelle verwenden, um ein privates untergeordnetes Zertifizierungsstellenzertifikat zu signieren, das AWS Private CA verwaltet.

Note

Verfahren zum Erstellen oder Abrufen eines externen Vertrauensdiensteanbieters liegen außerhalb des Geltungsbereichs dieses Handbuchs.

Wenn Sie eine externe übergeordnete Zertifizierungsstelle mit verwenden, AWS Private CA können Sie Einschränkungen des CA-[Namens erzwingen, wie im Abschnitt Namensbeschränkungen](#) von RFC 5280 definiert. Namensbeschränkungen bieten CA-Administratoren die Möglichkeit, Subjektnamen in Zertifikaten einzuschränken.

Wenn Sie ein privates untergeordnetes CA-Zertifikat mit einer externen CA signieren möchten, müssen Sie drei Aufgaben ausführen, bevor Sie eine funktionierende CA in haben AWS Private CA:

1. Generieren Sie eine Zertifikatsignierungsanforderung (Certificate Signing Request, CSR).
2. Senden Sie die CSR an Ihre externe Signaturstelle und geben Sie sie mit einem signierten Zertifikat und einer Zertifikatkette zurück.
3. Installieren Sie ein signiertes Zertifikat in AWS Private CA.

In den folgenden Verfahren wird beschrieben, wie Sie diese Aufgaben entweder mit der AWS Management Console oder der ausführen AWS CLI.

So rufen Sie ein extern signiertes CA-Zertifikat ab und installieren es (Konsole)

1. (Optional) Wenn Sie noch nicht auf der Detailseite der Zertifizierungsstelle sind, öffnen Sie die - AWS Private CA Konsole unter <https://console.aws.amazon.com/acm-pca/home>. 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 Untergeordnetes CA-Zertifikat installieren zu öffnen.
3. Wählen Sie auf der Seite Untergeordnetes CA-Zertifikat installieren unter CA-Typ auswählen die Option Externe private CA aus.
4. Unter CSR für diese CA zeigt die Konsole den Base64-encoded ASCII-Text der CSR an. Sie können den Text mit der Schaltfläche Kopieren kopieren oder CSR in eine Datei exportieren auswählen und lokal speichern.

 Note

Das genaue Format des CSR-Texts muss beim Coping und Einfügen beibehalten werden.

5. Wenn Sie die Offline-Schritte nicht sofort ausführen können, um ein signiertes Zertifikat von Ihrer externen Signaturstelle zu erhalten, können Sie die Seite schließen und zu ihr zurückkehren, sobald Sie über ein signiertes Zertifikat und eine Zertifikatkette verfügen.

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

- Fügen Sie den Base64-encoded ASCII-Text Ihres Zertifikatstexts und Ihrer Zertifikatkette in die entsprechenden Textfelder ein.
 - Wählen Sie Hochladen, um den Zertifikatstext und die Zertifikatkette aus lokalen Dateien in die entsprechenden Textfelder zu laden.
6. Wählen Sie Bestätigen und installieren aus.

So rufen Sie ein extern signiertes CA-Zertifikat (CLI) ab und installieren es

1. Verwenden Sie den [get-certificate-authority-csr](#) Befehl , um die Zertifikatsignierungsanforderung (Certificate Signing Request, CSR) für Ihre private Zertifizierungsstelle abzurufen. Wenn Sie die Zertifikatsignierungsanforderung an Ihre Anzeige senden möchten, verwenden Sie die `--output text`-Option, um CR-/LF-Zeichen vom 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 mit Base64 PEM codierte signierte Zertifikat und die Zertifikatkette enthalten.
3. Verwenden Sie den [import-certificate-authority-certificate](#) Befehl , um die private CA-Zertifikatsdatei und die Chain-Datei in zu importierenAWS Private CA.

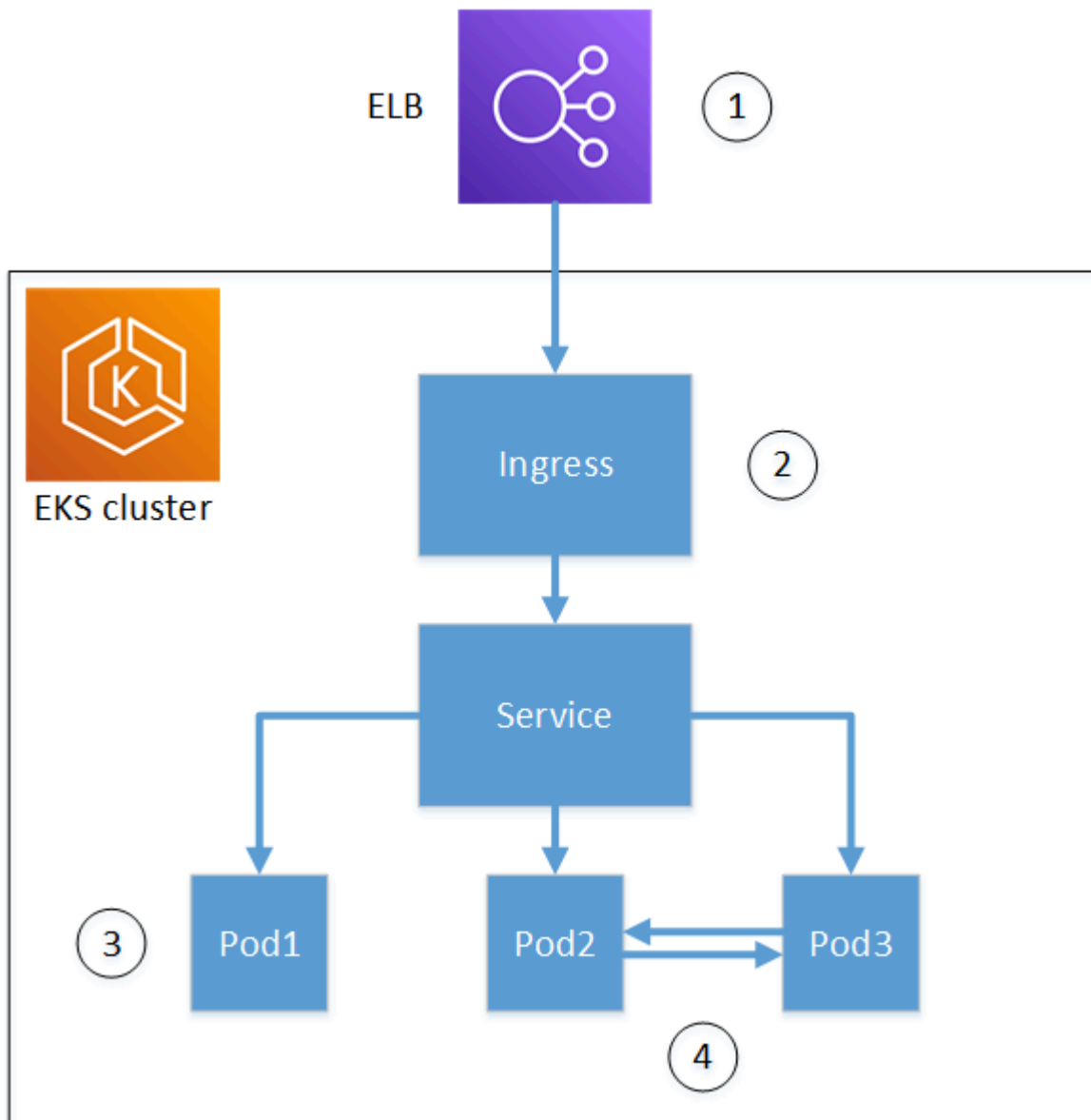
```
$ aws acm-pca import-certificate-authority-certificate \
--certificate-authority-arn arn:aws:acm-pca:region:account:\
certificate-authority/12345678-1234-1234-1234-123456789012 \
--certificate file://C:\example_ca_cert.pem \
--certificate-chain file://C:\example_ca_cert_chain.pem
```

Sichern von Kubernetes mit AWS Private CA

Kubernetes-Container und -Anwendungen verwenden digitale Zertifikate, um eine sichere Authentifizierung und Verschlüsselung über TLS bereitzustellen. Eine weit verbreitete Lösung für die Verwaltung des Lebenszyklus von TLS-Zertifikaten in Kubernetes ist [cert-manager](#), ein Add-on für Kubernetes, das Zertifikate anfordert, sie an Kubernetes-Container verteilt und die Zertifikatserneuerung automatisiert.

AWS Private CA bietet ein Open-Source-Plug-In für cert-manager, [aws-privateca-issuer](#), für cert-manager-Benutzer, die eine CA einrichten möchten, ohne private Schlüssel im Cluster zu speichern. Benutzer mit regulatorischen Anforderungen für die Kontrolle des Zugriffs auf und die Prüfung ihres CA-Betriebs können diese Lösung verwenden, um die Auditierbarkeit zu verbessern und die Compliance zu unterstützen. Sie können das AWS Private CA Issuer-Plugin mit Amazon Elastic Kubernetes Service (Amazon EKS), einem selbstverwalteten Kubernetes auf AWS oder in On-Premises-Kubernetes verwenden.

Das folgende Diagramm zeigt einige der Optionen, die für die Verwendung von TLS in einem Amazon-EKS-Cluster verfügbar sind. Dieser Beispiel-Cluster, der verschiedene Ressourcen enthält, befindet sich hinter einem Load Balancer. Die Zahlen identifizieren mögliche Endpunkte für die TLS-gesicherte Kommunikation, einschließlich des externen Load Balancers, des Eingangscontrollers, eines einzelnen Pods innerhalb eines Services und eines Pairs von Pods, die sicher miteinander kommunizieren.



1. Beendigung am Load Balancer.

Elastic Load Balancing (ELB) ist ein AWS Certificate Manager integrierter Service, was bedeutet, dass Sie ACM mit einer privaten Zertifizierungsstelle bereitstellen, ein Zertifikat damit signieren und es mithilfe der ELB-Konsole installieren können. Diese Lösung ermöglicht die verschlüsselte Kommunikation zwischen einem Remote-Client und dem Load Balancer. Daten werden unverschlüsselt an den EKS-Cluster übergeben. Alternativ können Sie einem Nicht-AWS-Load Balancer ein privates Zertifikat bereitstellen, um TLS zu beenden.

2. Beendigung am Kubernetes-Controller für eingehenden Datenverkehr.

Der Ingress-Controller befindet sich im EKS-Cluster als nativer Load Balancer und Router. Wenn Sie sowohl cert-manager als auch installiert aws-privateca-issuer und den Cluster mit einer privaten Zertifizierungsstelle bereitgestellt haben, kann Kubernetes ein signiertes TLS-Zertifikat auf dem Controller installieren, sodass er als Endpunkt für die externe Kommunikation des Clusters dienen kann. Die Kommunikation zwischen dem Load Balancer und dem Ingress Controller wird verschlüsselt, und nach dem Eingang werden Daten unverschlüsselt an die Ressourcen des Clusters übergeben.

3. Beendigung an einem Pod.

Jeder Pod ist eine Gruppe von einem oder mehreren Containern, die Speicher- und Netzwerkressourcen gemeinsam nutzen. Wenn Sie sowohl cert-manager als auch installiert aws-privateca-issuer und den Cluster mit einer privaten Zertifizierungsstelle bereitgestellt haben, kann Kubernetes nach Bedarf ein signiertes TLS-Zertifikat auf Pods installieren. Eine TLS-Verbindung, die auf einem Pod endet, ist standardmäßig für andere Pods im Cluster nicht verfügbar.

4. Sichere Kommunikation zwischen Pods.

Sie können auch mehrere Pods mit Zertifikaten bereitstellen, die es ihnen ermöglichen, miteinander zu kommunizieren. Die folgenden Szenarien sind möglich:

- Bereitstellung mit von Kubernetes generierten, selbstsignierten Zertifikaten. Dies sichert die Kommunikation zwischen Pods, selbstsignierte Zertifikate erfüllen jedoch nicht die HIPAA- oder FIPS-Anforderungen.
- Bereitstellung mit Zertifikaten, die von einer privaten Zertifizierungsstelle signiert wurden. Wie in den Nummern 2 und 3 oben erfordert dies die Installation von cert-manager und aws-privateca-issuer und die Bereitstellung des Clusters mit einer privaten Zertifizierungsstelle. Kubernetes kann dann nach Bedarf signierte TLS-Zertifikate auf den Pods installieren.

Kontoübergreifende Verwendung des cert-manager

Administratoren mit kontoübergreifendem Zugriff auf eine CA können cert-manager verwenden, um einen Kubernetes-Cluster bereitzustellen. Weitere Informationen finden Sie unter [Bewährte Methoden für die Sicherheit für den kontoübergreifenden Zugriff auf private CAs](#).

Note

Nur bestimmte AWS Private CA Zertifikatsvorlagen können in kontoübergreifenden Szenarien verwendet werden. Eine Liste der verfügbaren Vorlagen [the section called “Unterstützte Zertifikatsvorlagen”](#) finden Sie unter .

Unterstützte Zertifikatsvorlagen

In der folgenden Tabelle sind AWS Private CA Vorlagen aufgeführt, die mit cert-manager verwendet werden können, um einen Kubernetes-Cluster bereitzustellen.

Für Kubernetes unterstützte Vorlagen	Unterstützung für die kontoübergreifende Verwendung
BlankEndEntityCertificate_CSRPassthrough/V1-Definition	
CodeSigningCertificate/V1-Definition	
EndEntityCertificate/V1-Definition	✓
EndEntityClientAuthCertificate/V1-Definition	✓
EndEntityServerAuthCertificate/V1-Definition	✓
OCSPSigningCertificate/V1-Definition	

Beispiellösungen

Die folgenden Integrationslösungen zeigen, wie Sie den Zugriff auf AWS Private CA auf auf einem Amazon-EKS-Cluster konfigurieren.

- [TLS-fähige Kubernetes-Cluster mit AWS Private CA und Amazon EKS](#)
- [Einrichten der end-to-end TLS-Verschlüsselung auf Amazon EKS mit dem neuen AWS Load Balancer Controller](#)

AWS Private CA Connector for Active Directory

Was ist AWS Private CA Connector für Active Directory?

AWS Private CA kann Zertifikate ausstellen und verwalten, die von benötigt werdenAWS Managed Microsoft AD. Mit dem AWS Private CA Connector für Active Directory (Connector für AD) können Sie On-Premises-Unternehmen oder andere Drittanbieter-CAs durch eine verwaltete private CA ersetzen, die Sie besitzen, und die Zertifikatregistrierung für Benutzer, Gruppen und Maschinen bereitstellen, die von Ihrem AD verwaltet werden.

Sie können den Connector für AD mit verwendenAWS Managed Microsoft AD, um die On-Premises-Infrastruktur zu eliminieren, indem Sie Ihr AD und Ihre Infrastruktur für öffentliche Schlüssel in die Cloud migrieren. Für Kunden, die AWS Private CA mit ihrem On-Premises-AD verwenden möchten, ist diese Funktion auch in AWS Managed Microsoft AD Connector integriert.

Themen

- [Sind Sie ein erstmaliger Konnektor für AD-Benutzer?](#)
- [Zugreifen auf Connector für AD](#)
- [Preise für Connector für AD](#)

Sind Sie ein erstmaliger Konnektor für AD-Benutzer?

Wenn Sie Connector für AD zum ersten Mal verwenden, empfehlen wir Ihnen, zunächst die folgenden Abschnitte zu lesen:

- [Was ist AWS Private CA?](#)
- [Was ist AWS Directory Service?](#)

Zugreifen auf Connector für AD

Sie können über die Konsole, AWS CLI und APIs auf Connector for AD zugreifen. Sie können über die AWS Private CA Konsole, über Ihre AWS Directory Service Konsole oder durch Suchen nach Connector für AD in der AWS Management Console Suchleiste Zugriff auf den Connector in der Konsole erhalten.

Preise für Connector für AD

Connector für AD wird AWS Private CA ohne zusätzliche Kosten als Feature von angeboten. Sie zahlen nur für die privaten Zertifizierungsstellen und die Zertifikate, die Sie über sie ausstellen.

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

Erste Schritte mit AWS Private CA Connector für Active Directory

Mit AWS Private CA Connector für Active Directory können Sie Zertifikate von Ihrer privaten Zertifizierungsstelle für Ihre Active-Directory-Objekte zur Authentifizierung und Verschlüsselung ausstellen. Wenn Sie einen Connector erstellen, AWS Private Certificate Authority erstellt 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.

Themen

- [Connector für AD-Voraussetzungen](#)
- [Erstellen eines Konnektors](#)
- [Konfigurieren von AD-Richtlinien](#)
- [Erstellen einer Vorlage](#)
- [Verwalten von AD-Gruppenberechtigungen](#)

Connector für AD-Voraussetzungen

Für Connector for AD sind folgende Informationen erforderlich.

Um einen Connector für AD zu erstellen, müssen Sie ein AWS Private Certificate Authority (CA) und ein Verzeichnis einrichten. Anschließend müssen Sie die private Zertifizierungsstelle und das Verzeichnis für den Service Connector for AD freigeben. Sie müssen auch Sicherheitsgruppen und IAM-Richtlinien korrekt festlegen, um einen Connector zu erstellen.

AWS Private CA

Richten Sie ein AWS Private CA für die Ausstellung von Zertifikaten an Ihre Verzeichnisobjekte ein. Weitere Informationen finden Sie unter [Verwaltung privater Zertifizierungsstellen](#).

Der AWS Private CA muss sich im Active Status befinden, um einen Connector für AD zu erstellen. Der Betreffname der privaten Zertifizierungsstelle muss einen gemeinsamen Namen enthalten. Die Erstellung des Konnektors schlägt fehl, wenn Sie versuchen, einen Konnektor mit einer privaten Zertifizierungsstelle ohne gemeinsamen Namen zu erstellen.

Active Directory

Zusätzlich zu einem benötigen AWS Private CASie ein Active Directory in einer Virtual Private Cloud (VPC). Connector für AD unterstützt die folgenden Verzeichnistypen, die von angeboten werdenAWS Directory Service:

- [AWS Managed Microsoft Active Directory](#): Mit können AWS Directory Service Sie Microsoft Active Directory (AD) als verwalteten Service ausführen. wird AWS Directory Service for Microsoft Active Directory auch als bezeichnet AWS Managed Microsoft ADund wird von Windows Server 2019 unterstützt. Mit können AWS Managed Microsoft ADSie verzeichnishaftige Workloads in der ausführenAWS Cloud, einschließlich Microsoft Sharepoint und benutzerdefinierter .Net- und SQL Server-basierter Anwendungen.
- [Active Directory Connector](#) : AD Connector ist ein Verzeichnis-Gateway, das Verzeichnisanforderungen an Ihr lokales Microsoft Active Directory umleiten kann, ohne Informationen in der Cloud zwischenspeichern zu müssen. AD Connector unterstützt die Verbindung zu einer auf Amazon EC2 gehosteten Domain

Note

Die Registrierung von Domain-Controllern wird nicht unterstützt, wenn der Connector für AD mit verwendet wirdAWS Managed Microsoft AD.

Servicekonto

Wenn Sie Directory Service AD Connector verwenden, müssen Sie zusätzliche Berechtigungen an das Servicekonto delegieren. Legen Sie die Zugriffssteuerungsliste (ACL) im Servicekonto fest, um Folgendes zu ermöglichen:

- Hinzufügen und Entfernen eines Service-Prinzipalnamens (SPN) zu sich selbst
- 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 NT AuthCertificates Certification Authority (CA)-Objekt. Hinweis: Wenn das NTAAuthCertificates -CA-Objekt vorhanden ist, müssen Sie Berechtigungen dafür delegieren. Wenn das Objekt nicht vorhanden ist, müssen Sie die Möglichkeit delegieren, untergeordnete Objekte im Container Public Key Services zu erstellen.

```
#objects
CN=NTAuthCertificates,CN=Public Key Services,CN=Services,CN=Configuration
```

Note

Wenn Sie verwenden AWS Managed Microsoft AD, werden die zusätzlichen Berechtigungen automatisch delegiert, wenn Sie den Connector for AD-Service mit Ihrem Verzeichnis autorisieren. Sie können diesen erforderlichen Schritt überspringen.

Sie können dieses PowerShell Skript verwenden, um die zusätzlichen Berechtigungen zu delegieren. Es wird das NT-AuthCertificates Zertifizierungsstellenobjekt erstellen. Ersetzen Sie „myconnectoraccount“ durch den Namen des Servicekontos.

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

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

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

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

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

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

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

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

```

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

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

```

IAM-Richtlinie

Um einen Connector für AD zu erstellen, benötigen Sie eine IAM-Richtlinie, mit der Sie Connector-Ressourcen erstellen, Ihre private CA für den Connector für AD-Service freigeben und den Connector für AD-Service für Ihr Verzeichnis autorisieren können.

Dies ist ein Beispiel für eine vom Benutzer verwaltete Richtlinie:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "pca-connector-ad:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "acm-pca:DescribeCertificateAuthority",
        "acm-pca:GetCertificate",
        "acm-pca:GetCertificateAuthorityCertificate",
        "acm-pca:ListCertificateAuthorities",
        "acm-pca:ListTags",
        "acm-pca:PutPolicy"
      ]
    }
  ],

```

```

    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "acm-pca:IssueCertificate",
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
BlankEndEntityCertificate_ApiPassthrough/V*"
      },
      "ForAnyValue:StringEquals": {
        "aws:CalledVia": "pca-connector-ad.amazonaws.com"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "ds:AuthorizeApplication",
      "ds:DescribeDirectories",
      "ds:ListTagsForResource",
      "ds:UnauthorizeApplication",
      "ds:UpdateAuthorizedApplication"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateVpcEndpoint",
      "ec2:DescribeSecurityGroups",
      "ec2:DescribeSubnets",
      "ec2:DescribeVpcEndpoints",
      "ec2:DescribeVpcs",
      "ec2>DeleteVpcEndpoints"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeTags",
      "ec2>DeleteTags",

```

```

        "ec2:CreateTags"
      ],
      "Resource": "arn:*:ec2:*:*:vpc-endpoint/*"
    }
  ]
}

```

Connector für AD erfordert zusätzliche AWS RAM Berechtigungen, sowohl für die Verwendung der Konsole als auch für die Befehlszeile.

```

{
  "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": "*"
    }
  ]
}

```

AWS Private CA Mit Connector für AD teilen

Sie müssen Ihr AWS Private CA mithilfe der Service-Prinzipal-Freigabe für den AWS Resource Access Manager Connectors-Service freigeben.

Wenn Sie einen Connector in der AWS Konsole erstellen, wird die Ressourcenfreigabe automatisch für Sie erstellt.

Wenn Sie eine Ressourcenfreigabe mit der erstellenAWS 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 -Service-Prinzipal, der aufruft, CreateConnector verfügt über Berechtigungen zur Ausstellung von Zertifikaten auf der PCA. Um zu verhindern, dass Service-Prinzipale, die Connector für AD verwenden, allgemeinen Zugriff auf Ihre -AWS Private CARessourcen haben, beschränken Sie ihre Berechtigungen mit CalledVia.

Autorisieren von Connector für AD mit Ihrem Verzeichnis

Sie autorisieren den Connector for AD-Service mit Ihrem Verzeichnis, damit der Connector mit Ihrem Verzeichnis kommunizieren kann. Um den Service Connector für AD zu autorisieren, erstellen Sie eine Verzeichnisregistrierung. Weitere Informationen zum Erstellen einer Verzeichnisregistrierung finden Sie unter . [Verwaltung von Verzeichnisregistrierungen](#)

Sicherheitsgruppen

Die Kommunikation zwischen Ihrer VPC und dem Connector für AD erfolgt über AWS PrivateLink, was eine Sicherheitsgruppe(n) mit eingehenden Regeln erfordert, die Port 443 TCP und UDP 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).

Erstellen eines Konnektors

Anweisungen finden Sie im Verfahren [Erstellen eines Konnektors](#)

Konfigurieren von AD-Richtlinien

Connector für AD kann die Konfiguration des Gruppenrichtlinienobjekts (GPO) des Kunden nicht anzeigen oder verwalten. Das GPO steuert die Weiterleitung von AD-Anfragen an den AWS Private CA oder an andere Authentifizierungs- oder Zertifikatvending-Server des Kunden. Eine ungültige GPO-Konfiguration kann dazu führen, dass Ihre Anfragen falsch weitergeleitet werden. Es liegt an Kunden, die Konfiguration des Connectors für AD zu konfigurieren und zu testen.

Gruppenrichtlinien sind einem Connector zugeordnet, und ein können Sie wählen, mehrere Connectors für ein einzelnes AD zu erstellen. Es liegt an Ihnen, die Zugriffskontrolle für jeden Konnektor zu verwalten, wenn seine Gruppenrichtlinienkonfigurationen unterschiedlich sind.

Die Sicherheit der Aufrufe auf Datenebene hängt von Kerberos und Ihrer VPC-Konfiguration ab. Jeder, der Zugriff auf die VPC hat, kann Datenebenenaufrufe tätigen, solange er beim entsprechenden AD authentifiziert ist. Dies liegt außerhalb der Grenze von AWSAuth und die Verwaltung von Autorisierung und Authentifizierung liegt an Ihnen, dem Kunden.

Führen Sie in Active Directory die folgenden Schritte aus, um ein GPO zu erstellen, das auf den URI verweist, der beim Erstellen eines Konnektors generiert wurde. Dieser Schritt ist erforderlich, um Connector für AD über die Konsole oder die Befehlszeile zu verwenden.

Konfigurieren Sie GPOs .

1. Öffnen Sie Server Manager auf dem DC
2. Gehen Sie zu Tools und wählen Sie in der oberen rechten Ecke der Konsole Gruppenrichtlinienverwaltung aus.
3. Gehen Sie zu Forest > Domains . Wählen Sie Ihren Domänennamen aus und klicken Sie mit der rechten Maustaste auf Ihre Domäne. Wählen Sie GPO in dieser Domain erstellen aus, verknüpfen Sie sie hier ... und geben Sie PCA GPO als Namen ein.
4. Das neu erstellte GPO wird jetzt unter Ihrem Domänennamen aufgeführt.
5. Wählen Sie PCA GPO und dann Bearbeiten aus. Wenn ein Dialogfeld mit der Warnmeldung geöffnet wird. Dies ist ein Link und diese Änderungen werden global weitergegeben. Bestätigen

- Sie die Nachricht, 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. Wechseln Sie zum Objekttyp und wählen Sie Certificate Services Client – Certificate Registration Policy
 8. Ändern Sie in den Optionen Configuration Model in Enabled .
 9. Vergewissern Sie sich, dass die Active-Directory-Registrierungsrichtlinie aktiviert und aktiviert ist. Wählen Sie Hinzufügen aus.
 10. Das Fenster Zertifikatregistrierungsrichtlinienserver sollte geöffnet werden.
 11. Geben Sie den Server-Endpunkt der Zertifikatsregistrierungsrichtlinie ein, der beim Erstellen Ihres Connectors generiert wurde, im Feld Registrierungsserver-Richtlinien-URI eingeben.
 12. Behalten Sie den Authentifizierungstyp als Windows-integriert bei.
 13. Wählen Sie Validate aus. Nachdem die Validierung erfolgreich war, wählen Sie Hinzufügen aus. Das Dialogfeld wird geschlossen.
 14. Kehren Sie zu Certificate Services Client – Certificate Registration Policy zurück und aktivieren Sie das Kontrollkästchen neben dem neu erstellten Konnektor, um sicherzustellen, dass der Konnektor die Standardregistrierungsrichtlinie ist
 15. Wählen Sie Active Directory-Registrierungsrichtlinie und dann Entfernen aus.
 16. Wählen Sie im Bestätigungsdiaologfeld Ja aus, um die LDAP-basierte Authentifizierung zu löschen.
 17. Wählen Sie Anwenden und OK im Fenster Certificate Services Client > Certificate Registration Policy und schließen Sie es.
 18. Gehen Sie zum Ordner Richtlinien für öffentliche Schlüssel und wählen Sie Certificate Services Client – Auto-Enrollment aus.
 19. Ändern Sie die Option Konfigurationsmodell auf Aktiviert.
 20. Vergewissern Sie sich, dass sowohl abgelaufene Zertifikate verlängern als auch Zertifikate aktualisieren überprüft wurden. Behalten Sie die anderen Einstellungen unverändert bei.
 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. Befolgen Sie die von Schritt 6 bis Schritt 21 beschriebenen Verfahren, um die Richtlinien für öffentliche Schlüssel für die Benutzerkonfiguration zu konfigurieren.

Sobald Sie mit der Konfiguration von GPOs und Richtlinien für öffentliche Schlüssel fertig sind, fordern Objekte in der Domain Zertifikate von AWS Private CA Connector für AD an und erhalten Zertifikate, die von ausgestellt wurdenAWS Private CA.

Erstellen einer Vorlage

Anweisungen finden Sie im Verfahren [Eine Connector-Vorlage erstellen](#)

Verwalten von AD-Gruppenberechtigungen

Anweisungen finden Sie im Verfahren [AD-Gruppen und Berechtigungen für Vorlagen verwalten](#)

AWS Private CA Konnektor für Active Directory-Prozeduren

Die Verfahren in diesem Abschnitt beschreiben, wie Sie Connectors erstellen, Vorlagen konfigurieren und in AWS Private CA und Active Directory integrieren. Sie können diese Operationen über die Konsole AWS Private CA Connector für AD, über den Abschnitt Connector für AD der AWS CLI oder über die AWS Private CA Connector für AD API ausführen.

Note

Obwohl AWS Private CA Connector für AD eng in integriert istAWS Private CA, verfügen die beiden Services über separate APIs. Weitere Informationen finden Sie in der [AWS Private Certificate Authority -API-Referenz](#) und in der [AWS Private CA API-Referenz zum Konnektor für Active Directory](#).

Verfahren

- [Erstellen eines Konnektors](#)
- [Eine Connector-Vorlage erstellen](#)
- [Konnektoren für Active Directory auflisten](#)
- [Connector-Vorlagen auflisten](#)
- [Konnektordetails anzeigen](#)
- [Details zur Connector-Vorlage anzeigen](#)

- [Verwaltung von Verzeichnisregistrierungen](#)
- [AD-Gruppen und Berechtigungen für Vorlagen verwalten](#)
- [Konfiguration des Dienstprinzipalnamens](#)
- [Tagging-Connector für AD-Ressourcen](#)

Erstellen eines Konnektors

Gehen Sie wie folgt vor, um einen Connector mit der Konsole, der Befehlszeile oder der API für AWS Private CA Connector für Active Directory zu erstellen.

Erstellen eines Konnektors (Konsole)

Führen Sie die folgenden Verfahren aus, um einen Connector mithilfe der AWS Konsole zu erstellen und zu konfigurieren.

Aufgaben

- [Öffnen der Konsole](#)
- [Öffnen Sie Konnektor erstellen](#)
- [Auswählen oder Erstellen eines Verzeichnisses](#)
- [Auswählen einer privaten Zertifizierungsstelle](#)
- [Tagging konfigurieren](#)
- [Überprüfen und erstellen](#)

Öffnen der Konsole

Melden Sie sich bei Ihrem -AWS-Konto an und öffnen Sie die -AWS Private CAConnector-for-Active-Directory-Konsole unter <https://console.aws.amazon.com/pca-connector-ad/home>.

Öffnen Sie Konnektor erstellen

Wählen Sie auf der ersten Service-Landingpage oder auf der Seite Connectors für Active Directory die Option Konnektor erstellen aus.

Auswählen oder Erstellen eines Verzeichnisses

Geben Sie auf der Seite Create Private CA Connector for Active Directory Informationen im Abschnitt Active Directory ein.

- Wählen Sie unter Active-Directory-Typ auswählen einen der beiden verfügbaren Typen aus:
 - AWS Directory Service for Microsoft Active Directory – Gibt ein Active Directory an, das von verwaltet wird AWS Directory Service.
 - On-Premises Active Directory mit AWS AD Connector – Verwendet AD Connector, um auf ein Active Directory zuzugreifen, das Sie On-Premises hosten.
- Wählen Sie unter Verzeichnis auswählen Ihr Verzeichnis aus der Liste aus.

Alternativ können Sie Verzeichnis erstellen auswählen, wodurch die AWS Directory Service Konsole in einem neuen Fenster geöffnet wird. Wenn Sie mit der Erstellung eines neuen Verzeichnisses fertig sind, kehren Sie zur Konsole AWS Private CA Connector für Active Directory zurück und aktualisieren Sie die Liste der Verzeichnisse. Ihr neues Verzeichnis sollte zur Auswahl verfügbar sein.

Note

Beachten Sie beim Erstellen eines Verzeichnisses, dass Connector für AD nur die folgenden Verzeichnistypen unterstützt, die in der AWS Directory Service Konsole angeboten werden:

- AWS Managed Microsoft AD
- AD Connector

- Wählen Sie unter Sicherheitsgruppen für VPC-Endpunkt auswählen eine Sicherheitsgruppe aus der Liste aus.

Alternativ können Sie Sicherheitsgruppe erstellen auswählen, wodurch die Amazon EC2-Konsole in einem neuen Fenster zur Seite Sicherheitsgruppe erstellen geöffnet wird. Wenn Sie mit dem Erstellen einer Sicherheitsgruppe fertig sind, kehren Sie zur Konsole AWS Private CA Connector für Active Directory zurück und aktualisieren Sie die Liste der Sicherheitsgruppen. Ihre neue Sicherheitsgruppe sollte zur Auswahl verfügbar sein.

Auswählen einer privaten Zertifizierungsstelle

Wählen Sie im Abschnitt Private Zertifizierungsstelle eine private Zertifizierungsstelle aus der Liste aus.

Alternativ können Sie Private CA erstellen auswählen, wodurch die AWS Private CA Konsole in einem neuen Fenster zur Seite Private Zertifizierungsstellen geöffnet wird. Wenn Sie mit der Erstellung einer

CA fertig sind, kehren Sie zur AWS Private CA Konsole Connector für Active Directory zurück und aktualisieren Sie die Liste der CAs. Ihre neue Zertifizierungsstelle sollte zur Auswahl verfügbar sein.

Tagging konfigurieren

Im Bereich Tags – optional können Sie Metadaten auf Ihre AD-Ressource anwenden und entfernen. Tags sind Schlüssel-Wert-Zeichenfolgenpaare, bei denen der Schlüssel für die Ressource eindeutig sein muss und der Wert optional ist. Der Bereich zeigt alle vorhandenen Tags für die Ressource in einer Tabelle an. 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 Feld Schlüssel und optional das Feld Wert aus. Wählen Sie Änderungen speichern, um das Tag anzuwenden.
- Wählen Sie die Schaltfläche Entfernen neben einem Tag, um es zum Löschen zu markieren, und wählen Sie zur Bestätigung Änderungen speichern.

Überprüfen und erstellen

Nachdem Sie die erforderlichen Informationen angegeben und Ihre Auswahl überprüft haben, wählen Sie Konnektor erstellen aus. Dadurch wird die Detailseite Connectors für Active Directory geöffnet, auf der den Fortschritt Ihres Connectors während seiner Erstellung anzeigen kann.

Nachdem die Erstellung eines Konnektors abgeschlossen ist, weisen Sie ihm einen Service-Prinzipalnamen zu.

Erstellen eines Connectors für Active Directory (AWS CLI)

Um einen Connector für Active Directory mit der CLI zu erstellen, verwenden Sie den Befehl [create-connector](#) im Abschnitt AWS Private CA Connector für Active Directory der AWS CLI.

Erstellen eines Konnektors für Active Directory (API)

Um einen Connector für Active Directory mit der API zu erstellen, verwenden Sie die [CreateConnector](#)-Aktion in der AWS Private CA Connector für Active Directory API.

Eine Connector-Vorlage erstellen

Eine Connector-Vorlage erstellen (Konsole)

Gehen Sie wie folgt vor, um eine Connector-Vorlage mit dem zu erstellen und zu konfigurierenAWSKonsole.

Aufgaben

- [Konsole öffnen](#)
- [Stecker wählen](#)
- [Suchen Sie den Abschnitt „Vorlage“](#)
- [Methode zur Erstellung von Vorlagen](#)
- [Einstellungen für Vorlagen](#)
- [Konfigurieren Sie die Zertifikatseinstellungen](#)
- [Konfigurieren Sie die Einstellungen für die Bearbeitung von Anfragen und die Registrierung](#)
- [Konfigurieren Sie wichtige Nutzungserweiterungen](#)
- [Weisen Sie Anwendungsrichtlinien zu](#)
- [Konfigurieren Sie benutzerdefinierte Anwendungsrichtlinien](#)
- [Konfigurieren Sie die Kryptografieeinstellungen](#)
- [Gruppen und Berechtigungen konfigurieren](#)
- [Konfigurieren Sie ablösende Vorlagen](#)
- [Tagging konfigurieren](#)
- [Überprüfen und erstellen](#)

Konsole öffnen

Melde dich bei deinem anAWSKonto und öffnen Sie dasAWS Private CAConnector für die Active Directory-Konsole unter<https://console.aws.amazon.com/pca-connector-ad/home>.

Stecker wählen

Wählen Sie einen Konnektor aus derKonnektoren für Active Directorylisten Sie auf und wählen Sie dannDetails ansehen.

Suchen Sie den Abschnitt „Vorlage“

Suchen Sie auf der Detailseite für den Connector nach Vorlagen Abschnitt und dann wählen Vorlage erstellen.

Methode zur Erstellung von Vorlagen

Auf der Vorlage erstellen Seite, in der Methode zur Erstellung von Vorlagen Wählen Sie im Abschnitt eine der Methodenoptionen aus.

- Beginnen Sie mit einer vordefinierten Vorlage (Standard) — Wählen Sie aus einer Liste vordefinierter Vorlagen für AD-Anwendungen aus:
 - Codesignatur
 - Computer
 - Domänencontroller-Authentifizierung
 - EFS-Wiederherstellungsagent
 - Registrierungs-Agent
 - 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, die Sie zuvor erstellt haben.
- Beginnen Sie mit einer leeren Vorlage— Wählen Sie diese Option, um mit der Erstellung einer komplett neuen Vorlage zu beginnen.

Einstellungen für Vorlagen

In der Vorlageneinstellungen Geben Sie im Abschnitt die folgenden Informationen ein:

- Name der Vorlage— Der Name der Vorlage

- Version des Vorlagenschemas— Die Schemaversion der Vorlage. Die Schemaversion wirkt sich wie folgt auf die Verfügbarkeit der Vorlagenoptionen aus:

Schemaversion 2

- Unterstützt die Client-Kompatibilität von Windows XP/Windows Server 2003 und höher.
- Unterstützt nur ältere Anbieter von Kryptografiediensten.

Schemaversion 3

- Unterstützt die Client-Kompatibilität von Windows Vista/Windows Server 2008 und höher.
- Unterstützt die Möglichkeit, dass der Anforderer die Verlängerung mit dem vorhandenen Schlüssel durchführen kann.
- Unterstützt nur wichtige Speicheranbieter.

Schemaversion 4


- Unterstützt die Client-Kompatibilität von Windows 8/Windows Server 2012 und höher.
- Unterstützt die Möglichkeit, dass der Anforderer die Verlängerung mit dem vorhandenen Schlüssel durchführen kann.
- Unterstützt ältere Anbieter von kryptografischen Diensten und Schlüsselspeicheranbietern.
- Client-Kompatibilität— Die Mindestbetriebssystemebene, die mit der Vorlage kompatibel ist. Wählen Sie eine der aufgelisteten Optionen:
 - Windows XP/Windows Server 2003
 - Windows Vista/Windows Server 2008
 - Windows 7/Windows Server 2008 R2
 - Windows 8 und höher/Windows Server 2012
 - Windows 8 und höher/Windows Server 2012 R2
 - Windows 8 und höher/Windows Server 2016 und höher

Konfigurieren Sie die Zertifikatseinstellungen

In den Einstellungen des Zertifikats definieren Sie im Abschnitt die folgenden Einstellungen für Zertifikate, die auf dieser Vorlage basieren.


- Art des Zertifikats— Geben Sie an, ob erstellt werden soll Benutzer- oder Computerzertifikate.
- 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 von Stunden, Tagen, Wochen, Monaten oder Jahren an. Der Mindestwert beträgt 2 Stunden.
- **Verlängerungszeitraum**— Geben Sie einen Verlängerungszeitraum für das Zertifikat als Ganzzahl in Stunden, Tagen, Wochen, Monaten oder Jahren an. Der Verlängerungszeitraum darf nicht mehr als 75% des Gültigkeitszeitraums betragen.
- **Name des Betreffs**— 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 Name des Betreffs**— 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 den alternativen Betreffnamen angegeben werden.

- Verzeichnis-GUID
- DNS-Name
- Domain-DNS
- E-Mail
- Dienstprinzipalname (SPN)
- **Benutzerprinzipalname (UPN)**

Konfigurieren Sie die Einstellungen für die Bearbeitung von Anfragen und die Registrierung

In den Optionen zur Bearbeitung von Zertifikatsanfragen und zur Registrierung geben Sie in diesem Abschnitt den Zweck von Zertifikaten auf der Grundlage der Vorlage an 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

Konfigurieren Sie wichtige Nutzungserweiterungen

In den Einstellungen für wichtige Nutzungserweiterungen wählen Sie im Abschnitt die Option für die Verwendung von Signaturen und Verschlüsselungsschlüsseln aus.

Verwendung des Signaturschlüssels

- Digitale Signatur
- Die Unterschrift ist ein Herkunftsnachweis (Nichtabstreitbarkeit)

Verwendung des Verschlüsselungsschlüssels

- Erlauben Sie den Schlüsselaustausch ohne Schlüsselverschlüsselung (Schlüsselvereinbarung)
- Erlauben Sie den Schlüsselaustausch nur mit Schlüsselverschlüsselung (Schlüsselverschlüsselung)
- Erlaubt die Verschlüsselung von Benutzerdaten (Datenverschlüsselung)

Sie können sich auch dafür entscheiden. Machen Sie wichtige Nutzungserweiterungen entscheidend für beide Schlüsseltypen.

Weisen Sie Anwendungsrichtlinien zu

In der Anwendungsrichtlinien Wählen Sie im Abschnitt alle zutreffenden Anwendungsrichtlinien aus. Die verfügbaren Richtlinien sind auf mehreren Seiten aufgelistet. Einige Richtlinien sind möglicherweise aufgrund früherer Einstellungen vorausgewählt.

Konfigurieren Sie benutzerdefinierte Anwendungsrichtlinien

In der Benutzerdefinierte Anwendungsrichtlinien In diesem Abschnitt können Sie der Vorlage benutzerdefinierte OIDs hinzufügen und angeben, ob Anwendungsrichtlinienerweiterungen wichtig sind.

Konfigurieren Sie die Kryptografieeinstellungen

In der Kryptografie-Einstellungen Wählen Sie im Abschnitt die folgenden Kategorien von Kryptografieeinstellungen für Zertifikate aus, die auf dieser Vorlage basieren.

1. Der Inhalt oben im Abschnitt wird bestimmt durch [Methode zur Erstellung von Vorlagen](#) und [Einstellungen für Vorlagen](#) die du zuvor gewählt hast.

- Wenn Sie die Standardeinstellung akzeptiert haben Vorlagenversion 2 in [Einstellungen für Vorlagen](#), dann werden hier die folgenden Statusmeldungen angezeigt:
 - Kategorie des Kryptografieanbieters
 - Legacy-Anbieter für kryptografische Dienste

In diesem Fall müssen keine Einstellungen konfiguriert werden und Sie können mit dem nächsten Schritt fortfahren.

- Wenn Sie angegeben haben Vorlagenversion 3 in [Einstellungen für Vorlagen](#), dann werden hier die folgenden Statusmeldungen angezeigt:
 - Kategorie des Kryptografieanbieters
 - Wichtiger Speicheranbieter

Sie müssen auch einen wählen Schlüsselalgorithmus aus den aufgelisteten Optionen ECDH_P256, ECDH_P384, ECDH_P521, und RSA.

- Wenn Sie angegeben haben Vorlagenversion 4 in [Einstellungen für Vorlagen](#), dann müssen Sie wählen zwischen einem Wichtiger Speicheranbieter und ein Anbieter älterer kryptografischer Dienste. Wenn du wählst Schlüssel Speicher zur Verfügung stellen, dann ein Schlüsselalgorithmus muss auch aus den aufgelisteten Optionen ausgewählt werden ECDH_P256, ECDH_P384, ECDH_P521, und RSA.
2. Minimale Schlüsselgröße (Bits)— Geben Sie die minimale Schlüsselgröße an. Diese Einstellung wirkt sich darauf aus, welche Kryptografieanbieter verfügbar sind.
 3. Wählen Sie aus, welche Kryptografieanbieter für Anfragen verwendet werden können— Wählen Sie eine der beiden verfügbaren Optionen:
 - Für Anfragen kann jeder Anbieter verwendet werden, der auf dem Computer der betroffenen Person verfügbar ist
 - Anfragen müssen über einen der folgenden ausgewählten Anbieter gestellt werden

Wenn Sie diese Option wählen, wird ein geöffnet Kryptografie-Anbieter Liste. Sie können Anbieter auswählen und priorisieren, indem Sie die Schaltflächen in der Bestellung Spalte. Die folgenden Anbieter werden unterstützt:

- Microsoft Base Cryptographic Provider v1.0
- Microsoft Base DSS und Diffie-Hellman Cryptographic Provider
- Microsoft Base Smartcard-Kryptoanbieter
- Microsoft DH S-Channel-Kryptografieanbieter
- Microsoft Enhanced Cryptographic Provider v1.0
- Microsoft hat DSS und Diffie-Hellman Cryptographic Provider erweitert
- Verbesserter RSA- und AES-Kryptografieanbieter von Microsoft
- Microsoft RSA SChannel-Kryptografieanbieter

Gruppen und Berechtigungen konfigurieren

In der Gruppen und Berechtigungen In diesem Abschnitt können Sie sich die Vorlagen, vorhandenen Gruppen und Berechtigungen für die Registrierung ansehen, oder Sie können die Fügen Sie neue Gruppen und Berechtigungen hinzu Schaltfläche, um neue hinzuzufügen. Die Schaltfläche öffnet ein Formular, das die folgenden Informationen benötigt:

- Anzeigename
- Sicherheits-ID(SID)
- Melden Sie sich an, mit den Optionen ZULASSEN | VERWEIGERN | NICHT GESETZT
- Automatische Registrierung, mit den Optionen ZULASSEN | VERWEIGERN | NICHT GESETZT

Konfigurieren Sie ablösende Vorlagen

In der Vorlagen ersetzen In diesem Abschnitt 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 wählen Fügen Sie die zu ersetzende Vorlage aus dem Active Directory hinzu und geben Sie den allgemeinen Namen der ablösenden Vorlage an.

Tagging konfigurieren

In der Schlagworte — optional In diesem Bereich 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 Schlagworte verwalten um das zu öffnen Schlagworte verwalten Seite.
- Wählen Sie Neues Tag hinzufügen, um ein Tag zu erstellen. Füllen Sie das aus Schlüsselfeld und optional das Wertfeld. Wähle Änderungen speichern um das Tag anzuwenden.
- Wählen Sie das Entfernen klicken Sie neben einem Tag, um es zum Löschen zu markieren, und wählen Sie Änderungen speichern zur Bestätigung.

Überprüfen und erstellen

Nachdem Sie die erforderlichen Informationen angegeben und Ihre Auswahl überprüft haben, wählen Sie Vorlage erstellen. Das öffnet Einzelheiten zur Vorlage, wo Sie die Einstellungen der neuen Vorlage überprüfen, die Vorlage bearbeiten oder löschen, Gruppen und Berechtigungen verwalten, abgelöste Vorlagen verwalten, Tags verwalten und die automatische Neuregistrierung für Zertifikatsinhaber einrichten können.

Eine Connector-Vorlage (CLI) erstellen

Verwenden Sie den [Vorlage erstellen](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Erstellen einer Connector-Vorlage (API)

Verwenden Sie die [CreateTemplate](#)Aktion in derAWS Private CAKonnektor für die Active Directory-API.

Konnektoren für Active Directory auflisten

Sie können die verwendenAWS Private CAConnector für die Active Directory-Konsole oderAWS CLUm die Connectors aufzulisten, die Sie besitzen.

Um Ihre Konnektoren mithilfe der Konsole aufzulisten

1. Melden Sie sich bei Ihrem anAWSKonto und öffnen Sie dasAWS Private CAConnector für die Active Directory-Konsole unter<https://console.aws.amazon.com/pca-connector-ad/home> .
2. Überprüfen Sie die Informationen in derKonnektoren für Active DirectoryListe. Mithilfe der Seitenzahlen oben rechts können Sie durch mehrere Seiten mit Anschlüssen navigieren. Jeder Konnektor belegt standardmäßig eine Zeile, in der die folgenden Informationsspalten angezeigt werden.
 - Stecker-ID— Die eindeutige ID des Connectors.
 - Name des Verzeichnisses— Die dem Connector zugeordnete Active Directory-Ressource.
 - Status des Konnektors— Status des Steckverbinders. Mögliche Werte sind:Erstellen|Aktiv|Löschen|Gescheitert.
 - Status des Dienstprinzipalnamens— Status des Dienstprinzipalnamens (SPN), der dem Connector zugeordnet ist. Mögliche Werte sind:Erstellen|Aktiv|Löschen|Gescheitert.
 - Status der Verzeichnisregistrierung— Registrierungsstatus des stellvertretenden Direktors. Mögliche Werte sind:Erstellen|Aktiv|Löschen|Gescheitert.
 - 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 Anschlüsse mithilfe derGröße der SeitePräferenz.

Um Ihre Steckverbinder mit dem aufzulistenAWS CLI

Verwenden Sie den[List-Konnektoren](#)Befehl zum Auflisten Ihrer Konnektoren.

Um Ihre Konnektoren mithilfe der API aufzulisten

Verwenden Sie die [ListConnectors](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Connector-Vorlagen auflisten

Sie können das verwenden AWS Private CA Connector für die Active Directory-Konsole oder AWS CLI um Vorlagen für Connectoren aufzulisten, die Sie besitzen. Connector-Vorlagen basieren auf AWS Private CA [BlankEndEntityCertificate_APIPassthrough/v1](#) Vorlagen.

Um Ihre Vorlagen mithilfe der Konsole aufzulisten

1. Melden Sie sich bei Ihrem AWS Konto und öffnen Sie das AWS Private CA Connector für die Active Directory-Konsole unter <https://console.aws.amazon.com/pca-connector-ad/home>.
2. Wählen Sie einen Anschluss aus der Konnektoren für Active Directory listen Sie auf und wählen Sie dann Details ansehen.
3. Überprüfen Sie auf der Seite mit den Konnektordetails die Informationen in der Vorlagen Abschnitt. 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.
 - Name der Vorlage— Der für Menschen lesbare Name der Vorlage.
 - Status der Vorlage— Status der Vorlage. Mögliche Werte sind: Aktiv|Löschen.
 - Vorlagen-ID— Die eindeutige Kennung der Vorlage.

Um Ihre Vorlagen aufzulisten, verwenden Sie den AWS CLI

Verwenden Sie die [Listenvorlagen](#) Befehl zum Auflisten von Vorlagen für den angegebenen Konnektor.

Um Ihre Vorlagen mithilfe der API aufzulisten

Verwenden Sie die [ListTemplates](#) Aktion in der AWS Private CA Connector für Active Directory-API zum Auflisten von Vorlagen für den angegebenen Connector.

Konnektordetails anzeigen

Gehen Sie wie folgt vor, um die Konfigurationsdetails eines Connectors in der Konsole, Befehlszeile oder API für anzuzeigenAWS Private CAKonnektor für Active Directory.

Connector anzeigen (Konsole)

Um Details für einen Connector (Konsole) anzuzeigen

1. Melden Sie sich bei Ihrem anAWSKonto und öffnen Sie dasAWS Private CAConnector für die Active Directory-Konsole unter<https://console.aws.amazon.com/pca-connector-ad/home> .
2. Wählen Sie einen Anschluss aus derKonnektoren für Active Directorylisten Sie auf und wählen Sie dannDetails ansehen.
3. Überprüfen Sie auf der Seite mit den Connector-Details die Informationen im Bereich Connector-Details, der Folgendes umfasst:
 - Stecker-ID
 - Status des Steckverbinders
 - Zusätzliche Statusdetails
 - Anschluss ARN
 - Serverendpunkt für die Zertifikatsregistrierungsrichtlinie
 - Name des Verzeichnisses
 - Verzeichnis-ID
 - AWS Private CAGegenstand
 - AWS Private CAStatus
 - VPC-Endpunkt und Sicherheitsgruppen
4. In derVorlagenIn diesem Bereich können Sie Vorlagen erstellen oder verwalten, die dem Connector zugeordnet sind.
5. Aus demDienstprinzipalname (SPN)In diesem Bereich können Sie den Namen des Dienstprinzips anzeigen, der dem Konnektor zugeordnet ist.
6. Aus demRegistrierung des VerzeichnissesIn diesem Bereich können Sie die mit dem Connector verknüpfte Verzeichnisregistrierung anzeigen oder ändern.
7. Aus demSchlagworte —fakultativIn diesem Bereich können Sie dem Connector zugeordnete Tags erstellen oder verwalten.

Konnektor anzeigen (CLI)

Verwenden Sie die [get-Connector](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Konnektor (API) anzeigen

Verwenden Sie den [GetConnector](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

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 anzuzeigen AWS Private CA Connector für Active Directory

Vorlage anzeigen (Konsole)

Um Details für eine Connector-Vorlage (Konsole) anzuzeigen

1. Melden Sie sich bei Ihrem an AWS Konto und öffnen Sie das AWS Private CA Connector für die Active Directory-Konsole unter <https://console.aws.amazon.com/pca-connector-ad/home> .
2. Wählen Sie einen Anschluss aus der Konnektoren für Active Directory listen Sie auf und wählen Sie dann Details ansehen.
3. Überprüfen Sie auf der Seite mit den Konnektordetails die Informationen in der Vorlagen Abschnitt und wählen Sie die Vorlage aus, die Sie überprüfen möchten. Wählen Sie dann Einzelheiten ansehen.
4. Auf der Detailseite finden Sie Details zur Vorlage. Im Bereich werden 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 Namen des Antragstellers
- 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 auch die folgenden Aktionen ausführen, indem Sie [den Bearbeiten](#), [Löschen](#), und [Aktionen](#) Knöpfe.

- [Edit \(Bearbeiten\)](#)
- [Löschen](#)
- [Gruppen und Berechtigungen verwalten](#)— Weitere Informationen finden Sie unter [Gruppen und Berechtigungen konfigurieren](#).
- [Verwalte abgelöste Vorlagen](#)— Weitere Informationen finden Sie unter [Überprüfen und erstellen](#).
- [Schlagworte verwalten](#)— Weitere Informationen finden Sie unter [Tagging-Connector für AD-Ressourcen](#).
- [Melden Sie alle Zertifikatsinhaber erneut an](#)— 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— Anwendungsrichtlinien für Vorlagen anzeigen und verwalten. Weitere Informationen finden Sie unter [Weisen Sie Anwendungsrichtlinien zu](#).
- Abgelöste Vorlagen— Ersetzte Vorlagen anzeigen und verwalten. Weitere Informationen finden Sie unter [Überprüfen und erstellen](#).
- Etikettfakultativ— Tagging auf dieser Vorlage anzeigen und verwalten. Weitere Informationen finden Sie unter [Tagging-Connector für AD-Ressourcen](#).

Um Details für eine Connector-Vorlage anzuzeigen (AWS CLI)

Vorlage anzeigen (CLI)

Verwenden Sie die [get-Template](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Vorlage anzeigen (API)

Um Details für eine Connector-Vorlage (API) anzuzeigen

Verwenden Sie den [GetTemplate](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Verwaltung von Verzeichnisregistrierungen

So verwalten Sie Verzeichnisregistrierungen (Konsole)

Verzeichnisregistrierungen für Konnektoren können von der obersten Ebene des verwaltet werden AWS Private CA Connector für die Active Directory-Konsole. In diesem Thema werden die verfügbaren Verwaltungsoptionen beschrieben.

1. Melden Sie sich bei Ihrem an AWS Konto und öffnen Sie das AWS Private CA Connector für die Active Directory-Konsole unter <https://console.aws.amazon.com/pca-connector-ad/home> .
2. Wählen Sie im linken Navigationsbereich Verzeichnisregistrierungen.
3. Das Verzeichnisregistrierungen Auf der Seite wird eine Tabelle registrierter Verzeichnisse mit den folgenden Feldern angezeigt:
 - Verzeichnis-ID— Die eindeutige ID des Verzeichnisses
 - Name des Verzeichnisses— Der Name der Verzeichnis-Domain
 - Typ des Verzeichnisses

- **Registriert**— Der Status der Registrierung. Unterstützte Werte sind CREATING | ACTIVE | DELETING | FAILED.
- **Status des Verzeichnisses**— Der Status des Verzeichnisses

Der Benutzer kann verwenden **Verzeichnis registrieren** um eine neue Registrierung zu erstellen.

4. Sie können eine der aufgelisteten Registrierungen auswählen, um sie zu verwalten. Dies ermöglicht die **Registrierungsdetails anzeigen** und **Verzeichnis abmelden** Knöpfe. Die **Registrierungsdetails anzeigen** Die Schaltfläche öffnet die Detailseite für die Registrierung.
5. Die **Angaben zur Registrierung des Verzeichnisses** Im Bereich werden die folgenden Informationen angezeigt:
 - **Name der Verzeichnisdomäne**
 - **Verzeichnis-ID**— Die eindeutige ID des Verzeichnisses. Wenn Sie den Link wählen, gelangen Sie zum **AWS Directory Service** Konsole.
 - **Typ des Verzeichnisses**
 - **Status**— Status des Verzeichnisses
 - **Verzeichnisregistrierung ARN**— Der Amazon-Ressourcenname der Verzeichnisregistrierung
 - **Zusätzliche Statusinformationen**
6. In der **Konnektoren und Service Principal Name (SPNs)** In diesem Bereich können Sie SPNs für den Connector verwalten. Weitere Informationen finden Sie unter [Konnektordetails anzeigen](#).
7. In der **Schlagworte** — optional In diesem Bereich 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 **Schlagworte verwalten** um das zu öffnen **Schlagworte verwalten** Seite.
 - Wählen Sie **Neues Tag hinzufügen**, um ein Tag zu erstellen. Füllen Sie das **aus** **Schlüssel** Feld und, optional, das **Wert** Feld. Wähle **Speichern** Sie die Änderungen um das Tag anzuwenden.
 - Wählen Sie **Entfernen** klicken Sie neben einem Tag, um es zum Löschen zu markieren, und wählen Sie **Speichern** Sie die Änderungen zur Bestätigung.

Um Verzeichnisregistrierungen (CLI) zu verwalten

Erstellen: Benutze die [create-directory-registration](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Abrufen: [get-directory-registration](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Liste: [list-directory-registrations](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Löschen: [delete-directory-registration](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Um Verzeichnisregistrierungen (API) zu verwalten

Erstellen: [CreateDirectoryRegistration](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Abrufen: [GetDirectoryRegistration](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Liste: [ListDirectoryRegistrations](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Löschen: [DeleteDirectoryRegistration](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

AD-Gruppen und Berechtigungen für Vorlagen verwalten

So verwalten Sie Vorlagengruppen und Berechtigungen (Konsole)

Gruppen und Berechtigungen für eine bestehende Vorlage können von der Detailseite der Vorlage aus 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.

Suchen Sie in Active Directory nach der Gruppensicherheits-ID

Sie können das folgende Skript verwenden, um die Gruppensicherheits-ID in Active Directory nachzuschlagen.

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

So verwalten Sie Vorlagengruppen und Berechtigungen (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

Um Vorlagengruppen und Berechtigungen (API) zu verwalten

Erstellen: [CreateTemplateGroupAccessControlEntry](#)Aktion in der AWS Private CA Connector für die Active Directory-API.

Update: [UpdateTemplateGroupAccessControlEntry](#)Aktion im AWS Private CA Connector für die Active Directory-API.

Abrufen: [GetTemplateGroupAccessControlEntry](#)Aktion in der AWS Private CA Connector für die Active Directory-API.

Liste: [ListTemplateGroupAccessControlEntries](#)Aktion in der AWS Private CA Connector für die Active Directory-API.

Löschen: [DeleteTemplateGroupAccessControlEntry](#)Aktion in der AWS Private CA Connector für die Active Directory-API.

Konfiguration des Dienstprinzipalnamens

So verwalten Sie Dienstprinzipalnamen (Konsole)

Der Service Principal Name (SPN) eines vorhandenen AD-Connectors kann auf der Detailseite des Connectors verwaltet werden. Weitere Informationen finden Sie unter Verwaltung der Verzeichnisregistrierung [Konnektordetails anzeigen](#)

Um Service Principal Names (CLI) zu verwalten

Erstellen: [create-service-principal-name](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Abfragen: [get-service-principal-name](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Liste: [list-service-principal-names](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Löschen: [delete-service-principal-name](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Um Dienstprinzipalnamen (API) zu verwalten

Erstellen: [CreateServicePrincipalName](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Abfragen: [GetServicePrincipalName](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Liste: [ListServicePrincipalNames](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Löschen: [DeleteServicePrincipalName](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Tagging-Connector für AD-Ressourcen

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.

So verwalten Sie das Ressourcen-Tagging (Konsole)

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](#)

Um das Resource Tagging (CLI) zu verwalten

Tag: [Tag-Resource](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Stichwörter auflisten: [list-tags-for-resource](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Markierung aufheben: [Untag-Resource](#) Befehl in der AWS Private CA Abschnitt Connector für Active Directory der AWS CLI.

Um das Ressourcen-Tagging (API) zu verwalten

Tag: [TagResource](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Stichwörter auflisten: [ListTagsForResource](#) Aktion in der AWS Private CA Konnektor für die Active Directory-API.

Markierung aufheben: [UntagResource](#) Aktion in der AWS Private CA Konnektor 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.

Fehlerbehebung

Wenn Sie Probleme mit der Verwendung von AWS Private CA haben, lesen Sie in den folgenden Themen nach.

Themen

- [Ein privates CA-Zertifikat erstellen und signieren](#)
- [Latenz bei OCSP-Antworten](#)
- [Konfiguration von Amazon S3 für die Erstellung eines CRL-Buckets](#)
- [Widerrufen eines selbstsignierten CA-Zertifikats](#)
- [Umgang mit Ausnahmen](#)
- [Verwenden Sie den Matter-Standard](#)
- [Konnektor für AD-Fehler und -Ausfälle](#)
- [Fehler bei der Erstellung des Connectors für AD-Connector](#)

Ein privates CA-Zertifikat erstellen und signieren

Nachdem Sie Ihre private CA erstellt haben, müssen Sie die CSR abrufen und an eine Zwischen- oder Stamm-CA in Ihrer X.509-Infrastruktur übergeben. Ihr CA verwendet die CSR zum Erstellen Ihres privaten CA-Zertifikats und signiert das Zertifikat anschließend, bevor Sie es zurückerhalten.

Leider ist es nicht möglich, spezielle Ratschläge für Probleme zu erteilen, die beim Erstellen und Signieren Ihrer privaten CA-Zertifikate auftreten. Die Einzelheiten Ihrer X.509-Infrastruktur und der darin enthaltenen CA-Hierarchie würden den Rahmen dieser AWS Private CA Dokumentation sprengen.

Latenz bei OCSP-Antworten

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.

Konfiguration von Amazon S3 für die Erstellung eines CRL-Buckets

Ihre private CA kann möglicherweise keinen CRL-Bucket erstellen, wenn Amazon S3 Block Public Access (Bucket-Einstellungen) für Ihr Konto durchgesetzt wird. Überprüfen Sie in diesem Fall Ihre Amazon S3 S3-Einstellungen. Weitere Informationen finden Sie unter [Verwenden von Amazon S3 Block Public Access](#).

Widerrufen eines selbstsignierten CA-Zertifikats

Sie können ein selbstsigniertes CA-Zertifikat nicht widerrufen. Stattdessen müssen Sie die Zertifizierungsstelle (Certificate Authority, CA) löschen.

Umgang mit Ausnahmen

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	Beschreibung	Abhilfe
<code>AccessDeniedException</code>	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 unter Zuweisen von Zertifikatserneuerungsberechtigungen zu ACM .
<code>InvalidArgsException</code>	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 Schlüssel

Ausnahme Zurückgegeben von AWS Private CA	Beschreibung	Abhilfe
		typ 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 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 .	Wenden Sie sich an das AWS Support Center , um eine 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.

Ausnahme Zurückgegeben von AWS Private CA	Beschreibung	Abhilfe
<code>OtherException</code>	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 .
<code>RequestFailedException</code>	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.
<code>ResourceNotFoundException</code>	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	Beschreibung	Abhilfe
ThrottlingException	Eine angeforderte API-Aktion ist fehlgeschlagen, da sie ein Kontingent überschritten hat.	<p>Vergewissern Sie sich, dass Sie nicht mehr Anrufe tätigen als zulässig. AWS Private CA</p> <p>Ein <code>ThrottlingException</code> 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.</p> <p>Wenn Ihr Kontingent ausgeschöpft ist, können Sie möglicherweise eine Erhöhung beantragen. Weitere Informationen finden Sie im Allgemeinen AWS-Referenz Handbuch unter Service Quotas.</p>

Ausnahme Zurückgegeben von AWS Private CA	Beschreibung	Abhilfe
<code>ValidationException</code>	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 Ihrer privaten Zertifizierungsstelle .

Verwenden Sie den Matter-Standard

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 [Verwenden der AWS Private CA API zur Implementierung des Boler-Standards \(Java-Beispiele\)](#)

[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	<code>BasicConstraints</code> , <code>KeyUsage</code> , und <code>ExtensionKeyUsage</code> Erweiterungen müssen als kritisch markiert werden.	Stellen Sie sicher, dass Sie die richtige Vorlage für Ihren Anwen ausgewählt haben.
0x0000000	Die Erweiterung zur Identifizierung des Autorität	AWS Private CA legt die Erweiterung zur Identifizierung des Ausschlüssels für Stammzertifikate nicht fest. Sie müssen mithilfe einen Base64-codierten <code>AuthorityKeyIdentifier</code> Wert generieren

Fehlercode	Bedeutung	Abhilfe
	sschlüssels muss vorhanden sein.	ihn dann durch eine übergeben. CustomExtension Weitere Informationen finden Sie unter Aktivieren Sie eine Root CA für Node Operational Certificates (NOC) . und Aktivieren einer Product Attestation A (PAA) .
0x000000E	Das Zertifikat ist abgelaufen.	Stellen Sie sicher, dass das von Ihnen verwendete Zertifikat noch abgelaufen ist.

Fehlercode	Bedeutung	Abhilfe
0x0000004	Fehler bei der Validierung der Zertifikatskette.	<p>Dieser Fehler kann auftreten, wenn Sie versuchen, ein Matter-Knoten-zertifikat zu erstellen, ohne die bereitgestellten Java-APIs zu verwenden, die die AWS Private CA API verwenden, um ein ordnungsgemäß konfiguriertes Zertifikat zu übergeben. <code>KeyUsage</code> erweitert standardmäßig die <code>KeyUsage</code> Neun-Bit-Erweiterungswerte, wobei das neunte Bit zu einem zusätzlichen Byte führt. Matter ignoriert das zusätzliche Byte bei Formatkonvertierungen, was zu Fehlern bei der Kettenvalidierung führt. Ein Wert <code>CustomKeyUsageExtension</code> in der <code>APIPassthrough</code> Vorlage kann jedoch verwendet werden, um die genaue Anzahl der Byte im <code>KeyUsage</code> Wert festzulegen. Ein Beispiel finden Sie unter Erstellen eines Node Operational Certificate (NOC).</p> <p>Wenn Sie den Beispielcode ändern oder ein alternatives X.509-Parserprogramm wie OpenSSL verwenden, müssen Sie eine manuelle Überprüfung durchführen, um Fehler bei der Kettenvalidierung zu vermeiden.</p> <p>Um zu überprüfen, ob Konvertierungen verlustfrei sind:</p> <ol style="list-style-type: none"> 1. Verwenden Sie <code>openssl</code>, um zu überprüfen, ob ein Zertifikat ein Knotenzertifikat (Endentitätszertifikat), eine gültige Kette enthält. In diesem Beispiel <code>rcac.pem</code> ist es das Stammzertifizierungszertifikat, <code>icac.pem</code> das Zwischenzertifikat der Zertifizierung und <code>noc.pem</code> das Knotenzertifikat. <pre data-bbox="797 1507 1624 1625">openssl verify -verbose -CAfile <(cat rcac.pem icac.pem) noc.pem</pre> 2. Verwenden Sie chip-cert, um das Knotenzertifikat im PEM-Format in das TLV-Format (Tag, Länge, Wert) und wieder zurück zu konvertieren.

Fehlercode	Bedeutung	Abhilfe
		<pre>./chip-cert convert-cert noc.pem noc.chip -c ./chip-cert convert-cert noc.chip noc_converted.pem</pre> <p>Die Dateien <code>noc.pem</code> und <code>noc_converted.pem</code> sollten die gleichen sein, die durch ein Tool zum Vergleich von Zeilen bestätigt wurden.</p>

Konnektor für AD-Fehler und -Ausfälle

Verwenden Sie die hier aufgeführten Informationen, um Fehler und Erstellungsfehler zu diagnostizieren und zu beheben, wenn Sie mit Connector for AD arbeiten.

Themen

- [Fehler](#)
- [Fehler bei der Erstellung des Connectors](#)
- [Fehler bei der SPN-Erstellung](#)

Fehler

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	Bedeutung	Abhilfe
0x8FFFA000	Die Kerberos-Authentifizierung ist fehlgeschlagen.	Wenn Sie die automatische Registrierung verwenden, korrigieren Sie Ihren AWS Resource Service Principal. Wenn Sie die Active Directory-

Fehlercode	Bedeutung	Abhilfe
		Benutzeroberfläche verwenden, um ein Zertifikat zu erhalten, führen Sie es aus. <code>gpupdate /force</code>
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 Zertifizierungsstelle in den Status Aktiv. Wenn sich Ihre private Zertifizierungsstelle im Status „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.

Fehlercode	Bedeutung	Abhilfe
0x8FFFA005	In der Vorlage wurde das <code>directoryGuid</code> 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 <code>directoryGuid</code> für Ihr Verzeichnis generiert. Problembehandlung in Active Directory.
0x8FFFA006	In der Vorlage wurde das <code>dnsHostName</code> 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 <code>dnsHostName</code> 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	Bedeutung	Abhilfe
0x8FFFA008	Die SOAP-Nachricht muss den Aktionsheader entweder oder haben. 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 Werttyp 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	Bedeutung	Abhilfe
0x8FFFA00C	Das enthaltene ungültige CMS. BinarySecurityToken	Das Base64-Format ist gültig, aber die kryptografische Nachricht ensyntax (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 spezifisc hen Vorlage ausstellen.	Überprüfen Sie die Validieru ngsausnahme von AWS Private CA. Sie können die Validierungsausnah me in Amazon EventBridge oder einsehen AWS CloudTrail.
0x8FFFA00F	Die SOAP-Nachricht muss den Anforderungstyp haben. http://do cs.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 CertificateEnrollm entPolicyServerEndpoint URI-Feld in der XCEP-Antwort enthält.	Setzen Sie den Header des Sicherhei ts-Tokens für die Anforderung entweder auf das Certifica teEnrollmentPolicy ServerEndpoint Feld oder auf das URI-Feld in der XCEP-Antwort.
0x8FFFA011	Die SOAP-Nachricht darf nur einen Aktionsheader haben.	Überprüfen Sie den SOAP-Nach richtenheader des Sicherheits- Tokens für die Anforderung und legen Sie den Header korrekt fest.

Fehlercode	Bedeutung	Abhilfe
0x8FFFA012	Die SOAP-Nachricht darf nur einen Header haben. <code>messageId</code>	Überprüfen Sie den SOAP-Nachrichtenheader des Sicherheitstokens 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 Sicherheitstokens 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 <code>CertificateTemplateInformation</code> oder die <code>CertificateTemplateName</code> Erweiterung muss in der vorhanden sein. <code>BinarySecurityToken</code>	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 <code>createTemplate</code> .
0x8FFFA017	Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.	Verlangsamen Sie die Rate der Anfragen.

Fehlercode	Bedeutung	Abhilfe
0x8FFFA018	Die SOAP-Nachricht muss einen Header enthalten. to	Überprüfen Sie den Header der SOAP-Nachricht.
0x8FFFA019	Die SOAP-Nachricht konnte aufgrund eines unbekanntes 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 alternativen Namen des Antragstellers aufgenommen werden soll, aber das Attribut wurde nicht im AD-Objekt für den Antragsteller gefunden.	Fügen Sie dem Active Directory-Objekt einen UPN hinzu.

Fehler bei der Erstellung des Connectors

Die Erstellung eines Connectors kann aus verschiedenen Gründen fehlschlagen. Wenn die Konnektorerstellung 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 Konnektordetails im Feld Zusätzliche Statusdetails im Container mit den Konnektordetails angezeigt. In der folgenden Tabelle werden die Gründe für Fehler und empfohlene Lösungsschritte beschrieben.

Status des Fehlers	Beschreibung	Abhilfe
CA_CERTIFICATE_REGISTRATION_FAILED	Connector for AD kann keine CA-Zertifikate in Ihr Verzeichnis 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	Beschreibung	Abhilfe
		Dienstkonto delegiert haben, löschen Sie den ausgefallenen Connector und erstellen Sie einen neuen. Informationen zum Delegieren von Berechtigungen finden Sie unter Delegieren von Rechten an Ihr Dienstkonto im AWS Directory Service Administratorhandbuch.
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 IAM-Richtlinie 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.
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.

Status des Fehlers	Beschreibung	Abhilfe
PRIVATECA_ACCESS_DENIED	Connector für AD kann nicht auf Ihre private CA zugreifen.	<p>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 IAM-Richtlinie.</p> <p>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.</p> <p>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.</p>

Status des Fehlers	Beschreibung	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 Sicherheitsgruppen . 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 IAM-Richtlinie

Status des Fehlers	Beschreibung	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.

Fehler bei der SPN-Erstellung

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	Beschreibung	Abhilfe
DIRECTORY_ACCESS_DENIED	Connector für AD kann nicht auf Ihr Verzeichnis zugreifen.	

Status des Fehlers	Beschreibung	Abhilfe
		Gewähren Sie Connector for AD Zugriff auf Ihr Verzeichnis. Ein Beispiel für eine IAM-Richtlinie, die Berechtigungen beinhaltet, die Verzeichniszugriff gewähren, finden Sie unter IAM-Richtlinie .
DIRECTORY_NOT_REACHABLE	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_DIFFERENT_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	Beschreibung	Abhilfe
SPN_LIMIT_EXCEEDED	Connector for AD kann den SPN nicht erstellen, da Sie das Limit von SPNs pro Verzeichnis erreicht haben. Die maximale Anzahl von SPNs pro Verzeichnis ist 10.	Löschen Sie einen oder mehrere SPNs aus Ihrem Konto und versuchen Sie erneut, den SPN zu erstellen.

Fehler bei der Erstellung des Connectors für 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.

Begriffe und Konzepte

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

Themen

- [Vertrauensstellung](#)
- [TLS-Serverzertifikate](#)
- [Zertifikatsignatur](#)
- [Zertifizierungsstelle](#)
- [Stammzertifizierungsstelle](#)
- [CA-Zertifikat](#)
- [CA-Stammzertifikat](#)
- [Endentitätszertifikat](#)
- [Selbstsignierte Zertifikate](#)
- [Privates Zertifikat](#)
- [Zertifikatpfad](#)
- [Einschränkung 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.

Zertifikatsignatur

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 wie SHA256 für Zertifikatsinhalte 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 [Zertifikatsignatur](#).

Mit AWS Private CA können Sie eine private Zertifizierungsstelle erstellen und diese 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 Trust Stores so, dass sie nur vertrauenswürdige Zertifizierungsstellen enthalten. Administratoren aktualisieren oder erstellen die Vertrauensspeicher in den Betriebssystemen, Instances und Host-Computer-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, wird zwischen den Ressourcen ein „Handshake“ ausgeführt. 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.

Mit AWS Private CA können Sie eine private Stammzertifizierungsstelle oder eine private untergeordnete Zertifizierungsstelle erstellen, die jeweils durch ein CA-Zertifikat gesichert 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 in einem AWS Private CA-X.509-CA-Zertifikat. 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 (Certificate Authority, CA) befindet sich in der Regel innerhalb einer hierarchischen Struktur, die mehrere andere CAs mit klar definierten übergeordneten und untergeordneten Beziehungen zwischen ihnen enthält. Untergeordnete Zertifizierungsstellen werden von ihren übergeordneten Zertifizierungsstellen zertifiziert, wodurch eine Zertifikatkette 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.

Endentitätszertifikat

Ein Endentitätszertifikat identifiziert eine Ressource, z. B. einen Server, eine Instance, einen Container oder ein Gerät. Im Gegensatz zu CA-Zertifikaten können Endentitätszertifikate nicht zum Ausstellen von Zertifikaten verwendet werden. Andere gängige Begriffe für Endentitätszertifikate sind „Client“- oder „Blatt“-Zertifikate.

Selbstsignierte Zertifikate

Ein Zertifikat, das anstatt von einer höheren Zertifizierungsstelle vom Aussteller signiert ist. Im Gegensatz zu Zertifikaten, die aus einem sicheren Stamm ausgestellt wurden, der von einer Zertifizierungsstelle verwaltet wird, fungieren selbstsignierte Zertifikate als ihr eigenes Stammverzeichnis und haben daher erhebliche Einschränkungen: Sie können verwendet werden, um die Kabelverschlüsselung bereitzustellen, aber nicht, um die Identität zu überprüfen, und sie können nicht widerrufen 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 in Ihrer Organisation verwenden können, aber 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
```

Zertifikatpfad

Ein Client, der auf einem Zertifikat basiert, 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 Endentitätszertifikat nicht abgelaufen ist, Integrität hat (nicht manipuliert oder geändert wurde) und ob Einschränkungen im Zertifikat durchgesetzt werden.

Einschränkung der Pfadlänge

Die grundlegenden Einschränkungen `pathLenConstraint` für ein CA-Zertifikat legen die Anzahl der untergeordneten CA-Zertifikate fest, die in der Kette darunter vorhanden sein können. Beispielsweise darf ein CA-Zertifikat mit einer Pfadlängenbeschränkung von Null keine untergeordneten CAs

haben. Eine Zertifizierungsstelle mit der Pfadlängenbeschränkung „Eins“ kann bis zu einer Ebene untergeordneter Zertifizierungsstellen haben. [RFC 5280](#) definiert dies als „die maximale Anzahl von non-self-issued Zwischenzertifikaten, die diesem Zertifikat in einem gültigen Zertifizierungspfad folgen dürfen“. Der Wert für die Pfadlänge schließt das Endentitätszertifikat aus, obwohl die freie Sprache über die „Länge“ oder „Tiefe“ einer Validierungskette dies enthalten kann... was zu Verwirrung führt.

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
Neue Anleitung zur Fehlerbehebung bei Konnektoren	Es wurden zwei neue Abschnitte zur Behebung von Fehlern bei der Connector- und SPN-Erstellung hinzugefügt.	4. April 2024
CDP-Erweiterung für Matter wird 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	Einführung neuer Namen für AWS verwaltete IAM-Richt	13. Februar 2023

[geändert, dass sie dem neuen Dienstnamen entsprechen](#)

linien, die Standardberechtigungen für festlegen. AWS Private CA Weitere Informationen finden Sie unter [AWS Verwaltete Richtlinien](#).

[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 Zertifizierungsstellen, die kurzlebige Zertifikate ausstellen](#)

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 Zertifizierungsstellen und Zertifikate zu konfigurieren, und das [CustomExtensionObjekt](#), 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, Zertifikate über private Zertifizierungsstellen auszustellen, die sie nicht besitzen. Weitere Informationen finden Sie unter Kontoübergreifender Zugriff auf private Zertifizierungsstellen .	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

Ein spezieller Sicherheitsbereich wurde 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 AWS CloudFormation. Weitere Informationen finden Sie unter ACMPCA-Ressourcenreferenz im AWS 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 den neuen APIs <code>TagResource</code> , <code>UntagResource</code> und <code>ListTagsForResource</code> unterstützt. 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 Namensbeschränkungen 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 Stammzertifizierungsstellen ist keine externe übergeordnete Zertifizierungsstelle erforderlich.	20. Juni 2019

[Zusätzlicher Support für Regionen](#)

Regionsunterstützung für die Regionen AWS GovCloud (US-West und US-Ost) hinzugefügt. Eine vollständige 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-Pazifik (Mumbai und Seoul), USA West (Nordkalifornien) und EU (Paris und Stockholm) hinzugefügt. Eine vollständige 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 Erneuerungsgworkflow testen. Weitere Informationen finden Sie unter [Testen der Konfiguration von ACM verwalteter Erneuerung.](#)

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

[Gelöschte Zertifizierungsstellen wiederherstellen](#)

Die Wiederherstellung privater Zertifizierungsstellen ermöglicht es Kunden, private Zertifizierungsstellen (CAs) für bis zu 30 Tage nach ihrer Löschung wiederherzustellen. 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.

Änderung	Beschreibung	Datum
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.