Panduan Pengguna

AWS Private Certificate Authority

Versi latest

AWS Private Certificate Authority: Panduan Pengguna

Table of Contents

Apa itu AWS Private CA?

AWS Private CA memungkinkan pembuatan hierarki otoritas sertifikat swasta (CA), termasuk CA root dan bawahan, tanpa biaya investasi dan pemeliharaan pengoperasian CA lokal. CA privat Anda dapat mengeluarkan sertifikat X.509 entitas akhir yang berguna dalam skenario termasuk:

- Membuat saluran komunikasi TLS terenkripsi
- Mengautentikasi pengguna, komputer, titik akhir API, dan perangkat IoT
- Kode penandatanganan kriptografi
- Menerapkan Protokol Status Sertifikat Online (OCSP) untuk mendapatkan status pencabutan sertifikat

AWS Private CA operasi dapat diakses dari AWS Management Console, menggunakan AWS Private CA API, atau menggunakan AWS CLI.

Topik

Apa layanan sertifikat terbaik untuk kebutuhan saya?

Ada dua AWS layanan untuk menerbitkan dan menyebarkan sertifikat X.509. Pilih salah satu yang paling sesuai dengan kebutuhan Anda. Pertimbangan termasuk apakah Anda memerlukan sertifikat publik atau pribadi, sertifikat khusus, sertifikat yang ingin Anda terapkan ke AWS layanan lain, atau manajemen dan pembaruan sertifikat otomatis.

1. AWS Private CALayanan ini ditujukan untuk pelanggan perusahaan yang membangun infrastruktur kunci publik (PKI) di dalam AWS cloud dan ditujukan untuk penggunaan pribadi dalam suatu

organisasi. Dengan AWS Private CA, Anda dapat membuat hierarki CA Anda sendiri dan mengeluarkan sertifikat dengannya untuk mengautentikasi pengguna internal, komputer, aplikasi, layanan, server, dan perangkat lain, dan untuk menandatangani kode komputer. Sertifikat yang dikeluarkan oleh CA privat hanya dipercaya di dalam organisasi Anda, bukan di internet.

Setelah membuat CA privat, Anda memiliki kemampuan untuk menerbitkan sertifikat secara langsung (yaitu, tanpa memperoleh validasi dari CA pihak ketiga) dan menyesuaikannya untuk memenuhi kebutuhan internal organisasi Anda. Misalnya, Anda mungkin ingin:

- Membuat sertifikat dengan nama subjek apa saja.
- Membuat sertifikat dengan tanggal kedaluwarsa.
- Menggunakan algoritme kunci privat yang didukung dan panjang kunci.
- Menggunakan algoritme penandatanganan yang didukung.
- Kontrol penerbitan sertifikat menggunakan templat.

Anda berada di tempat yang tepat untuk layanan ini. Untuk memulai, masuk ke konsol https://console.aws.amazon.com/acm-pca/.

2. AWS Certificate Manager (ACM) — Layanan ini mengelola sertifikat untuk pelanggan perusahaan yang membutuhkan kehadiran web aman tepercaya publik menggunakan TLS. Anda dapat menerapkan sertifikat ACM ke AWS Elastic Load Balancing, CloudFront Amazon, Amazon API Gateway, dan layanan terintegrasi lainnya. Aplikasi paling umum dari jenis ini adalah situs web publik yang aman dengan persyaratan lalu lintas yang signifikan.

   Dengan layanan ini, Anda dapat menggunakan sertifikat publik yang disediakan oleh ACM (sertifikat ACM) atau sertifikat yang Anda impor ke ACM. Jika Anda menggunakan AWS Private CA untuk membuat CA, ACM dapat mengelola penerbitan sertifikat dari CA pribadi tersebut dan mengotomatiskan perpanjangan sertifikat.

   Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS Certificate Manager.

Wilayah

Seperti kebanyakan AWS sumber daya, otoritas sertifikat swasta (CA) adalah sumber daya Regional. Untuk menggunakan CA privat di lebih dari satu Wilayah, Anda harus membuat CA di Wilayah tersebut. Anda tidak dapat menyalin CA privat antar Wilayah. Kunjungi AWS Wilayah dan Titik Akhir di Referensi Umum AWSatau Tabel AWS Wilayah untuk melihat ketersediaan Regional. AWS Private CA

> **Note**
>
> ACM saat ini tersedia di beberapa wilayah yang AWS Private CA tidak.

Layanan terintegrasi dengan AWS Private Certificate Authority

Jika Anda menggunakan AWS Certificate Manager untuk meminta sertifikat pribadi, Anda dapat mengaitkan sertifikat itu dengan layanan apa pun yang terintegrasi dengan ACM. Ini berlaku baik untuk sertifikat yang dirantai ke AWS Private CA root dan sertifikat yang dirantai ke root eksternal. Untuk informasi selengkapnya, lihat Layanan Terpadu di Panduan AWS Certificate Manager Pengguna.

Anda juga dapat mengintegrasikan CA privat ke dalam Amazon Elastic Kubernetes Service untuk memberikan penerbitan sertifikat di dalam klaster Kubernetes. Untuk informasi selengkapnya, lihat Mengamankan Kubernetes dengan AWS Private CA.

> **Note**
>
> Amazon Elastic Kubernetes Service bukan layanan terintegrasi ACM.

Jika Anda menggunakan AWS Private CA API atau AWS CLI menerbitkan sertifikat atau mengekspor sertifikat pribadi dari ACM, Anda dapat menginstal sertifikat di mana pun Anda inginkan.

Algoritme kriptografi yang didukung

AWS Private CA mendukung algoritma kriptografi berikut untuk pembuatan kunci pribadi dan penandatanganan sertifikat.

Algoritme yang didukung

Algoritme kunci privat	Algoritme penandatanganan
RSA_2048	SHA256WITHECDSA
RSA_4096	SHA384WITHECDSA
EC_prime256v1	SHA512WITHECDSA

Algoritme kunci privat	Algoritme penandatanganan
EC_secp384r1	SHA256WITHRSA SHA384WITHRSA SHA512WITHRSA

Daftar ini hanya berlaku untuk sertifikat yang dikeluarkan langsung AWS Private CA melalui konsol, API, atau baris perintahnya. Saat AWS Certificate Manager mengeluarkan sertifikat menggunakan CA dari AWS Private CA, ini mendukung beberapa tetapi tidak semua algoritme ini. Untuk informasi selengkapnya, lihat Meminta Sertifikat Pribadi di Panduan AWS Certificate Manager Pengguna.

> **Note**
>
> Dalam semua kasus, keluarga algoritma penandatanganan yang ditentukan (RSA atau ECDSA) harus cocok dengan keluarga algoritma kunci pribadi CA.

Kuota

AWS Private CA memberikan kuota ke jumlah sertifikat dan otoritas sertifikat yang diizinkan. Tarif permintaan untuk tindakan API juga tunduk pada kuota. AWS Private CA kuota khusus untuk AWS akun dan Wilayah.

AWS Private CA membatasi permintaan API pada tingkat yang berbeda tergantung pada operasi API. Throttling berarti AWS Private CA menolak permintaan yang valid karena permintaan melebihi kuota operasi untuk jumlah permintaan per detik. Ketika permintaan dibatasi, AWS Private CA mengembalikan kesalahan. ThrottlingException AWS Private CA tidak menjamin tingkat permintaan minimum untuk API.

Untuk melihat kuota apa yang dapat disesuaikan, lihat tabel AWS Private CA kuota di. Referensi Umum AWS

Anda dapat melihat kuota saat ini dan meminta kenaikan kuota menggunakan Service AWS Quotas.

Untuk melihat up-to-date daftar AWS Private CA kuota Anda

1. Masuk ke AWS akun Anda.
2. Buka konsol Kuota Layanan di https://console.aws.amazon.com/servicequotas/.

3. Dalam daftar Layanan, pilih AWS Certificate Manager Private Certificate Authority (ACM PCA). Setiap kuota dalam daftar Service Quotas menunjukkan nilai kuota yang Anda gunakan saat ini, nilai kuota default, dan apakah kuota dapat disesuaikan atau tidak. Pilih nama kuota untuk informasi lebih lanjut.

Untuk meminta peningkatan kuota

1. Dalam daftar Service Quotas, pilih tombol radio untuk kuota yang dapat disesuaikan.
2. Pilih tombol Minta peningkatan kuota.
3. Lengkapi dan kirimkan formulir Permintaan peningkatan kuota.

AWS Private CA terintegrasi dengan AWS Certificate Manager. Anda dapat menggunakan konsol ACM, AWS CLI, atau ACM API untuk meminta sertifikat pribadi dari CA pribadi yang ada. Sertifikat PKI privat ini, yang dikelola oleh ACM, bergantung pada kuota PCA dan kuota yang ditempatkan ACM pada sertifikat publik dan impor. Untuk informasi selengkapnya tentang persyaratan ACM, lihat Meminta Sertifikat Pribadi dan Kuota di AWS Certificate Manager Panduan Pengguna.

Kepatuhan RFC

AWS Private CA tidak memberlakukan batasan tertentu yang ditentukan dalam RFC 5280. Situasi sebaliknya juga benar: kendala tambahan tertentu yang sesuai untuk CA privat diberlakukan.

Ditegakkan

- Tidak Setelah tanggal. Sesuai dengan RFC 5280, AWS Private CA mencegah penerbitan sertifikat yang memuat tanggal lebih lambat dari `Not After` tanggal penerbitan sertifikat CA. `Not After`
- Kendala dasar. AWS Private CA memberlakukan batasan dasar dan panjang jalur dalam sertifikat CA yang diimpor.

  Kendala dasar menunjukkan apakah sumber daya yang diidentifikasi oleh sertifikat adalah CA dan dapat mengeluarkan sertifikat. Sertifikat CA yang diimpor AWS Private CA harus menyertakan ekstensi kendala dasar, dan ekstensi harus ditandai. `critical` Selain `critical` bendera, `CA=true` harus diatur. AWS Private CA memberlakukan kendala dasar dengan gagal dengan pengecualian validasi karena alasan berikut:

  - Ekstensi tidak termasuk dalam sertifikat CA.
  - Ekstensi tidak ditandai `critical`.

Panjang jalur (jalur LenConstraint) menentukan berapa banyak CA bawahan yang mungkin ada di hilir dari sertifikat CA yang diimpor. AWS Private CA memberlakukan panjang jalur dengan gagal dengan pengecualian validasi karena alasan berikut:

  - Mengimpor sertifikat CA akan melanggar kendala panjang jalur dalam sertifikat CA atau sertifikat CA apa pun dalam rantai.
  - Menerbitkan sertifikat akan melanggar kendala panjang jalur.
- Batasan nama menunjukkan ruang nama di mana semua nama subjek dalam sertifikat berikutnya di jalur sertifikasi harus ditempatkan. Pembatasan berlaku untuk nama subjek yang dibedakan dan nama alternatif subjek.

Tidak ditegakkan

- Kebijakan sertifikat. Kebijakan sertifikat mengatur kondisi di mana CA mengeluarkan sertifikat.
- Menghambat AnyPolicy. Digunakan dalam sertifikat yang dikeluarkan untuk CA.
- Nama Alternatif Penerbit. Memungkinkan identitas tambahan dikaitkan dengan penerbit sertifikat CA.
- Kendala kebijakan. Kendala ini membatasi kapasitas CA untuk menerbitkan sertifikat CA bawahan.
- Pemetaan Kebijakan. Digunakan dalam sertifikat CA. Daftar satu atau lebih pasangan OID; setiap pasangan termasuk issuerDomainPolicy dan subjekDomainPolicy.
- Atribut Direktori Subjek. Digunakan untuk menyampaikan atribut identifikasi subjek.
- Akses Informasi Subjek. Cara mengakses informasi dan layanan untuk subjek sertifikat di mana ekstensi muncul.
- Subject Key Identifier (SKI) dan Authority Key Identifier (AKI). RFC memerlukan sertifikat CA yang berisi ekstensi SKI. Sertifikat yang dikeluarkan oleh CA harus berisi ekstensi AKI yang cocok dengan SKI sertifikat CA. AWS tidak menegakkan persyaratan ini. Jika Sertifikat CA Anda tidak berisi SKI, entitas akhir yang diterbitkan atau sertifikat CA bawahan AKI akan menjadi hash SHA-1 dari kunci publik penerbit.
- SubjectPublicKeyInfodan Nama Alternatif Subjek (SAN). Saat menerbitkan sertifikat, salin ekstensi AWS Private CA SubjectPublicKeyInfo dan SAN dari CSR yang disediakan tanpa melakukan validasi.

Harga

Akun Anda dikenai harga bulanan untuk setiap CA privat mulai dari saat Anda membuatnya. Anda juga dikenakan biaya untuk setiap sertifikat yang Anda keluarkan. Biaya ini mencakup sertifikat yang Anda ekspor dari ACM dan sertifikat yang Anda buat dari AWS Private CA API atau AWS Private CA CLI. Anda tidak dikenakan biaya untuk CA privat setelah dihapus. Namun, jika Anda memulihkan CA privat, Anda akan dikenakan biaya untuk waktu antara penghapusan dan pemulihan. Sertifikat privat yang kunci privatnya tidak dapat Anda akses gratis. Ini termasuk sertifikat yang digunakan dengan Layanan Terpadu seperti Elastic Load Balancing, CloudFront, dan API Gateway.

Untuk informasi AWS Private CA harga terbaru, lihat AWS Private Certificate Authority Harga. Anda juga dapat menggunakan kalkulator AWS harga untuk memperkirakan biaya.

Keamanan di AWS Private Certificate Authority

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan dari cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program AWS Kepatuhan Program AWS Kepatuhan . Untuk mempelajari tentang program kepatuhan yang berlaku AWS Private Certificate Authority, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan AWS Private CA. Topik berikut menunjukkan cara mengonfigurasi AWS Private CA untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan Layanan AWS yang lain yang membantu Anda memantau dan mengamankan AWS Private CA sumber daya Anda.

Topik

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

Akses ke AWS Private CA memerlukan kredensil yang AWS dapat digunakan untuk mengautentikasi permintaan Anda. Topik berikut memberikan detail tentang bagaimana Anda dapat menggunakan (IAM)AWS Identity and Access Management untuk membantu mengamankan otoritas sertifikasi (CA) privat Anda dengan mengontrol siapa yang dapat mengaksesnya.

Di AWS Private CA, sumber daya utama yang Anda gunakan adalah otoritas sertifikat (CA). Setiap CA privat yang Anda miliki atau kendalikan diidentifikasi oleh Amazon Resource Name (ARN), yang memiliki formulir berikut.

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

Pemilik sumber daya adalah entitas utama dari AWS akun tempat AWS sumber daya dibuat. Contoh berikut menggambarkan cara kerjanya.

- Jika Anda menggunakan kredensi Anda Pengguna root akun AWS untuk membuat CA pribadi, AWS akun Anda memiliki CA.

  > **Important**
  > - Kami tidak menyarankan menggunakan Pengguna root akun AWS untuk membuat CA.
  > - Kami sangat menyarankan penggunaan otentikasi multi-faktor (MFA) setiap kali Anda mengakses. AWS Private CA

- Jika Anda membuat pengguna IAM di AWS akun Anda, Anda dapat memberikan izin kepada pengguna tersebut untuk membuat CA pribadi. Namun, akun tempat pengguna tersebut memiliki CA.
- Jika Anda membuat peran IAM di AWS akun Anda dan memberinya izin untuk membuat CA pribadi, siapa pun yang dapat mengambil peran tersebut dapat membuat CA. Namun, akun yang memiliki peran tersebut akan memiliki CA privat.

Kebijakan izin menjelaskan siapa yang memiliki akses ke suatu objek. Diskusi berikut menjelaskan pilihan yang tersedia untuk membuat kebijakan izin.

> **Note**
>
> Dokumentasi ini membahas penggunaan IAM dalam konteks. AWS Private CA Bagian ini tidak memberikan informasi yang mendetail tentang layanan IAM. Untuk dokumentasi lengkap IAM, lihat Panduan Pengguna IAM. Untuk informasi tentang sintaks dan deskripsi kebijakan IAM, lihat Referensi Kebijakan AWS IAM.

AWS Private CA Operasi dan izin API

Ketika Anda menyiapkan dan kontrol akses dan kebijakan izin yang Anda rencanakan untuk lampirkan ke identitas IAM (kebijakan berbasis identitas), gunakan tabel berikut sebagai referensi. Kolom pertama dalam tabel mencantumkan setiap operasi AWS Private CA API. Anda menentukan tindakan di elemen `Action` kebijakan. Kolom yang tersisa memberikan informasi tambahan.

AWS Private CA Operasi API	Izin yang diperlukan	Sumber daya
CreateCertificateAuthority	`acm-pca:CreateCertificateAuthority` `acm-pca:TagCertificateAuthority` (Hanya diperlukan saat membuat CA dengan tag.)	`arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566`
CreateCertificateAuthorityAuditReport	`acm-pca:CreateCertificateAuthorityAuditReport`	`arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566`
CreatePermission	`acm-pca:CreatePermission`	`arn:aws:acm-pca:us-east-1:111122223333:certificate-authority/11223344-`

AWS Private CA Operasi API	Izin yang diperlukan	Sumber daya
		1234-1122-2233-112 233445566
DeleteCertificateAuthority	acm-pca:DeleteCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DeletePermission	acm-pca:DeletePermission	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DeletePolicy	acm-pca:DeletePolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DescribeCertificateAuthority	acm-pca:DescribeCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

AWS Private CA Operasi API	Izin yang diperlukan	Sumber daya
DescribeCertificateAuthorityAuditReport	acm-pca:DescribeCertificateAuthorityAuditReport	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
GetCertificate	acm-pca:GetCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
GetCertificateAuthorityCertificate	acm-pca:GetCertificateAuthorityCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
GetCertificateAuthorityCsr	acm-pca:GetCertificateAuthorityCsr	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
GetPolicy	acm-pca:GetPolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

AWS Private CA Operasi API	Izin yang diperlukan	Sumber daya
ImportCertificateAuthorityCertificate	acm-pca:ImportCertificateAuthorityCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
IssueCertificate	acm-pca:IssueCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListCertificateAuthorities	acm-pca:ListCertificateAuthorities	N/A
ListPermissions	acm-pca:ListPermissions	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListTags	acm-pca:ListTags	N/A
PutPolicy	acm-pca:PutPolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

AWS Private CA Operasi API	Izin yang diperlukan	Sumber daya
RevokeCertificate	acm-pca:RevokeCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
TagCertificateAuthority	acm-pca:TagCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
UntagCertificateAuthority	acm-pca:UntagCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
UpdateCertificateAuthority	acm-pca:UpdateCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

Untuk memberikan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti petunjuk dalam Buat set izin dalam Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti petunjuk dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.

- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti petunjuk dalam Membuat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
  - (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk di Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

AWS kebijakan terkelola

AWS Private CA mencakup serangkaian kebijakan AWS terkelola yang telah ditentukan sebelumnya untuk AWS Private CA administrator, pengguna, dan auditor. Memahami kebijakan ini dapat membantu Anda menerapkan Kebijakan yang dikelola pelanggan.

Pilih salah satu kebijakan yang tercantum di bawah ini untuk melihat detail dan contoh kode kebijakan.

AWSPrivateCAFullAccess

Memberikan kontrol administratif yang tidak terbatas.

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

AWSPrivateCAReadOnly

Memberikan akses terbatas pada operasi API hanya-baca.

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

AWSPrivateCAPrivilegedUser

Memberikan kemampuan untuk menerbitkan dan mencabut sertifikat CA. Kebijakan ini tidak memiliki kemampuan administratif lain dan tidak memiliki kemampuan untuk menerbitkan sertifikat entitas akhir. Izin bersifat saling eksklusif dengan kebijakan Pengguna.

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

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

AWSPrivateCAUser

Memberikan kemampuan untuk menerbitkan dan mencabut sertifikat entitas akhir. Kebijakan ini tidak memiliki kemampuan administratif dan tidak memiliki kemampuan untuk menerbitkan sertifikat CA. Izin saling eksklusif dengan kebijakan. PrivilegedUser

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

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

```
}
```

AWSPrivateCAAuditor

Berikan akses ke operasi API hanya-baca dan izin untuk membuat laporan audit CA.

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

Pembaruan kebijakan AWS terkelola untuk AWS Private CA

Dalam tabel berikut, lihat detail tentang pembaruan kebijakan AWS terkelola AWS Private CA sejak layanan mulai melacak perubahan ini. Untuk peringatan otomatis tentang semua perubahan AWS Private CA, berlangganan umpan RSS di halaman. Riwayat Dokumen

Perubahan kebijakan terkelola

Perubahan	Deskripsi	Tanggal
Nama kebijakan baru: • `AWSPrivateCAFullAccess` • `AWSPrivateCAReadOnly` • `AWSPrivateCAPrivilegedUser` • `AWSPrivateCAAuditor` • `AWSPrivateCAUser`	Awalan nama kebijakan diubah dari `AWSCertificateManagerPrivateCA` menjadi. `AWSPrivateCA` Fungsionalitas tetap tidak berubah.	13 Februari 2023

Kebijakan yang dikelola pelanggan

Sebagai praktik terbaik, jangan gunakan Anda Pengguna root akun AWS untuk berinteraksi AWS, termasuk AWS Private CA. Sebagai gantinya gunakan AWS Identity and Access Management (IAM) untuk membuat pengguna IAM, peran IAM, atau pengguna federasi. Buat grup administrator dan tambahkan Anda sendiri ke dalamnya. Kemudian, masuk sebagai administrator. Tambahkan pengguna tambahan ke grup sesuai kebutuhan.

Praktik terbaik lainnya adalah membuat kebijakan IAM yang dikelola pelanggan yang dapat Anda tetapkan kepada pengguna. Kebijakan terkelola pelanggan adalah kebijakan berbasis identitas yang dapat Anda buat dan yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam akun AWS Anda. Kebijakan semacam itu membatasi pengguna untuk hanya melakukan AWS Private CA tindakan yang Anda tentukan.

Contoh kebijakan yang dikelola pelanggan berikut memungkinkan pengguna membuat laporan audit CA. Ini hanya contoh. Anda dapat memilih AWS Private CA operasi apa pun yang Anda inginkan. Untuk contoh lainnya, lihat Kebijakan inline.

Untuk membuat kebijakan terkelola pelanggan

1. Masuk ke konsol IAM menggunakan kredensial administrator AWS .
2. Di panel navigasi kiri konsol tersebut, pilih Kebijakan.

3. Pilih Buat kebijakan.
4. Pilih tab JSON.
5. Salin kebijakan bucket berikut dan tempelkan ke dalam editor.

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

6. Pilih Tinjau kebijakan.
7. Untuk Nama, ketik `PcaListPolicy`.
8. (Opsional) Ketik deskripsi.
9. Pilih Buat kebijakan.

Administrator dapat melampirkan kebijakan ke setiap pengguna IAM untuk membatasi AWS Private CA tindakan apa yang dapat dilakukan pengguna. Untuk cara menerapkan kebijakan izin, lihat Mengubah Izin untuk Pengguna IAM di Panduan Pengguna IAM.

Kebijakan inline

Kebijakan inline adalah kebijakan yang Anda buat dan kelola dan sematkan secara langsung ke dalam satu pengguna, grup, atau peran. Contoh kebijakan berikut menunjukkan cara menetapkan izin untuk melakukan tindakan AWS Private CA . Untuk informasi umum tentang mengelola kebijakan inline, lihat Bekerja dengan Kebijakan Inline di Panduan Pengguna IAM. Anda dapat menggunakan AWS Management Console, the AWS Command Line Interface (AWS CLI), atau IAM API untuk membuat dan menyematkan kebijakan sebaris.

> **Important**
>
> Kami sangat menyarankan penggunaan otentikasi multi-faktor (MFA) setiap kali Anda mengakses. AWS Private CA

Topik

Mencantumkan CA privat

Kebijakan berikut memungkinkan pengguna untuk membuat daftar semua CA privat dalam akun.

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

Mengambil sertifikat CA privat

Kebijakan berikut memungkinkan pengguna untuk mengambil sertifikat CA privat tertentu.

```
{
   "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"
```

```
   }
}
```

Mengimpor sertifikat CA privat

Kebijakan berikut memungkinkan pengguna untuk mengimpor sertifikat CA privat.

```
{
   "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"
   }
}
```

Menghapus CA privat

Kebijakan berikut memungkinkan pengguna untuk menghapus CA privat tertentu.

```
{
   "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: Melampirkan tag ke CA pada saat pembuatan

Kebijakan berikut memungkinkan pengguna untuk menerapkan tag selama pembuatan CA.

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

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

Tag-on-create: Penandaan terbatas

tag-on-create Kebijakan berikut mencegah penggunaan pasangan nilai kunci Environment=Prod selama pembuatan CA. Menandai dengan pasangan nilai kunci lainnya diperbolehkan.

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

Mengontrol akses ke CA Pribadi menggunakan tag

Kebijakan berikut hanya mengizinkan akses ke CA dengan pasangan nilai kunci Environment=. PreProd Hal ini juga mengharuskan CA baru menyertakan tag ini.

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

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

Akses hanya-baca ke AWS Private CA

Kebijakan berikut memungkinkan pengguna untuk menjelaskan dan membuat daftar otoritas sertifikat privat dan untuk mengambil sertifikat CA privat dan rantai sertifikat.

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

Akses penuh ke AWS Private CA

Kebijakan berikut memungkinkan pengguna untuk melakukan AWS Private CA tindakan apa pun.

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

Akses administrator ke semua sumber daya AWS

Kebijakan berikut memungkinkan pengguna untuk melakukan tindakan apa pun pada AWS sumber daya apa pun.

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

Praktik terbaik keamanan untuk akses lintas akun ke CA pribadi

AWS Private CA Administrator dapat berbagi CA dengan prinsipal (pengguna, peran, dll.) di akun lain. AWS Ketika saham telah diterima dan diterima, prinsipal dapat menggunakan CA untuk menerbitkan sertifikat entitas akhir menggunakan AWS Private CA atau AWS Certificate Manager sumber daya. Kepala sekolah dapat menggunakan CA untuk menerbitkan sertifikat CA bawahan menggunakan AWS Private CA.

> **Important**
>
> Biaya yang terkait dengan sertifikat yang dikeluarkan dalam skenario lintas akun ditagih ke AWS akun yang mengeluarkan sertifikat.

Untuk berbagi akses ke CA, AWS Private CA administrator dapat memilih salah satu dari metode berikut:

- Gunakan AWS Resource Access Manager (RAM) untuk berbagi CA sebagai sumber daya dengan prinsipal di akun lain atau dengan AWS Organizations. RAM adalah metode standar untuk berbagi AWS sumber daya di seluruh akun. Untuk informasi lebih lanjut tentang RAM, lihat Panduan Pengguna AWS RAM. Untuk informasi selengkapnya AWS Organizations, lihat Panduan AWS Organizations Pengguna.
- Gunakan AWS Private CA API atau CLI untuk melampirkan kebijakan berbasis sumber daya ke CA, sehingga memberikan akses ke prinsipal di akun lain. Untuk informasi selengkapnya, lihat Kebijakan berbasis sumber daya.

Bagian Mengontrol akses ke CA privat panduan ini menyediakan alur kerja untuk memberikan akses ke CA dalam skenario akun tunggal dan lintas akun.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah kebijakan izin yang Anda buat dan lampirkan secara manual ke sumber daya (dalam hal ini, CA pribadi), bukan ke identitas atau peran pengguna. Atau, alih-alih membuat kebijakan sendiri, Anda dapat menggunakan kebijakan AWS terkelola untuk AWS Private CA. Menggunakan AWS RAM untuk menerapkan kebijakan berbasis sumber daya, AWS Private CA administrator dapat berbagi akses ke CA dengan pengguna di AWS akun yang berbeda secara langsung atau melalui. AWS Organizations Sebagai alternatif, AWS Private CA administrator dapat menggunakan API PCA, dan PutPolicyGetPolicy, atau AWS CLI perintah terkait put-policy DeletePolicy, get-policy, dan delete-policy, untuk menerapkan dan mengelola kebijakan berbasis sumber daya.

Untuk informasi umum tentang kebijakan berbasis sumber daya, lihat Kebijakan Berbasis Identitas dan Kebijakan Berbasis Sumber Daya dan Mengontrol Akses Menggunakan Kebijakan.

Untuk melihat daftar kebijakan berbasis sumber daya AWS terkelola AWS Private CA, navigasikan ke pustaka izin terkelola di AWS Resource Access Manager konsol, lalu cari. CertificateAuthority

Seperti halnya kebijakan apa pun, sebelum Anda menerapkannya, kami sarankan untuk menerapkan kebijakan di lingkungan pengujian untuk memastikan bahwa kebijakan tersebut memenuhi persyaratan Anda.

AWS Certificate Manager (ACM) pengguna dengan akses bersama lintas akun ke CA pribadi dapat menerbitkan sertifikat terkelola yang ditandatangani oleh CA. Penerbit lintas akun dibatasi oleh kebijakan berbasis sumber daya dan hanya memiliki akses ke templat sertifikat entitas akhir berikut:

- EndEntityCertificate/V1
- EndEntityClientAuthCertificate/V1
- EndEntityServerAuthCertificate/V1
- BlankEndEntityCertificate_apiPassthrough/v1
- BlankEndEntityCertificate_apicsrPassthrough/v1
- Subordinatecacertificate_ 0/V1 PathLen

Contoh kebijakan

Bagian ini memberikan contoh kebijakan lintas akun untuk berbagai kebutuhan. Dalam semua kasus, pola perintah berikut digunakan untuk menerapkan kebijakan:

```
$ 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
```

Selain menentukan ARN CA, administrator memberikan ID AWS akun atau ID AWS Organizations yang akan diberikan akses ke CA. JSON dari masing-masing kebijakan berikut diformat sebagai file untuk keterbacaan, tetapi juga dapat diberikan sebagai argumen CLI inline.

> **Note**
>
> Struktur kebijakan berbasis sumber daya JSON yang ditunjukkan di bawah ini harus sama persis. Hanya bidang ID untuk prinsipal (nomor AWS akun atau ID AWS Organisasi) dan CA ARN yang dapat dikonfigurasi oleh pelanggan.

1. File: policy1.json — Berbagi akses ke CA dengan pengguna di akun yang berbeda

Ganti *5555555555* dengan ID AWS akun yang membagikan CA.

Untuk ARN sumber daya, ganti yang berikut ini dengan nilai Anda sendiri:

- *aws*- AWS Partisi. Misalnya,, `aws` `aws-us-gov`,`aws-cn`, dll.
- *us-east-1*- AWS Wilayah tempat sumber daya tersedia, seperti`us-west-1`.
- *111122223333*- ID AWS akun pemilik sumber daya.
- *11223344-1234-1122-2233-112233445566*- ID sumber daya dari otoritas sertifikat.

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

           "AWS":"555555555555"

        },
        "Action":[
           "acm-pca:DescribeCertificateAuthority",
           "acm-pca:GetCertificate",
           "acm-pca:GetCertificateAuthorityCertificate",
           "acm-pca:ListPermissions",
           "acm-pca:ListTags"

        ],

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

```
         "Resource":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
         "Condition":{
            "StringEquals":{
               "acm-pca:TemplateArn":"arn:aws:acm-pca:::template/
EndEntityCertificate/V1"
            }
         }
      }
   ]
}
```

2. File: policy2.json — Berbagi akses ke CA melalui AWS Organizations

   Ganti *o-a1b2c3d4z5* dengan ID. AWS Organizations

   Untuk ARN sumber daya, ganti yang berikut ini dengan nilai Anda sendiri:

   - *aws*- AWS Partisi. Misalnya,, `aws`aws-us-gov,aws-cn, dll.
   - *us-east-1*- AWS Wilayah tempat sumber daya tersedia, sepertius-west-1.
   - *111122223333*- ID AWS akun pemilik sumber daya.
   - *11223344-1234-1122-2233-112233445566*- ID sumber daya dari otoritas sertifikat.

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

Perlindungan data di AWS Private Certificate Authority

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di AWS Private Certificate Authority. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensyal dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk

memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan logging aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan AWS Private CA atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Kepatuhan penyimpanan dan keamanan kunci AWS Private CA pribadi

Kunci pribadi untuk CA pribadi disimpan dalam modul keamanan perangkat keras AWS terkelola (HSM). HSM mematuhi Persyaratan Keamanan FIPS PUB 140-2 Level 3 untuk Modul Kriptografi.

Enkripsi data di AWS Private CA Konektor untuk Direktori Aktif

AWS Private CA Konektor untuk AD menyimpan data konfigurasi pelanggan mengenai konektor, templat, pendaftaran direktori, nama utama layanan, dan entri kontrol akses grup templat. Data ini dienkripsi dalam perjalanan dan saat istirahat. Informasi tentang sertifikat yang dikeluarkan melalui Connector for AD dapat ditemukan menggunakan GetCertificatetindakan di AWS Private CA API. Tidak ada informasi mengenai sertifikat yang dikeluarkan, atau mengenai klien atau mesin yang meminta sertifikat, disimpan oleh AWS.

Validasi kepatuhan untuk AWS Private Certificate Authority

Auditor pihak ketiga menilai keamanan dan kepatuhan AWS Private Certificate Authority sebagai bagian dari beberapa program AWS kepatuhan. Program ini mencakup SOC, PCI, FedRAMP, HIPAA, dan lainnya.

Untuk daftar AWS layanan dalam lingkup program kepatuhan tertentu, lihat AWS Layanan dalam Lingkup menurut Program Kepatuhan Layanan AWS . Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan AWS Private CA ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Untuk organisasi yang diharuskan mengenkripsi bucket Amazon S3 mereka, topik berikut menjelaskan cara mengonfigurasi enkripsi untuk mengakomodasi aset: AWS Private CA
  - Mengenkripsi Laporan Audit Anda
  - Mengenkripsi CRL Anda
- Panduan Memulai Cepat Keamanan dan Kepatuhan Panduan Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan. AWS
- Arsitektur untuk Whitepaper Keamanan dan Kepatuhan HIPAA — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan untuk membuat aplikasi yang sesuai dengan HIPAA. AWS
- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- Mengevaluasi Sumber Daya dengan Aturan di Panduan Developer AWS Config — Layanan AWS Config menilai seberapa baik konfigurasi sumber daya Anda sesuai dengan praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— AWS Layanan ini memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

Menggunakan laporan audit dengan CA privat Anda

Anda dapat membuat laporan audit untuk mencantumkan semua sertifikat yang telah dikeluarkan atau dicabut oleh CA privat Anda. Laporan disimpan di bucket S3 baru atau yang sudah ada yang Anda tentukan pada input.

Untuk informasi tentang cara menambahkan perlindungan enkripsi ke laporan audit Anda, lihat Mengenkripsi laporan audit .

File laporan audit memiliki jalur dan nama file berikut. The ARN untuk bucket Amazon S3 adalah nilai untuk `bucket-name`. `CA_ID` adalah pengenal unik penerbitan CA. `UUID` adalah pengenal unik laporan audit.

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

Anda dapat membuat laporan baru setiap 30 menit dan mengunduhnya dari bucket Anda. Contoh berikut menunjukkan laporan terpisah CSV.

```
awsAccountId,requestedByServicePrincipal,certificateArn,serial,subject,notBefore,notAfter,issue
123456789012,,arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/
certificate_ID,00:11:22:33:44:55:66:77:88:99:aa:bb:cc:dd:ee:ff,"2.5.4.5=#012345678901,2.5.4.44=
 Company,L=Seattle,ST=Washington,C=US",2020-03-02T21:43:57+0000,2020-04-07T22:43:57+0000,2020-0
pca:::template/EndEntityCertificate/V1
123456789012,acm.amazonaws.com,arn:aws:acm-pca:region:account:certificate-
authority/CA_ID/
certificate/
certificate_ID,ff:ee:dd:cc:bb:aa:99:88:77:66:55:44:33:22:11:00,"2.5.4.5=#012345678901,2.5.4.44=
 Company,L=Seattle,ST=Washington,C=US",2020-03-02T20:53:39+0000,2020-04-07T21:53:39+0000,2020-0
pca:::template/EndEntityCertificate/V1
```

Contoh berikut menunjukkan laporan berformat JSON.

```
[
   {
      "awsAccountId":"123456789012",
      "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
      "serial":"00:11:22:33:44:55:66:77:88:99:aa:bb:cc:dd:ee:ff",

 "subject":"2.5.4.5=#012345678901,2.5.4.44=#0a1b3c4d,2.5.4.65=#0a1b3c4d5e6f,2.5.4.43=#0a1b3c4d5
 Company,L=Seattle,ST=Washington,C=US",
```

```
      "notBefore":"2020-02-26T18:39:57+0000",
      "notAfter":"2021-02-26T19:39:57+0000",
      "issuedAt":"2020-02-26T19:39:58+0000",
      "revokedAt":"2020-02-26T20:00:36+0000",
      "revocationReason":"UNSPECIFIED",
      "templateArn":"arn:aws:acm-pca:::template/EndEntityCertificate/V1"
   },
   {
      "awsAccountId":"123456789012",
      "requestedByServicePrincipal":"acm.amazonaws.com",
      "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
      "serial":"ff:ee:dd:cc:bb:aa:99:88:77:66:55:44:33:22:11:00",

 "subject":"2.5.4.5=#012345678901,2.5.4.44=#0a1b3c4d,2.5.4.65=#0a1b3c4d5e6f,2.5.4.43=#0a1b3c4d5
 Company,L=Seattle,ST=Washington,C=US",
      "notBefore":"2020-01-22T20:10:49+0000",
      "notAfter":"2021-01-17T21:10:49+0000",
      "issuedAt":"2020-01-22T21:10:49+0000",
      "templateArn":"arn:aws:acm-pca:::template/EndEntityCertificate/V1"
   }
]
```

> **Note**
>
> Saat AWS Certificate Manager memperbarui sertifikat, laporan audit CA pribadi mengisi `requestedByServicePrincipal` bidang dengan. `acm.amazonaws.com` Ini menunjukkan bahwa AWS Certificate Manager layanan disebut `IssueCertificate` tindakan AWS Private CA API atas nama pelanggan untuk memperbarui sertifikat.

Menyiapkan bucket Amazon S3 untuk laporan audit

Untuk menyimpan laporan audit, Anda perlu menyiapkan bucket Amazon S3. Untuk informasi selengkapnya, lihat Bagaimana Cara Membuat bucket S3?

Bucket S3 Anda harus diamankan dengan kebijakan izin yang dilampirkan. Pengguna resmi dan kepala layanan memerlukan `Put` izin AWS Private CA untuk mengizinkan menempatkan objek di ember, dan `Get` izin untuk mengambilnya. Kami menyarankan Anda menerapkan kebijakan yang ditunjukkan di bawah ini, yang membatasi akses ke AWS akun dan ARN CA pribadi. Untuk informasi selengkapnya, lihat Menambahkan kebijakan bucket menggunakan konsol Amazon S3.

> **Note**
>
> Selama prosedur konsol untuk membuat laporan audit, Anda dapat memilih untuk mengizinkan AWS Private CA membuat bucket baru dan menerapkan kebijakan izin default. Kebijakan default tidak berlaku `SourceArn` pembatasan pada CA dan oleh karena itu lebih permisif daripada kebijakan yang direkomendasikan. Jika Anda memilih default, Anda selalu dapat memodifikasinya nanti.

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

Membuat laporan audit

Anda dapat membuat laporan audit baik dari konsol tersebut atau AWS CLI.

Untuk membuat laporan audit (konsol)

1. Masuk ke AWS akun Anda dan buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home.
2. Pada halaman Private Certificate Authorities, pilih CA pribadi Anda dari daftar.
3. Dari menu Tindakan, pilih Hasilkan laporan audit.
4. Di bawah tujuan laporan Audit, untuk Buat bucket S3 baru? , pilih Ya dan ketik nama bucket unik, atau pilih Tidak dan pilih bucket yang ada dari daftar.

   Jika Anda memilih Ya, AWS Private CA buat dan lampirkan kebijakan default ke bucket Anda. Jika Anda memilih Tidak, Anda harus melampirkan kebijakan ke bucket agar dapat membuat laporan audit. Gunakan pola kebijakan yang dijelaskan dalamMenyiapkan bucket Amazon S3 untuk laporan audit. Untuk informasi tentang melampirkan kebijakan, lihat Menambahkan kebijakan bucket menggunakan konsol Amazon S3
5. Di bawah format Output, pilih JSON untuk Notasi JavaScript Objek atau CSV untuk nilai yang dipisahkan koma.
6. Pilih Hasilkan laporan audit.

Untuk membuat laporan audit (AWS CLI)

1. Jika Anda belum memiliki bucket S3 untuk digunakan, buat satu.
2. Lampirkan kebijakan ke bucket Anda Gunakan pola kebijakan yang dijelaskan dalamMenyiapkan bucket Amazon S3 untuk laporan audit. Untuk informasi tentang melampirkan kebijakan, lihat Menambahkan kebijakan bucket menggunakan konsol Amazon S3
3. Gunakan perintah create-certificate-authority-audit-report untuk membuat laporan audit dan menempatkannya di bucket S3 yang sudah disiapkan.

```
$ 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
```

Mengambil laporan audit

Untuk mengambil laporan audit untuk pemeriksaan, gunakan konsol Amazon S3, API, CLI, atau SDK. Untuk informasi selengkapnya, lihat Mengunduh objek di Panduan Pengguna Amazon Simple Storage Service.

Mengenkripsi laporan audit

Anda dapat mengonfigurasi enkripsi secara opsional di bucket Amazon S3 yang berisi laporan audit Anda. AWS Private CA mendukung dua mode enkripsi untuk aset di S3:

- Enkripsi sisi server otomatis dengan kunci AES-256 terkelola Amazon S3.
- Enkripsi terkelola pelanggan menggunakan AWS Key Management Service dan AWS KMS key dikonfigurasi dengan spesifikasi Anda.

> **Note**
>
> AWS Private CA tidak mendukung penggunaan kunci KMS default yang dihasilkan secara otomatis oleh S3.

Prosedur berikut menjelaskan cara menyiapkan setiap opsi enkripsi.

Untuk mengkonfigurasi enkripsi otomatis

Selesaikan langkah-langkah berikut untuk mengaktifkan enkripsi sisi server S3.

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Di tabel Bucket, pilih ember yang akan menampung AWS Private CA aset Anda.
3. Pada halaman untuk bucket Anda, pilih tab Properti.
4. Pilih kartu Enkripsi default.
5. Pilih Aktifkan.
6. Pilih Kunci Amazon S3 (SSE-S3).
7. Pilih Simpan Perubahan.

Untuk mengkonfigurasi enkripsi kustom

Selesaikan langkah-langkah berikut untuk mengaktifkan enkripsi menggunakan kunci kustom.

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Di tabel Bucket, pilih ember yang akan menampung AWS Private CA aset Anda.
3. Pada halaman untuk bucket Anda, pilih tab Properti.
4. Pilih kartu Enkripsi default.
5. Pilih Aktifkan.
6. Pilih AWS Key Management Service kunci (SSE-KMS).
7. Pilih Pilih dari AWS KMS kunci Anda atau Masukkan AWS KMS key ARN.
8. Pilih Simpan Perubahan.
9. (Opsional) Jika Anda belum memiliki kunci KMS, buat satu menggunakan perintah AWS CLI create-key berikut:

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

Outputnya berisi ID kunci dan Nama Sumber Daya Amazon (ARN) dari kunci KMS. Berikut ini adalah contoh output:

```
{
    "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. Dengan menggunakan langkah-langkah berikut, Anda memberikan izin kepada kepala AWS Private CA layanan untuk menggunakan kunci KMS. Secara default, semua kunci KMS bersifat pribadi; hanya pemilik sumber daya yang dapat menggunakan kunci KMS untuk mengenkripsi dan mendekripsi data. Namun, pemilik sumber daya dapat memberikan izin untuk mengakses kunci KMS ke pengguna dan sumber daya lain. Prinsipal layanan harus berada di Wilayah yang sama dengan tempat kunci KMS disimpan.

a. Pertama, simpan kebijakan default untuk kunci KMS Anda seperti `policy.json` menggunakan get-key-policyperintah berikut:

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

b. Buka file `policy.json` di editor teks. Pilih salah satu pernyataan kebijakan berikut dan tambahkan ke kebijakan yang ada.

Jika kunci bucket Amazon S3 diaktifkan, gunakan pernyataan berikut:

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

Jika kunci bucket Amazon S3 dinonaktifkan, gunakan pernyataan berikut:

```
{
   "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. Terakhir, terapkan kebijakan yang diperbarui menggunakan put-key-policyperintah berikut:

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

Keamanan infrastruktur di AWS Private Certificate Authority

Sebagai layanan terkelola, AWS Private Certificate Authority dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses AWS Private CA melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani dengan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan pengguna utama IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

AWS Private CA Titik akhir VPC ()AWS PrivateLink

Anda dapat membuat koneksi pribadi antara VPC Anda dan AWS Private CA dengan mengonfigurasi titik akhir VPC antarmuka. Endpoint antarmuka didukung oleh AWS PrivateLink, sebuah teknologi untuk mengakses AWS Private CA operasi API secara pribadi. AWS PrivateLink merutekan semua lalu lintas jaringan antara VPC Anda dan AWS Private CA melalui jaringan Amazon, menghindari paparan di internet terbuka. Setiap titik akhir VPC diwakili oleh satu atau lebih antarmuka jaringan elastis dengan alamat IP pribadi di subnet VPC Anda.

Titik akhir VPC antarmuka menghubungkan VPC Anda secara langsung AWS Private CA tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi. AWS Direct Connect Instans di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan AWS Private CA API.

Untuk menggunakan AWS Private CA melalui VPC Anda, Anda harus terhubung dari instance yang ada di dalam VPC. Atau, Anda dapat menghubungkan jaringan pribadi Anda ke VPC Anda dengan menggunakan AWS Virtual Private Network (AWS VPN) atau. AWS Direct Connect Untuk selengkapnya AWS VPN, lihat Koneksi VPN di Panduan Pengguna Amazon VPC. Untuk selengkapnya AWS Direct Connect, lihat Membuat Sambungan di Panduan AWS Direct Connect Pengguna.

AWS Private CA tidak memerlukan penggunaan AWS PrivateLink, tetapi kami merekomendasikannya sebagai lapisan keamanan tambahan. Untuk informasi selengkapnya tentang AWS PrivateLink dan titik akhir VPC, lihat Mengakses Layanan Melalui. AWS PrivateLink

Pertimbangan untuk titik akhir AWS Private CA VPC

Sebelum Anda mengatur titik akhir VPC antarmuka untuk AWS Private CA, perhatikan pertimbangan berikut:

- AWS Private CA mungkin tidak mendukung titik akhir VPC di beberapa Availability Zone. Saat Anda membuat VPC endpoint, periksa terlebih dahulu dukungan di konsol manajemen. Availability Zone yang tidak didukung ditandai "Layanan tidak didukung di Availability Zone ini".
- VPC endpoint saat ini tidak mendukung permintaan lintas Wilayah. Pastikan bahwa Anda membuat titik akhir Anda di Wilayah yang sama tempat Anda berencana untuk mengeluarkan panggilan API ke AWS Private CA.
- VPC endpoint hanya support Amazon yang disediakan DNS melalui Amazon Route 53. Jika Anda ingin menggunakan DNS Anda sendiri, Anda dapat menggunakan penerusan DNS bersyarat. Untuk informasi selengkapnya, lihat Pengaturan DHCP dalam Panduan Pengguna Amazon VPC.

- Grup keamanan yang dilampirkan ke VPC endpoint harus mengizinkan koneksi masuk pada port 443 dari subnet privat VPC.
- AWS Certificate Manager tidak mendukung titik akhir VPC.
- Endpoint FIPS (dan Wilayah mereka) tidak mendukung VPC endpoint.

AWS Private CA API saat ini mendukung titik akhir VPC sebagai berikut: Wilayah AWS

- AS Timur (Ohio)
- AS Timur (Virginia Utara)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Afrika (Cape Town)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Kanada (Pusat)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Paris)
- Eropa (Stockholm)
- Eropa (Milan)
- Israel (Tel Aviv)
- Timur Tengah (Bahrain)
- Amerika Selatan (Sao Paulo)

Membuat titik akhir VPC untuk AWS Private CA

Anda dapat membuat titik akhir VPC untuk AWS Private CA layanan menggunakan konsol VPC di https://console.aws.amazon.com/vpc/ atau. AWS Command Line Interface Untuk informasi selengkapnya, lihat prosedur Membuat Titik Akhir Antarmuka di Panduan Pengguna Amazon VPC. AWS Private CA mendukung panggilan ke semua operasi API di dalam VPC Anda.

Jika Anda telah mengaktifkan nama host DNS pribadi untuk titik akhir, maka titik akhir default sekarang akan diselesaikan ke AWS Private CA titik akhir VPC Anda. Untuk daftar lengkap titik akhir layanan default, lihat Titik Akhir dan Kuota Layanan.

Jika Anda belum mengaktifkan nama host DNS privat, Amazon VPC menyediakan nama titik akhir DNS yang dapat Anda gunakan dalam format berikut:

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

> **Note**
>
> *Wilayah* nilai mewakili pengenal Wilayah untuk AWS Wilayah yang didukung oleh AWS Private CA, seperti `us-east-2` untuk Wilayah Timur AS (Ohio). Untuk daftar AWS Private CA, lihat AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.

Untuk informasi selengkapnya, lihat Titik akhir AWS Private CA VPC (AWS PrivateLink) di Panduan Pengguna Amazon VPC.

Membuat kebijakan titik akhir VPC untuk AWS Private CA

Anda dapat membuat kebijakan untuk titik akhir VPC Amazon AWS Private CA untuk menentukan hal berikut:

- Prinsip-prinsip yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan
- Sumber daya yang dapat digunakan untuk mengambil tindakan

Untuk informasi selengkapnya, lihat Mengendalikan Akses ke Layanan dengan Titik Akhir VPC dalam Panduan Pengguna Amazon VPC.

Contoh - Kebijakan titik akhir VPC untuk tindakan AWS Private CA

Ketika dilampirkan ke titik akhir, kebijakan berikut memberikan akses untuk semua prinsipal untuk AWS Private CA tindakan`IssueCertificate`,,,,`DescribeCertificateAuthority`, `GetCertificate` dan. `GetCertificateAuthorityCertificate` `ListPermissions` `ListTags` Sumber daya di setiap bait adalah CA privat. Bait pertama mengizinkan pembuatan sertifikat entitas akhir menggunakan CA privat dan templat sertifikat yang ditentukan. Jika Anda tidak ingin mengontrol templat yang digunakan, bagian `Condition` tidak diperlukan. Namun, menghapus ini memungkinkan semua entitas keamanan untuk membuat sertifikat CA serta sertifikat entitas akhir.

```
{
      "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"
```

```
            ]
         }
      ]
   }
```

Penebangan dan pemantauan untuk AWS Private Certificate Authority

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja AWS Private Certificate Authority dan AWS solusi Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi.

Topik berikut menjelaskan alat AWS pemantauan cloud yang tersedia untuk digunakan. AWS Private CA

Topik

CloudWatch Metrik yang didukung

Amazon CloudWatch adalah layanan pemantauan untuk AWS sumber daya. Anda dapat menggunakannya CloudWatch untuk mengumpulkan dan melacak metrik, menyetel alarm, dan bereaksi secara otomatis terhadap perubahan sumber daya Anda AWS . CloudWatch metrik diterbitkan setidaknya sekali.

AWS Private CA mendukung CloudWatch metrik berikut.

Metrik	Deskripsi
`CRLGenerated`	Daftar pencabutan sertifikat (CRL) telah dibuat. Metrik ini hanya berlaku untuk CA privat.
`MisconfiguredCRLBucket`	Bucket S3 yang ditentukan untuk CRL tidak dikonfigurasi dengan benar. Periksa kebijakan

Metrik	Deskripsi
	bucket. Metrik ini hanya berlaku untuk CA privat.
`Time`	Waktu dalam milidetik antara permintaan penerbitan dan penyelesaian (atau kegagalan) penerbitan. Metrik ini hanya berlaku untuk IssueCertificateoperasi.
`Success`	Sertifikat berhasil diterbitkan. Metrik ini hanya berlaku untuk IssueCertificateoperasi.
`Failure`	Operasi gagal. Metrik ini hanya berlaku untuk IssueCertificateoperasi.

Untuk informasi selengkapnya tentang CloudWatch metrik, lihat topik berikut:

- Menggunakan CloudWatch Metrik Amazon
- Membuat CloudWatch Alarm Amazon

Menggunakan CloudWatch Acara

Anda dapat menggunakan Amazon CloudWatch Events untuk mengotomatiskan AWS layanan Anda dan merespons secara otomatis peristiwa sistem seperti masalah ketersediaan aplikasi atau perubahan sumber daya. Acara dari AWS layanan dikirimkan ke CloudWatch Acara dalam waktu nyaris nyata. Anda dapat menulis aturan sederhana untuk menunjukkan acara mana yang menarik bagi Anda dan tindakan otomatis yang harus diambil ketika suatu acara cocok dengan aturan. CloudWatch Acara diterbitkan setidaknya sekali. Untuk informasi selengkapnya, lihat Membuat Aturan CloudWatch Peristiwa yang Memicu Peristiwa.

CloudWatch Acara diubah menjadi tindakan menggunakan Amazon EventBridge. Dengan EventBridge, Anda dapat menggunakan peristiwa untuk memicu target termasuk AWS Lambda fungsi, AWS Batch pekerjaan, topik Amazon SNS, dan banyak lainnya. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge?

Berhasil atau gagal saat membuat CA privat

Peristiwa ini dipicu oleh CreateCertificateAuthorityoperasi.

Berhasil

Pada keberhasilan, operasi mengembalikan ARN CA baru.

```
{
   "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"
   }
}
```

Kegagalan

Pada kegagalan, operasi mengembalikan ARN untuk CA. Menggunakan ARN, Anda dapat menelepon DescribeCertificateAuthorityuntuk menentukan status CA.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA Creation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T19:14:56Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   ],
```

```
   "detail":{
      "result":"failure"
   }
}
```

Keberhasilan atau kegagalan saat menerbitkan sertifikat

Peristiwa ini dipicu oleh IssueCertificateoperasi.

Berhasil

Pada keberhasilan, operasi mengembalikan ARN CA dan sertifikat baru.

```
{
   "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"
   }
}
```

Kegagalan

Pada kegagalan, operasi mengembalikan sertifikat ARN dan ARN CA. Dengan sertifikat ARN, Anda dapat menelepon GetCertificateuntuk melihat alasan kegagalan.

```
{
   "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"
   }
}
```

Keberhasilan saat mencabut sertifikat

Peristiwa ini dipicu oleh RevokeCertificateoperasi.

Tidak ada peristiwa yang dikirim jika pencabutan gagal atau jika sertifikat telah dicabut.

Sukses

Pada keberhasilan, operasi mengembalikan ARN CA dan sertifikat akan dicabut.

```
{
   "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"
   }
}
```

Keberhasilan atau kegagalan saat membuat CRL

Peristiwa ini dipicu oleh RevokeCertificateoperasi, yang akan menghasilkan pembuatan daftar pencabutan sertifikat (CRL).

Berhasil

Pada keberhasilan, operasi akan mengembalikan ARN CA yang berkaitan dengan CRL.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA CRL Generation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T21:07:08Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   ],
   "detail":{
      "result":"success"
   }
}
```

Kegagalan 1 – CRL tidak dapat disimpan ke Amazon S3 karena kesalahan izin

Periksa izin bucket Amazon S3 Anda jika kesalahan ini terjadi.

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

Kegagalan 2 – CRL tidak dapat disimpan ke Amazon S3 karena kesalahan internal

Coba operasi lagi jika kesalahan ini terjadi.

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

Kegagalan 3 — AWS Private CA gagal membuat CRL

Untuk memecahkan masalah kesalahan ini, periksa metrik AndaCloudWatch .

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

Keberhasilan atau kegagalan saat membuat laporan audit CA

Peristiwa ini dipicu oleh CreateCertificateAuthorityAuditReportoperasi.

Berhasil

Jika berhasil, operasi mengembalikan ARN CA dan ID laporan audit.

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA Audit Report Generation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-04T21:54:20Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "audit_report_ID"
   ],
   "detail":{
      "result":"success"
   }
}
```

Kegagalan

Laporan audit dapat gagal jika AWS Private CA tidak memiliki PUT izin di bucket Amazon S3, saat enkripsi diaktifkan di bucket, atau karena alasan lain.

```
{
   "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"
   }
}
```

Menggunakan CloudTrail

Anda dapat menggunakan AWS CloudTrailuntuk merekam panggilan API yang dibuat oleh AWS Private Certificate Authority. Untuk informasi selengkapnya, lihat topik berikut.

Topik

- Membuat kebijakan
- Mengambil kebijakan
- Menghapus kebijakan
- Membuat otoritas sertifikasi (CA)
- GenerateCRL
- HasilkanOCSResponse
- Membuat laporan audit
- Menghapus otoritas sertifikasi (CA)
- Memulihkan otoritas sertifikasi (CA)
- Menggambarkan otoritas sertifikasi (CA)
- Mengambil sertifikat otoritas sertifikasi (CA)
- Mengambil permintaan penandatanganan otoritas sertifikasi (CA)
- Mengambil sertifikat
- Mengimpor sertifikat otoritas sertifikasi (CA)
- Menerbitkan sertifikat

- Daftar otoritas sertifikasi (CA)
- Mencantumkan tanda
- Mencabut sertifikat
- Penandaan Private Certificate Authority
- Menghapus tanda dari Private Certificate Authority
- Memperbarui otoritas sertifikasi (CA)

Membuat kebijakan

CloudTrail Contoh berikut menunjukkan hasil panggilan ke PutPolicyoperasi.


```
{
   "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"
}
```

Mengambil kebijakan

CloudTrail Contoh berikut menunjukkan hasil panggilan ke GetPolicyoperasi.

```
{
   "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"
}
```

Menghapus kebijakan

CloudTrail Contoh berikut menunjukkan hasil panggilan ke DeletePolicyoperasi.

```
{
   "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"
}
```

Membuat otoritas sertifikasi (CA)

CloudTrail Contoh berikut menunjukkan hasil panggilan ke CreateCertificateAuthorityoperasi.

```
{
   "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

CloudTrail Contoh berikut menunjukkan catatan untuk acara GenerateCRL.

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

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

HasilkanOCSResponse

CloudTrail Contoh berikut menunjukkan catatan untuk acara generateOcspResponse.

```
{
   "eventVersion":"1.08",
   "userIdentity":{
      "accountId":"account",
      "invokedBy":"acm-pca.amazonaws.com"
   },
   "eventTime":"2021-02-08T23:52:29Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"GenerateOCSPResponse",
   "awsRegion":"region",
   "sourceIPAddress":"acm-pca.amazonaws.com",
   "userAgent":"acm-pca.amazonaws.com",
   "eventID":"01234567-89ab-cdef-0123-456789abcdef",
   "readOnly":false,
   "resources":[
```

```
      {
         "type":"AWS::ACMPCA::Certificate",
         "ARN":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
      }
   ]
}
```

Membuat laporan audit

CloudTrail Contoh berikut menunjukkan hasil panggilan ke CreateCertificateAuthorityAuditReportoperasi.

```
{
   "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"
}
```

Menghapus otoritas sertifikasi (CA)

CloudTrail Contoh berikut menunjukkan hasil panggilan ke DeleteCertificateAuthorityoperasi. Dalam contoh ini, otoritas sertifikasi (CA) tidak dapat dihapus karena dalam negara bagian `ACTIVE`.


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


Memulihkan otoritas sertifikasi (CA)

CloudTrail Contoh berikut menunjukkan hasil panggilan ke RestoreCertificateAuthorityoperasi. Dalam contoh ini, otoritas sertifikasi (CA) tidak dapat dihapus karena dalam negara bagian `DELETED`.


```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
```

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

Menggambarkan otoritas sertifikasi (CA)

CloudTrail Contoh berikut menunjukkan hasil panggilan ke DescribeCertificateAuthorityoperasi.

```
{
   "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"
}
```

Mengambil sertifikat otoritas sertifikasi (CA)

CloudTrail Contoh berikut menunjukkan hasil panggilan ke GetCertificateAuthorityCertificateoperasi.

```
{
   "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"
}
```

Mengambil permintaan penandatanganan otoritas sertifikasi (CA)

CloudTrail Contoh berikut menunjukkan hasil panggilan ke GetCertificateAuthorityCsroperasi.

```
{
   "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"
}
```

Mengambil sertifikat

CloudTrail Contoh berikut menunjukkan hasil panggilan ke GetCertificateoperasi.

```
{
   "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"
}
```

Mengimpor sertifikat otoritas sertifikasi (CA)

CloudTrail Contoh berikut menunjukkan hasil panggilan ke ImportCertificateAuthorityCertificateoperasi.

```
{
   "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"
}
```

Menerbitkan sertifikat

CloudTrail Contoh berikut menunjukkan hasil panggilan ke IssueCertificateoperasi.

```
{
   "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"
}
```

Daftar otoritas sertifikasi (CA)

CloudTrail Contoh berikut menunjukkan hasil panggilan ke ListCertificateAuthoritiesoperasi.

```
{
   "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"
}
```

Mencantumkan tanda

CloudTrail Contoh berikut menunjukkan hasil panggilan ke ListTagsoperasi.

```
{
   "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"
}
```

Mencabut sertifikat

CloudTrail Contoh berikut menunjukkan hasil panggilan ke RevokeCertificateoperasi.

```
{
   "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"
}
```

Penandaan Private Certificate Authority

CloudTrail Contoh berikut menunjukkan hasil panggilan ke TagCertificateAuthorityoperasi.

```
{
   "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"
}
```

Menghapus tanda dari Private Certificate Authority

CloudTrail Contoh berikut menunjukkan hasil panggilan ke UntagCertificateAuthorityoperasi.

```
{
   "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"
}
```

Memperbarui otoritas sertifikasi (CA)

CloudTrail Contoh berikut menunjukkan hasil panggilan ke UpdateCertificateAuthorityoperasi.

```
{
   "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"
}
```

Merencanakan AWS Private CA penyebaran Anda

AWS Private CA memberi Anda kontrol lengkap berbasis cloud atas PKI pribadi organisasi Anda (infrastruktur kunci publik), mulai dari otoritas sertifikat root (CA), melalui CA bawahan, hingga sertifikat entitas akhir. Perencanaan menyeluruh sangat penting untuk PKI yang aman, dapat dipelihara, dapat diperluas, dan sesuai dengan kebutuhan organisasi Anda. Bagian ini memberikan panduan tentang merancang hierarki CA, mengelola CA privat Anda dan siklus hidup sertifikat entitas akhir privat, dan menerapkan praktik terbaik untuk keamanan.

Bagian ini menjelaskan cara AWS Private CA mempersiapkan penggunaan sebelum Anda membuat Private Certificate Authority (CA). Ini juga menjelaskan opsi untuk menambahkan dukungan pencabutan melalui Online Certificate Status Protocol (OCSP) atau daftar pencabutan sertifikat (CRL).

Selain itu, Anda harus menentukan apakah organisasi Anda lebih suka meng-host kredenal CA root pribadinya di tempat daripada dengan. AWS Dalam hal ini, Anda perlu mengatur dan mengamankan PKI pribadi yang dikelola sendiri sebelum menggunakan. AWS Private CA Dalam skenario ini, Anda kemudian membuat CA bawahan yang AWS Private CA didukung oleh CA induk di luar. AWS Private CA Untuk informasi selengkapnya, lihat Menginstal sertifikat CA bawahan yang ditandatangani oleh CA induk eksternal.

Topik

- Menyiapkan AWS akun Anda dan AWS CLI
- Merancang hierarki CA
- Mengelola siklus hidup CA privat
- Menyiapkan metode pencabutan sertifikat
- Mode otoritas sertifikat
- Perencanaan ketahanan

Menyiapkan AWS akun Anda dan AWS CLI

Jika Anda belum menjadi pelanggan Amazon Web Services (AWS), Anda harus mendaftar untuk dapat menggunakannya AWS Private CA. Akun Anda secara otomatis memiliki akses ke semua layanan yang tersedia, namun Anda hanya akan dikenakan biaya untuk layanan yang Anda gunakan.

> **Note**
>
> AWS Private CA tidak tersedia di Tingkat AWS Gratis.

Topik

- Mendaftar untuk Akun AWS
- Buat pengguna dengan akses administratif
- Instal AWS Command Line Interface

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.

2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Instal AWS Command Line Interface

Jika Anda belum menginstal AWS CLI tetapi ingin menggunakannya, ikuti petunjuk di AWS Command Line Interface. Dalam panduan ini, kami berasumsi bahwa Anda telah mengonfigurasi titik akhir, Wilayah, dan detail otentikasi Anda, dan kami menghilangkan parameter ini dari perintah sampel.

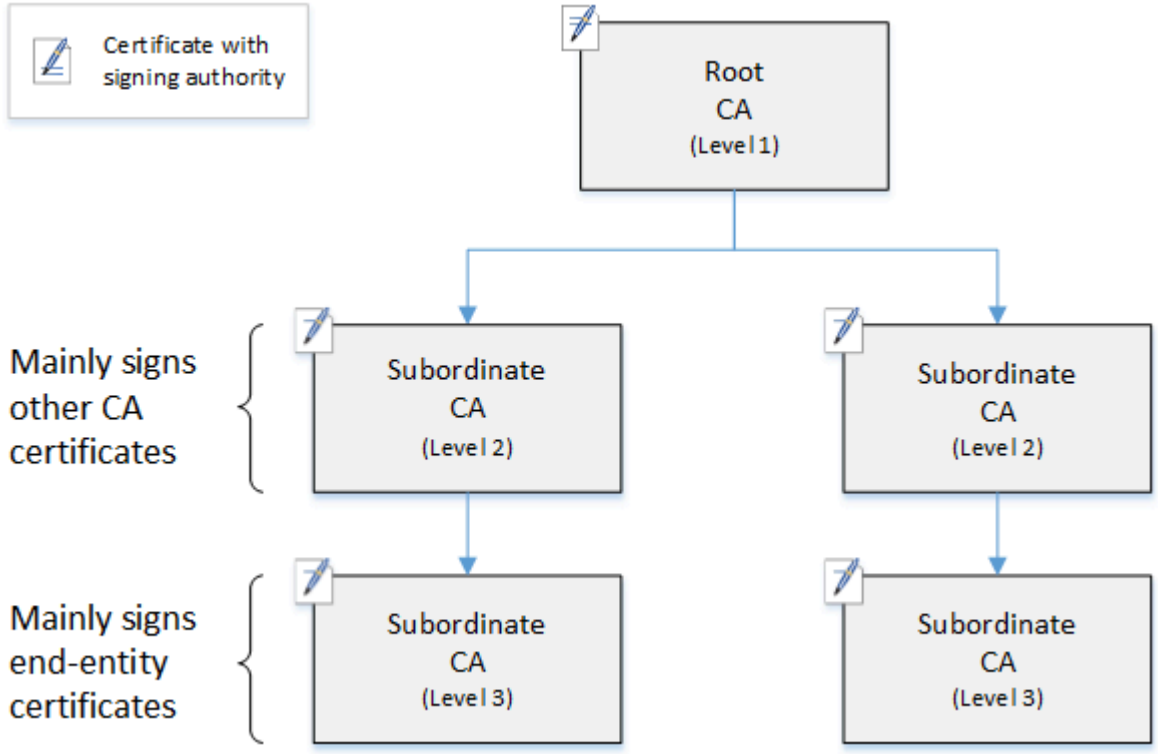
Merancang hierarki CA

Dengan AWS Private CA, Anda dapat membuat hierarki otoritas sertifikat hingga lima level. CA akar, di bagian atas pohon hierarki, dapat memiliki sejumlah cabang. CA akar dapat memiliki sebanyak empat tingkat CA bawahan di setiap cabang. Anda juga dapat membuat beberapa hierarki, masing-masing dengan akarnya sendiri.

Hierarki CA yang dirancang dengan baik menawarkan manfaat berikut:

- Kontrol keamanan terperinci yang sesuai untuk setiap CA
- Pembagian tugas administratif untuk penyeimbangan beban dan keamanan yang lebih baik
- Penggunaan CA dengan kepercayaan terbatas yang dapat dibatalkan untuk operasi sehari-hari
- Masa validitas dan batas jalur sertifikat

Diagram berikut mengilustrasikan hierarki CA tiga tingkat yang sederhana.



Setiap CA di pohon didukung oleh sertifikat X.509 v3 dengan otoritas penandatanganan (dilambangkan dengan ikon). pen-and-paper Ini berarti bahwa sebagai CA, sertifikat lain yang berada di bawahnya dapat ditandatangani. Ketika CA menandatangani sertifikat CA tingkat yang lebih rendah, itu memberikan otoritas terbatas yang dapat dibatalkan pada sertifikat yang ditandatangani. CA akar di level 1 menandatangani sertifikat CA bawahan tingkat tinggi di level 2. CA ini, pada gilirannya, menandatangani sertifikat untuk CA di level 3 yang digunakan oleh administrator PKI (infrastruktur kunci publik) yang mengelola sertifikat entitas akhir.

Keamanan dalam hierarki CA harus dikonfigurasikan menjadi yang terkuat di bagian atas hierarki. Pengaturan ini melindungi sertifikat CA akar dan kunci privatnya. CA akar menambatkan kepercayaan untuk semua CA bawahan dan sertifikat entitas akhir di bawahnya. Sementara kerusakan lokal dapat dihasilkan dari kompromi sertifikat entitas akhir, kompromi akar menghancurkan kepercayaan di seluruh PKI. Root dan CA bawahan tingkat tinggi jarang digunakan (biasanya untuk menandatangani sertifikat CA lainnya). Akibatnya, mereka dikontrol dengan ketat dan diaudit untuk memastikan risiko kompromi yang lebih rendah. Di tingkat hierarki yang lebih rendah, keamanan tidak terlalu ketat. Pendekatan ini memungkinkan tugas administratif rutin menerbitkan dan mencabut sertifikat entitas akhir untuk pengguna, host komputer, dan layanan perangkat lunak.

> **Note**
>
> Menggunakan CA akar untuk menandatangani sertifikat bawahan adalah peristiwa langka yang terjadi hanya dalam beberapa keadaan:
>
> - Saat PKI dibuat
> - Ketika otoritas sertifikat tingkat tinggi perlu diganti
> - Saat penanggap daftar pencabutan sertifikat (CRL) atau Protokol Status Sertifikat Online (OCSP) perlu dikonfigurasi
>
> Akar dan CA tingkat tinggi lainnya memerlukan proses operasional dan protokol kontrol akses yang sangat aman.

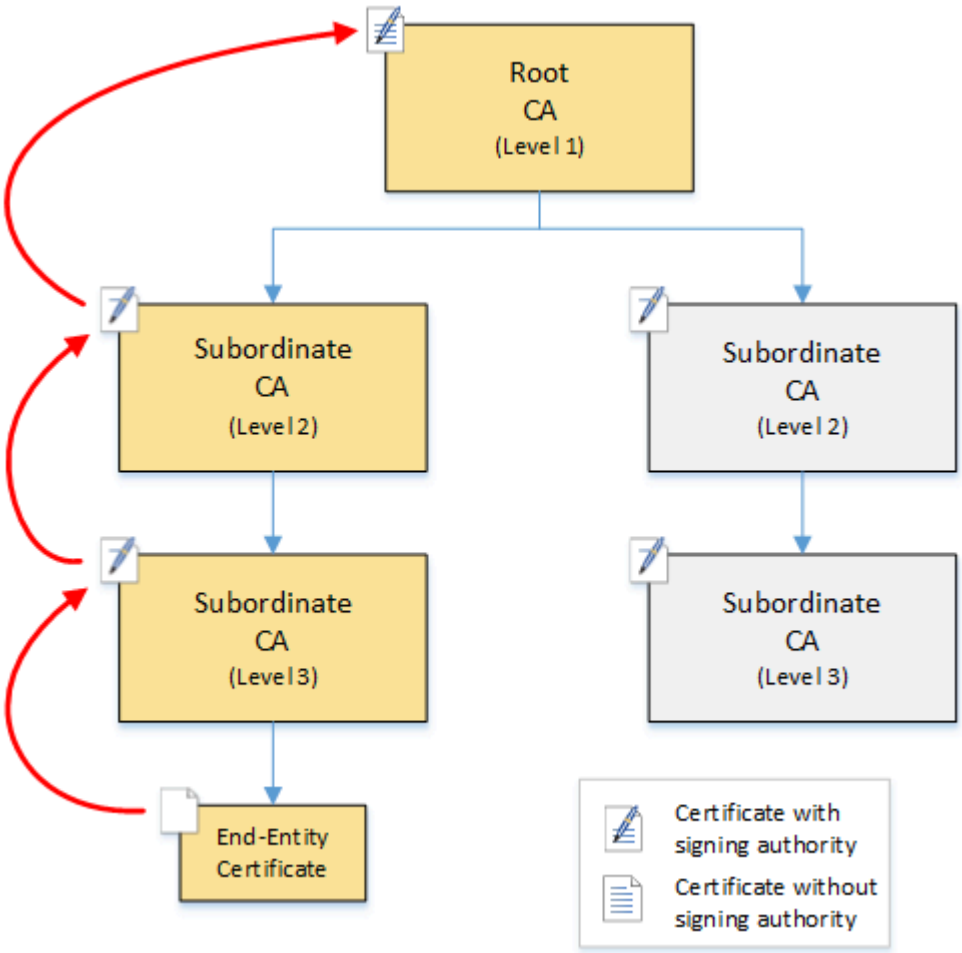
Topik

Memvalidasi sertifikat entitas akhir

Sertifikat entitas akhir mendapatkan kepercayaannya dari jalur sertifikasi yang mengarah kembali melalui CA bawahan ke CA akar. Ketika disajikan dengan sertifikat entitas akhir, peramban web atau klien lain mencoba membangun rantai kepercayaan. Misalnya, mungkin memeriksa untuk melihat bahwa nama khusus penerbit sertifikat dan nama khusus subjek cocok dengan bidang yang sesuai dari sertifikat CA penerbit. Pencocokan akan berlanjut di setiap tingkat berturut-turut ke atas hierarki hingga klien mencapai akar tepercaya yang terkandung dalam penyimpanan kepercayaannya.

Penyimpanan kepercayaan adalah pustaka CA tepercaya yang berisi peramban atau sistem operasi. Untuk PKI privat, TI organisasi Anda harus memastikan bahwa setiap peramban atau sistem sebelumnya telah menambahkan CA akar privat ke penyimpanan kepercayaannya. Jika tidak, jalur sertifikasi tidak dapat divalidasi, yang mengakibatkan kesalahan klien.

Diagram berikutnya menunjukkan jalur validasi yang diikuti peramban saat disajikan dengan sertifikat X.509 entitas akhir. Perhatikan bahwa sertifikat entitas akhir tidak memiliki otoritas penandatanganan dan hanya berfungsi untuk mengautentikasi entitas yang memilikinya.



Peramban memeriksa sertifikat entitas akhir. Peramban menemukan bahwa sertifikat menawarkan tanda tangan dari CA bawahan (level 3) sebagai kredensial kepercayaannya. Sertifikat untuk CA bawahan harus disertakan dalam file PEM yang sama. Atau, mereka juga dapat berada dalam file terpisah yang berisi sertifikat yang membentuk rantai kepercayaan. Setelah menemukan ini, peramban memeriksa sertifikat CA bawahan (tingkat 3) dan menemukan bahwa sertifikat CA bawahan menawarkan tanda tangan dari CA bawahan (tingkat 2). Pada gilirannya, CA bawahan (level 2) menawarkan tanda tangan dari CA akar (level 1) sebagai kredensial kepercayaannya. Jika peramban menemukan salinan sertifikat CA akar privat yang telah diinstal sebelumnya di penyimpanan kepercayaannya, peramban akan memvalidasi sertifikat entitas akhir sebagai tepercaya.

Biasanya, peramban juga memeriksa setiap sertifikat terhadap daftar pencabutan sertifikat (CRL). Sertifikat yang kedaluwarsa, dicabut, atau salah konfigurasi ditolak dan validasi gagal.

Merencanakan struktur hierarki CA

Secara umum, hierarki CA Anda harus mencerminkan struktur organisasi Anda. Bertujuan untuk panjang jalur (yaitu, jumlah tingkat CA) tidak lebih dari yang diperlukan untuk mendelegasikan peran administratif dan keamanan. Menambahkan CA ke hierarki berarti meningkatkan jumlah sertifikat di jalur sertifikasi, yang meningkatkan waktu validasi. Menjaga panjang jalur seminimal mungkin juga mengurangi jumlah sertifikat yang dikirim dari server ke klien saat memvalidasi sertifikat entitas akhir.

Secara teori, CA root, yang tidak memiliki pathLenConstraintparameter, dapat mengotorisasi tingkat CA bawahan yang tidak terbatas. CA bawahan dapat memiliki CA bawahan anak sebanyak yang diizinkan oleh konfigurasi internalnya. AWS Private CA hierarki terkelola mendukung jalur sertifikasi CA hingga kedalaman lima tingkat.

Struktur CA yang dirancang dengan baik memiliki beberapa manfaat:

- Kontrol administratif terpisah untuk unit organisasi lain
- Kemampuan untuk mendelegasikan akses ke CA bawahan
- Struktur hierarkis yang melindungi CA tingkat tinggi dengan kontrol keamanan tambahan

Dua struktur CA umum mencakup semua ini:

- Dua level CA: CA akar dan CA bawahan

  Ini adalah struktur CA paling sederhana yang memungkinkan administrasi, kontrol, dan kebijakan keamanan terpisah untuk CA akar dan CA bawahan. Anda dapat mempertahankan kontrol dan kebijakan yang membatasi untuk CA akar Anda sambil mengizinkan akses yang lebih permisif untuk CA bawahan. Yang kedua ini digunakan untuk penerbitan massal sertifikat entitas akhir.

- Tiga level CA: root CA dan dua lapisan CA bawahan

  Mirip dengan yang di atas, struktur ini menambahkan lapisan CA tambahan untuk lebih memisahkan CA akar dari operasi CA tingkat rendah. Lapisan CA tengah hanya digunakan untuk menandatangani CA bawahan yang melakukan penerbitan sertifikat entitas akhir.

Struktur CA yang kurang umum meliputi berikut ini:

- Empat atau lebih level CA

Meskipun kurang umum daripada hierarki tiga tingkat, hierarki CA dengan empat atau lebih tingkat dimungkinkan dan mungkin diperlukan untuk memungkinkan delegasi administratif.

- Satu level CA: root CA saja

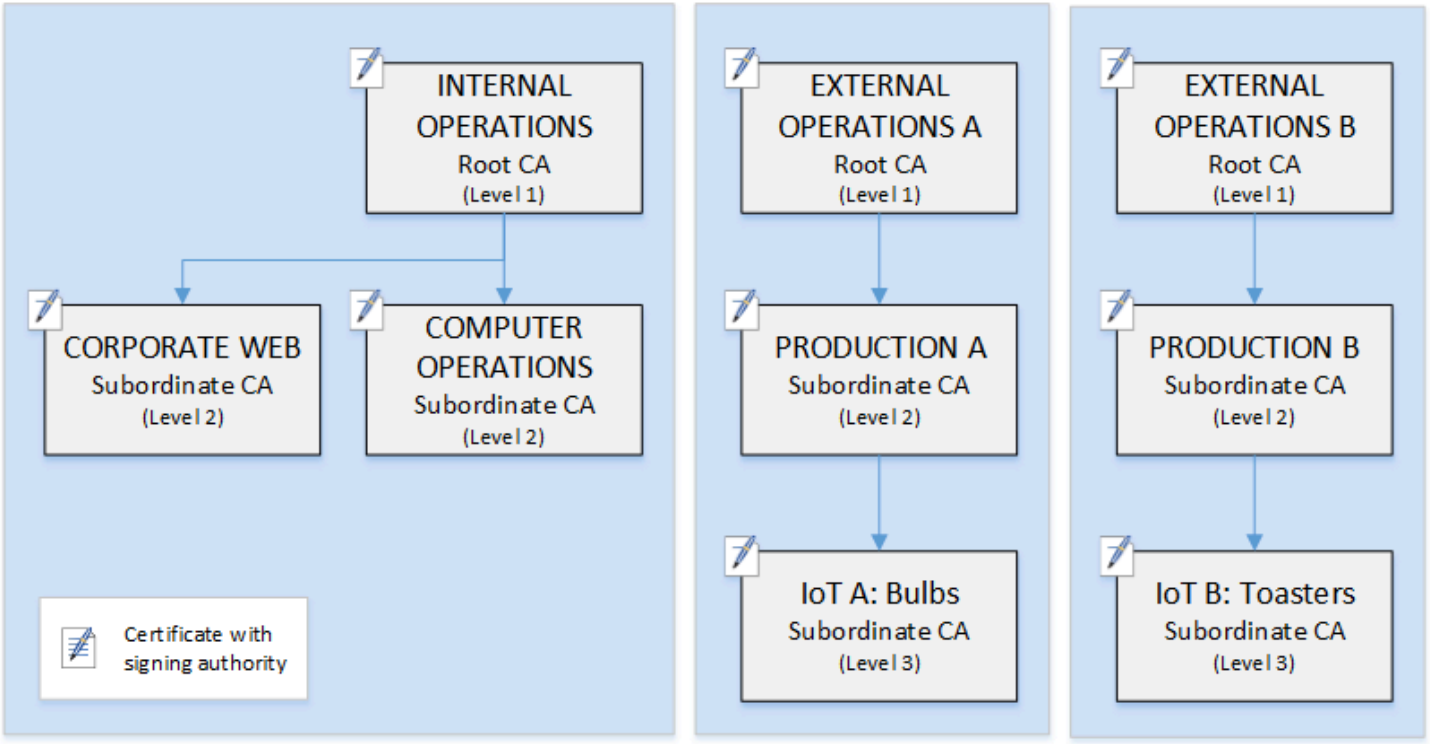
  Struktur ini biasanya digunakan untuk developer dan pengujian ketika rantai kepercayaan penuh tidak diperlukan. Digunakan dalam produksi, itu tidak lazim. Selain itu, ini melanggar praktik terbaik dalam mempertahankan kebijakan keamanan terpisah untuk CA akar dan CA yang menerbitkan sertifikat entitas akhir.

  Namun, jika Anda sudah menerbitkan sertifikat langsung dari CA root, Anda dapat bermigrasi ke. AWS Private CA Melakukannya memberikan keuntungan keamanan dan kontrol dibandingkan menggunakan root CA yang dikelola dengan OpenSSL atau perangkat lunak lain.

Contoh PKI privat untuk produsen

Dalam contoh ini, perusahaan teknologi hipotetis memproduksi dua produk Internet untuk Segala (IoT), bola lampu pintar dan pemanggang roti pintar. Selama produksi, setiap perangkat menerbiktan sertifikat entitas akhir sehingga dapat berkomunikasi dengan aman melalui internet dengan produsen. PKI perusahaan juga mengamankan infrastruktur komputernya, termasuk situs web internal dan berbagai layanan komputer dengan host mandiri yang menjalankan keuangan dan operasi bisnis.

Hasilnya, model hierarki CA hampir mirip dengan aspek administratif dan operasional bisnis ini.



Hierarki ini berisi tiga akar, satu untuk Operasi Internal dan dua untuk Operasi Eksternal (satu CA akar untuk setiap lini produk). Ini juga menggambarkan beberapa panjang jalur sertifikasi, dengan dua tingkat CA untuk Operasi Internal dan tiga tingkat untuk Operasi Eksternal.

Penggunaan CA root terpisah dan lapisan CA bawahan tambahan di sisi Operasi Eksternal adalah keputusan desain yang melayani kebutuhan bisnis dan keamanan. Dengan beberapa pohon CA, PKI tahan terhadap reorganisasi, divestasi, atau akuisisi perusahaan nantinya. Saat terjadi perubahan, seluruh hierarki CA akar dapat bergerak dengan teratur dengan divisi yang diamankannya. Dan dengan dua tingkat CA bawahan, CA akar memiliki tingkat isolasi yang tinggi dari CA tingkat 3 yang bertanggung jawab untuk menandatangani sertifikat secara massal untuk ribuan atau jutaan item yang diproduksi.

Di sisi internal, web perusahaan dan operasi komputer internal melengkapi hierarki dua tingkat. Tingkat ini memungkinkan administrator web dan teknisi operasi untuk mengelola penerbitan sertifikat secara independen untuk domain kerjanya sendiri. Pengelompokan PKI ke dalam domain fungsional yang berbeda adalah praktik terbaik keamanan dan melindungi masing-masing dari kompromi yang mungkin memengaruhi yang lain. Administrator web menerbitkan sertifikat entitas akhir untuk digunakan oleh peramban web di seluruh perusahaan, mengautentikasi dan mengenkripsi komunikasi di situs web internal. Teknisi operasi menerbitkan sertifikat entitas akhir yang mengautentikasi host pusat data dan layanan komputasi satu sama lain. Sistem ini membantu menjaga keamanan data sensitif dengan mengenkripsinya di LAN.

Menetapkan batasan panjang pada jalur sertifikasi

Struktur hierarki CA ditentukan dan ditegakkan oleh ekstensi batasan dasar yang ada dalam setiap sertifikat. Ekstensi menentukan dua kendala:

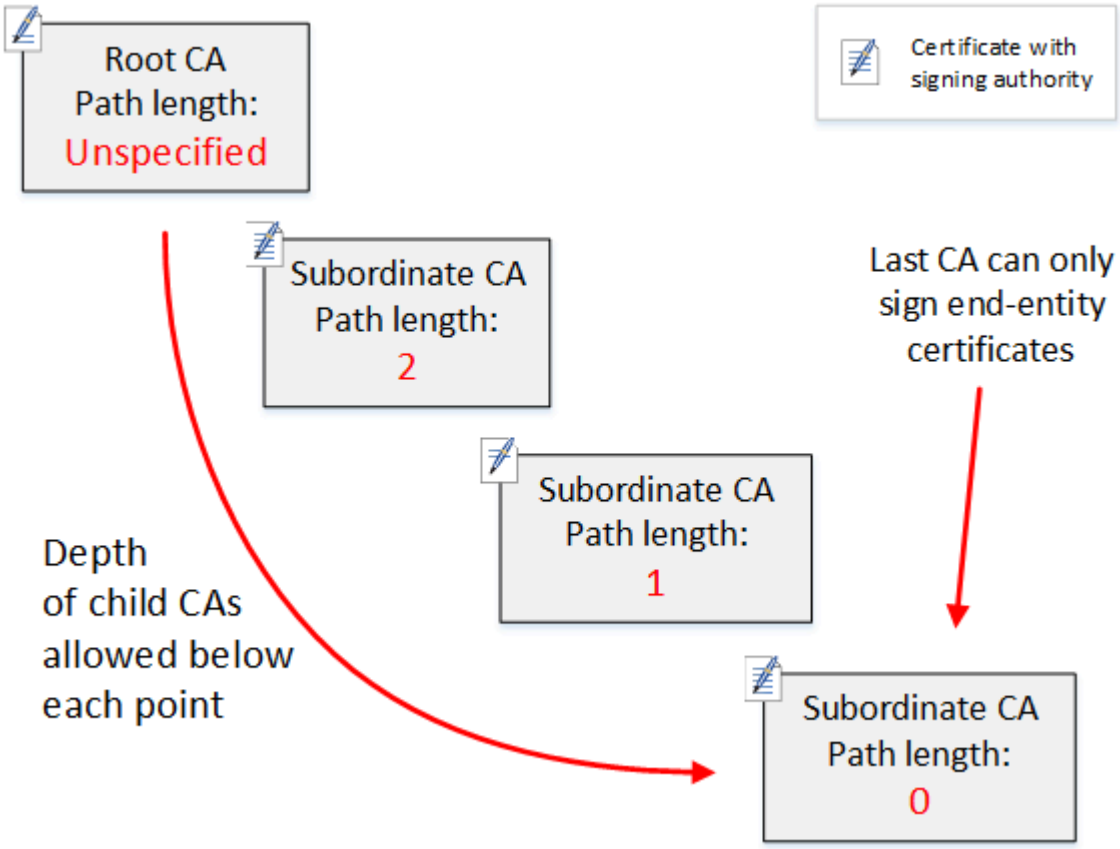
- `cA` – Apakah sertifikat menentukan CA. Jika nilai ini SALAH (default), maka sertifikat tersebut adalah sertifikat entitas akhir.
- `pathLenConstraint`— Jumlah maksimum CA bawahan tingkat rendah yang dapat ada dalam rantai kepercayaan yang valid. Sertifikat entitas akhir tidak dihitung karena bukan sertifikat CA.

Sertifikat CA akar membutuhkan fleksibilitas maksimum dan tidak menyertakan batasan panjang jalur. Ini memungkinkan root untuk menentukan jalur sertifikasi dengan panjang berapa pun.

> **Note**
>
> AWS Private CA membatasi jalur sertifikasi hingga lima tingkat.

CA bawahan memiliki nilai `pathLenConstraint` yang sama dengan atau lebih besar dari nol, bergantung pada lokasi dalam penempatan hierarki dan fitur yang diinginkan. Misalnya, dalam hierarki dengan tiga CA, tidak ada batasan jalur yang ditentukan untuk CA akar. CA bawahan pertama memiliki panjang jalur 1 dan karena itu dapat menandatangani CA turunan. Masing-masing CA turunan ini harus memiliki nilai nol `pathLenConstraint`. Ini berarti bahwa mereka dapat menandatangani sertifikat entitas akhir, tetapi tidak dapat menerbitkan sertifikat CA tambahan. Membatasi kekuatan untuk membuat CA baru adalah kontrol keamanan yang penting.

Diagram berikut mengilustrasikan penyebaran otoritas terbatas ini ke bawah hierarki.



Dalam hierarki empat tingkat ini, akar tidak dibatasi (seperti biasa). Tetapi CA bawahan pertama memiliki nilai 2 `pathLenConstraint`, yang membatasi CA turunannya untuk naik lebih dari dua tingkat. Hasilnya, untuk jalur sertifikasi yang valid, nilai kendala harus dikurangi menjadi nol di dua tingkat berikutnya. Jika peramban web menemukan sertifikat entitas akhir dari cabang ini yang memiliki panjang jalur lebih dari empat, validasi gagal. Sertifikat tersebut dapat merupakan hasil dari CA yang dibuat secara tidak sengaja, CA yang salah konfigurasi, atau penerbitan yang tidak sah.

Mengelola panjang jalur dengan template

AWS Private CA menyediakan template untuk menerbitkan sertifikat root, subordinat, dan entitas akhir. Templat ini merangkum praktik terbaik untuk nilai-nilai batasan dasar, termasuk panjang jalur. Templat meliputi hal-hal berikut:

- RootCACertificate/V1
- Subordinatecacertificate_ 0/V1 PathLen
- Subordinatecacertificate_ 1/V1 PathLen
- Subordinatecacertificate_ 2/V1 PathLen
- Subordinatecacertificate_ 3/V1 PathLen

- EndEntityCertificate/V1

`IssueCertificate` API akan mengembalikan kesalahan jika Anda mencoba membuat CA dengan panjang jalur lebih besar atau sama dengan panjang jalur sertifikat CA yang diterbitkannya.

Untuk informasi selengkapnya tentang templat sertifikat, lihat Memahami templat sertifikat.

Mengotomatiskan penyiapan hierarki CA dengan AWS CloudFormation

Ketika Anda telah menetapkan desain untuk hierarki CA Anda, Anda dapat mengujinya dan memasukkannya ke dalam produksi menggunakan AWS CloudFormation templat. Untuk contoh templat seperti itu, lihat Mendeklarasikan Hirarki CA Privat di Panduan Pengguna AWS CloudFormation .

Mengelola siklus hidup CA privat

Sertifikat CA memiliki masa pakai tetap, atau masa berlaku. Saat sertifikat CA kedaluwarsa, semua sertifikat yang diterbitkan secara langsung atau tidak langsung oleh CA bawahan di bawahnya dalam hierarki CA menjadi tidak valid. Anda dapat menghindari kedaluwarsa sertifikat CA dengan merencanakan terlebih dahulu.

Memilih masa berlaku

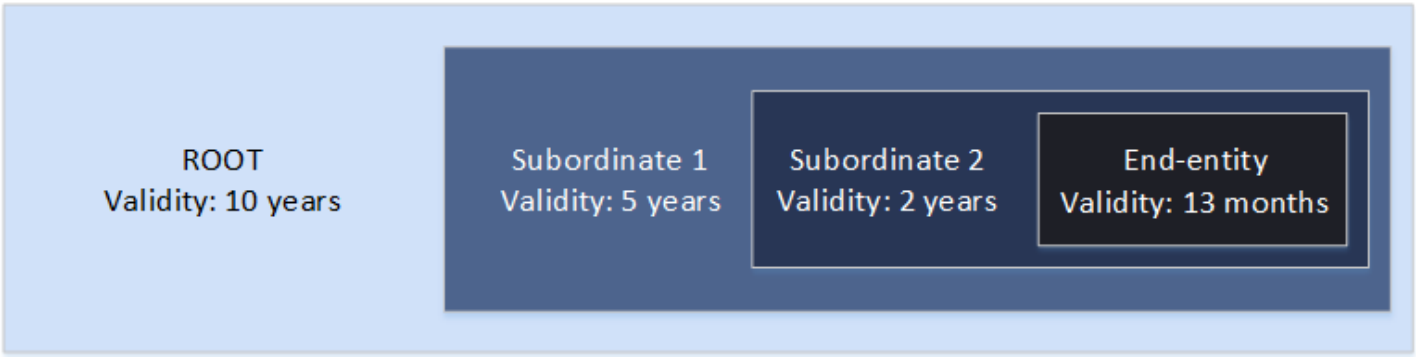
Masa berlaku sertifikat X.509 adalah bidang sertifikat dasar yang diperlukan. Menentukan rentang waktu saat menerbitkan CA yang disertifikasi yang menyatakan bahwa sertifikat dapat dipercaya, pembatasan pencabutan. (Sertifikat akar, yang ditandatangani sendiri, menyatakan masa berlakunya sendiri.)

AWS Private CA dan AWS Certificate Manager membantu konfigurasi periode validitas sertifikat tunduk pada kendala berikut:

- Sertifikat yang dikelola oleh AWS Private CA harus memiliki masa berlaku yang lebih pendek dari atau sama dengan masa berlaku CA yang menerbitkannya. Dengan kata lain, CA turunan dan sertifikat entitas akhir tidak dapat hidup lebih lama dari sertifikat induknya. Mencoba menggunakan `IssueCertificate` API untuk menerbitkan sertifikat CA dengan masa berlaku lebih dari atau sama dengan CA induk gagal.
- Sertifikat yang diterbitkan dan dikelola oleh AWS Certificate Manager (yang ACM menghasilkan kunci pribadi) memiliki masa berlaku 13 bulan (395 hari). ACM mengelola proses perpanjangan

untuk sertifikat ini. Jika Anda menggunakan AWS Private CA untuk menerbitkan sertifikat secara langsung, Anda dapat memilih periode validitas apa pun.

Diagram berikut menunjukkan konfigurasi khas periode validitas nest. Sertifikat akar adalah yang paling berumur panjang; sertifikat entitas akhir relatif berumur pendek; dan CA bawahan berkisar antar ekstrem ini.



Saat Anda merencanakan hierarki CA, tentukan masa pakai yang optimal untuk sertifikat CA Anda. Bekerja mundur dari masa pakai yang diinginkan dari sertifikat entitas akhir yang ingin Anda terbitkan.

Sertifikat entitas akhir

Sertifikat entitas akhir harus memiliki masa berlaku yang sesuai dengan kasus penggunaan. Masa pakai yang singkat meminimalkan paparan sertifikat jika kunci privatnya hilang atau dicuri. Namun, masa pakai yang singkat berarti perpanjangan yang sering. Kegagalan untuk memperpanjang sertifikat yang kedaluwarsa dapat mengakibatkan waktu henti.

Penggunaan sertifikat entitas akhir yang terdistribusi juga dapat menimbulkan masalah logistik jika ada pelanggaran keamanan. Perencanaan Anda harus memperhitungkan perpanjangan dan distribusi sertifikat, pencabutan sertifikat yang disusupi, dan seberapa cepat pencabutan menyebar ke klien yang mengandalkan sertifikat.

Masa berlaku default untuk sertifikat entitas akhir yang diterbitkan melalui ACM adalah 13 bulan (395 hari). Di AWS Private CA, Anda dapat menggunakan `IssueCertificate` API untuk menerapkan periode validitas apa pun asalkan kurang dari CA yang diterbitkan.

Sertifikat CA Bawahan

Sertifikat CA bawahan harus memiliki masa berlaku yang jauh lebih lama daripada sertifikat yang mereka terbitkan. Rentang yang baik untuk validitas sertifikat CA adalah dua hingga lima kali periode

sertifikat CA turunan atau sertifikat entitas akhir yang diterbitkannya. Misalnya, anggap Anda memiliki hierarki CA dua tingkat (CA akar dan satu CA bawahan). Jika Anda ingin menerbitkan sertifikat entitas akhir dengan masa pakai satu tahun, Anda dapat mengkonfigurasi masa pakai CA penerbit bawahan menjadi tiga tahun. Ini adalah periode validitas default untuk sertifikat CA bawahan di. AWS Private CA Sertifikat CA bawahan dapat diubah tanpa mengganti sertifikat CA akar.

Sertifikat Akar

Perubahan sertifikat CA akar mempengaruhi seluruh PKI (infrastruktur kunci publik) dan mengharuskan Anda untuk memperbarui semua tergantung klien sistem operasi dan peramban penyimpanan kepercayaan. Untuk meminimalkan dampak operasional, Anda harus memilih masa berlaku yang panjang untuk sertifikat akar. AWS Private CA Default untuk sertifikat root adalah sepuluh tahun.

Mengelola suksesi CA

Anda memiliki dua cara untuk mengelola suksesi CA: Ganti CA lama, atau terbitkan ulang CA dengan masa berlaku baru.

Mengganti CA lama

Untuk mengganti CAN lama, Anda membuat CA baru dan menghubungkannya ke CA induk yang sama. Setelah itu, Anda menerbitkan sertifikat dari CA baru.

Sertifikat yang diterbitkan dari CA baru memiliki rantai CA baru. Setelah CA baru dibuat, Anda dapat menonaktifkan CA lama untuk mencegahnya menerbitkan sertifikat baru. Meskipun dinonaktifkan, CA lama mendukung pencabutan untuk sertifikat lama yang dikeluarkan dari CA, dan, jika dikonfigurasi untuk melakukannya, ia terus memvalidasi sertifikat melalui OCSP dan/atau daftar pencabutan sertifikat (CRL). Ketika sertifikat terakhir yang diterbitkan dari CA lama kedaluwarsa, Anda dapat menghapus CA lama. Anda dapat membuat laporan audit untuk semua sertifikat yang diterbitkan dari CA untuk mengonfirmasi bahwa semua sertifikat yang diterbitkan telah kedaluwarsa. Jika CA lama memiliki CA bawahan, Anda juga harus menggantinya, karena CA bawahan kedaluwarsa pada saat yang sama atau sebelum CA induknya. Mulailah dengan mengganti CA tertinggi dalam hierarki yang perlu diganti. Kemudian buat CA bawahan pengganti yang baru berikutnya di setiap tingkat yang lebih rendah.

AWS merekomendasikan agar Anda menyertakan pengidentifikasi generasi CA dalam nama CA sesuai kebutuhan. Sebagai contoh, menganggap bahwa Anda nama generasi pertama CA “Corporate Root CA”. Jika Anda membuat CA generasi kedua, beri nama “Corporate Root CA G2.”

Konvensi penamaan sederhana ini dapat membantu menghindari kebingungan ketika kedua CA belum kedaluwarsa.

Metode suksesi CA ini lebih disukai karena memutar kunci privat CA. Memutar kunci privat adalah praktik terbaik untuk kunci CA. Frekuensi rotasi harus proporsional dengan frekuensi penggunaan kunci: CA yang menerbitkan lebih banyak sertifikat harus dirotasi lebih sering.

> **Note**
>
> Sertifikat privat yang dikeluarkan melalui ACM tidak dapat diperpanjang jika Anda mengganti CA. Jika Anda menggunakan ACM untuk penerbitan dan perpanjangan, Anda harus menerbitkan ulang sertifikat CA untuk memperpanjang masa pakai CA.

Menerbitkan kembali CA lama

Ketika CA mendekati kedaluwarsa, metode alternatif untuk memperpanjang umurnya adalah dengan menerbitkan kembali sertifikat CA dengan tanggal kedaluwarsa baru. Penerbitan kembali meninggalkan semua metadata CA di tempatnya dan mempertahankan kunci pribadi dan publik yang ada. Dalam skenario ini, rantai sertifikat yang ada dan sertifikat entitas akhir yang belum kedaluwarsa yang dikeluarkan oleh CA tetap berlaku hingga kedaluwarsa. Penerbitan sertifikat baru juga dapat berlanjut tanpa gangguan. Untuk memperbarui CA dengan sertifikat yang diterbitkan kembali, ikuti prosedur instalasi yang biasa dijelaskan dalamMembuat dan menginstal sertifikat CA.

> **Note**
>
> Kami merekomendasikan untuk mengganti CA yang kedaluwarsa daripada menerbitkan kembali sertifikatnya karena keuntungan keamanan yang diperoleh dengan memutar ke key pair baru.

Mencabut CA

Anda mencabut CA dengan mencabut sertifikat yang mendasarinya. Ini juga secara efektif mencabut semua sertifikat yang dikeluarkan oleh CA. Informasi pencabutan didistribusikan kepada klien melalui OCSP atau CRL. Anda harus mencabut sertifikat CA hanya jika Anda ingin mencabut semua entitas akhir yang diterbitkan dan sertifikat CA bawahan.

Menyiapkan metode pencabutan sertifikat

Saat Anda merencanakan PKI pribadi Anda AWS Private CA, Anda harus mempertimbangkan cara menangani situasi di mana Anda tidak lagi ingin titik akhir mempercayai sertifikat yang dikeluarkan, seperti ketika kunci pribadi dari titik akhir diekspos. Pendekatan umum untuk masalah ini adalah dengan menggunakan sertifikat berumur pendek atau untuk mengkonfigurasi pencabutan sertifikat. Sertifikat berumur pendek kedaluwarsa dalam waktu yang singkat, dalam jam atau hari, pencabutan itu tidak masuk akal, dengan sertifikat menjadi tidak valid dalam waktu yang hampir bersamaan yang diperlukan untuk memberi tahu titik akhir pencabutan. Bagian ini menjelaskan opsi pencabutan untuk AWS Private CA pelanggan, termasuk konfigurasi dan praktik terbaik.

Pelanggan yang mencari metode pencabutan dapat memilih Online Certificate Status Protocol (OCSP), daftar pencabutan sertifikat (CRL), atau keduanya.

> **Note**
>
> Jika Anda membuat CA tanpa mengonfigurasi pencabutan, Anda selalu dapat mengonfigurasinya nanti. Untuk informasi selengkapnya, lihat Memperbarui CA privat Anda.

- Protokol Status Sertifikat Online (OCSP)

  AWS Private CA menyediakan solusi OCSP yang dikelola sepenuhnya untuk memberi tahu titik akhir bahwa sertifikat telah dicabut tanpa perlu pelanggan mengoperasikan infrastruktur sendiri. Pelanggan dapat mengaktifkan OCSP pada CA baru atau yang sudah ada dengan satu operasi menggunakan AWS Private CA konsol, API, CLI, atau melalui. AWS CloudFormation Sedangkan CRL disimpan dan diproses pada titik akhir dan dapat menjadi basi, penyimpanan OCSP dan persyaratan pemrosesan ditangani secara serempak di backend responder.

  Saat Anda mengaktifkan OCSP untuk CA, AWS Private CA sertakan URL responden OCSP dalam ekstensi Authority Information Access (AIA) dari setiap sertifikat baru yang dikeluarkan. Ekstensi ini memungkinkan klien seperti browser web untuk menanyakan responden dan menentukan apakah sertifikat CA entitas akhir atau bawahan dapat dipercaya. Responden mengembalikan pesan status yang ditandatangani secara kriptografi untuk memastikan keasliannya.

  Responden AWS Private CA OCSP sesuai dengan RFC 5019.

  Pertimbangan OCSP

  - Pesan status OCSP ditandatangani menggunakan algoritma penandatanganan yang sama dengan CA penerbit yang dikonfigurasi untuk digunakan. CA yang dibuat di AWS Private CA konsol menggunakan algoritma tanda tangan SHA256WITHRSA secara default. Algoritma lain yang didukung dapat ditemukan di dokumentasi CertificateAuthorityConfigurationAPI.
  - Templat sertifikat ApiPassThrough dan CSRPassThrough tidak akan berfungsi dengan ekstensi AIA jika responden OCSP diaktifkan.
  - Titik akhir dari layanan OCSP yang dikelola dapat diakses di internet publik. Pelanggan yang menginginkan OCSP tetapi memilih untuk tidak memiliki titik akhir publik perlu mengoperasikan infrastruktur OCSP mereka sendiri.
- Daftar Pencabutan Sertifikat (CRL)

  CRL berisi daftar sertifikat yang dicabut. Saat Anda mengonfigurasi CA untuk menghasilkan CRL, AWS Private CA sertakan ekstensi Titik Distribusi CRL di setiap sertifikat baru yang dikeluarkan. Ekstensi ini menyediakan URL untuk CRL. Ekstensi ini memungkinkan klien seperti browser web untuk menanyakan CRL dan menentukan apakah sertifikat CA entitas akhir atau bawahan dapat dipercaya.

Karena klien harus mengunduh CRL dan memprosesnya secara lokal, penggunaannya lebih intensif memori daripada OCSP. CRL dapat mengkonsumsi lebih sedikit bandwidth jaringan karena daftar CRL diunduh dan di-cache, dibandingkan dengan OCSP yang memeriksa status pencabutan untuk setiap upaya koneksi baru.

> **Note**
>
> Baik OCSP dan CRL menunjukkan beberapa penundaan antara pencabutan dan ketersediaan perubahan status.
>
> - Tanggapan OCSP dapat memakan waktu hingga 60 menit untuk mencerminkan status baru saat Anda mencabut sertifikat. Secara umum, OCSP cenderung mendukung distribusi informasi pencabutan yang lebih cepat karena, tidak seperti CRL yang dapat di-cache oleh klien selama berhari-hari, respons OCSP biasanya tidak di-cache oleh klien.
> - CRL biasanya diperbarui sekitar 30 menit setelah sertifikat dicabut. Jika karena alasan apa pun pembaruan CRL gagal, lakukan AWS Private CA upaya lebih lanjut setiap 15 menit.

Persyaratan umum untuk konfigurasi pencabutan

Persyaratan berikut berlaku untuk semua konfigurasi pencabutan.

- Konfigurasi yang menonaktifkan CRL atau OCSP harus berisi hanya `Enabled=False` parameter, dan akan gagal jika parameter lain seperti `CustomCname` atau disertakan. `ExpirationInDays`
- Dalam konfigurasi CRL, `S3BucketName` parameter harus sesuai dengan aturan penamaan bucket Amazon Simple Storage Service.
- Konfigurasi yang berisi parameter Nama Canonical kustom (CNAME) untuk CRL atau OCSP harus sesuai dengan pembatasan RFC7230 pada penggunaan karakter khusus dalam CNAME.
- Dalam konfigurasi CRL atau OCSP, nilai parameter CNAME tidak boleh menyertakan awalan protokol seperti “http://” atau “https://”.

Topik

- Merencanakan daftar pencabutan sertifikat (CRL)
- Mengkonfigurasi URL Kustom untuk AWS Private CA OCSP

Merencanakan daftar pencabutan sertifikat (CRL)

Sebelum Anda dapat mengonfigurasi CRL sebagai bagian dari proses pembuatan CA, beberapa pengaturan sebelumnya mungkin diperlukan. Bagian ini menjelaskan prasyarat dan opsi yang harus Anda pahami sebelum membuat CA dengan CRL terlampir.

Untuk informasi tentang menggunakan Online Certificate Status Protocol (OCSP) sebagai alternatif atau suplemen CRL, lihat Opsi pencabutan sertifikat dan. Mengkonfigurasi URL Kustom untuk AWS Private CA OCSP

Topik

- Struktur CRL
- Kebijakan akses untuk CRL di Amazon S3
- Mengaktifkan S3 Block Public Access (BPA) dengan CloudFront
- Mengenkripsi CRL Anda
- Menentukan URI Titik Distribusi CRL (CDP)

Struktur CRL

Setiap CRL adalah file DER yang dikodekan. Untuk mengunduh file dan menggunakan OpenSSL untuk melihatnya, gunakan perintah yang mirip dengan berikut ini:

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

CRL memiliki format sebagai berikut:

```
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**
>
> CRL hanya akan disimpan di Amazon S3 setelah sertifikat dikeluarkan yang merujuknya. Sebelum itu, hanya akan ada file `acm-pca-permission-test-key` yang terlihat di bucket Amazon S3.

Kebijakan akses untuk CRL di Amazon S3

Jika Anda berencana membuat CRL, Anda perlu menyiapkan ember Amazon S3 untuk menyimpannya. AWS Private CA secara otomatis menyetor CRL di bucket Amazon S3 yang Anda tunjuk dan memperbaruinya secara berkala. Untuk informasi selengkapnya, lihat Membuat bucket.

Bucket S3 Anda harus diamankan dengan kebijakan izin IAM terlampir. Pengguna resmi dan kepala layanan memerlukan `Put` izin AWS Private CA untuk mengizinkan menempatkan objek di ember, dan `Get` izin untuk mengambilnya. Selama prosedur konsol untuk membuat CA, Anda dapat memilih untuk mengizinkan AWS Private CA membuat bucket baru dan menerapkan kebijakan izin default.

> **Note**
>
> Konfigurasi kebijakan IAM tergantung pada yang Wilayah AWS terlibat. Wilayah terbagi dalam dua kategori:
>
> - Default-enabled Regions — Wilayah yang diaktifkan secara default untuk semua. Akun AWS
> - Wilayah yang dinonaktifkan default — Wilayah yang dinonaktifkan secara default, tetapi dapat diaktifkan secara manual oleh pelanggan.
>
> Untuk informasi selengkapnya dan daftar Wilayah yang dinonaktifkan default, lihat Mengelola. Wilayah AWS Untuk diskusi tentang prinsip-prinsip layanan dalam konteks IAM, lihat prinsip AWS layanan di Wilayah opt-in.

Saat Anda mengonfigurasi CRL sebagai metode pencabutan sertifikat, AWS Private CA buat CRL dan publikasikan ke bucket S3. Bucket S3 memerlukan kebijakan IAM yang memungkinkan kepala AWS Private CA layanan untuk menulis ke bucket. Nama kepala layanan bervariasi sesuai dengan Wilayah yang digunakan, dan tidak semua kemungkinan didukung.

PCA	S3	Pemimpin layanan
Keduanya di wilayah yang sama		`acm-pca.amazonaws.com`
Diaktifkan	Diaktifkan	`acm-pca.amazonaws.com`
Dinonaktifkan	Diaktifkan	`acm-pca.Region.amazonaws.com`
Diaktifkan	Dinonaktifkan	Tidak didukung

Kebijakan default tidak berlaku `SourceArn` pembatasan pada CA. Kami menyarankan Anda secara manual menerapkan kebijakan yang kurang permisif yang ditunjukkan di bawah ini, yang membatasi akses ke AWS akun tertentu dan CA pribadi tertentu. Untuk informasi selengkapnya, lihat Menambahkan kebijakan bucket menggunakan konsol Amazon S3.

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

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

Jika Anda memilih untuk mengizinkan kebijakan default, Anda selalu dapat memodifikasinya nanti.

Mengaktifkan S3 Block Public Access (BPA) dengan CloudFront

Bucket Amazon S3 yang baru dikonfigurasi secara default dengan fitur Blokir Akses Publik (BPA) yang aktif. Termasuk dalam praktik terbaik keamanan Amazon S3, BPA adalah seperangkat kontrol akses yang dapat digunakan pelanggan untuk menyempurnakan akses ke objek di bucket S3 mereka dan ke ember secara keseluruhan. Ketika BPA aktif dan dikonfigurasi dengan benar, hanya AWS pengguna yang berwenang dan diautentikasi yang memiliki akses ke ember dan isinya.

AWS merekomendasikan penggunaan BPA pada semua bucket S3 untuk menghindari paparan informasi sensitif terhadap musuh potensial. Namun, perencanaan tambahan diperlukan jika klien PKI Anda mengambil CRL di internet publik (yaitu, saat tidak masuk ke akun AWS). Bagian ini menjelaskan cara mengonfigurasi solusi PKI pribadi menggunakan Amazon CloudFront, jaringan pengiriman konten (CDN), untuk melayani CRL tanpa memerlukan akses klien yang diautentikasi ke bucket S3.

> **Note**
>
> Menggunakan CloudFront menimbulkan biaya tambahan pada akun Anda AWS . Untuk informasi selengkapnya, lihat CloudFront Harga Amazon.
> Jika Anda memilih untuk menyimpan CRL Anda di bucket S3 dengan BPA diaktifkan, dan Anda tidak menggunakannya CloudFront, Anda harus membangun solusi CDN lain untuk memastikan bahwa klien PKI Anda memiliki akses ke CRL Anda.

Siapkan Amazon S3 dengan BPA

Dalam S3, membuat bucket baru untuk CRL Anda, seperti biasa, lalu aktifkan BPA di sana.

Untuk mengonfigurasi bucket Amazon S3 yang memblokir akses publik ke CRL Anda

1. Buat bucket S3 baru menggunakan prosedur di Membuat bucket. Selama prosedur, pilih opsi Blokir semua akses publik.

   Untuk informasi lebih lanjut, lihat Memblokir akses publik ke penyimpanan Amazon S3 Anda.
2. Saat bucket telah dibuat, pilih namanya dari daftar, navigasikan ke tab Izin, pilih Edit di bagian Kepemilikan objek, dan pilih Pemilik bucket yang disukai.
3. Juga pada tab Izin, tambahkan kebijakan IAM ke bucket seperti yang dijelaskan dalam. Kebijakan akses untuk CRL di Amazon S3

Siapkan CloudFront untuk BPA

Buat CloudFront distribusi yang akan memiliki akses ke bucket S3 pribadi Anda, dan dapat melayani CRL ke klien yang tidak diautentikasi.

Untuk mengkonfigurasi CloudFront distribusi untuk CRL

1. Buat CloudFront distribusi baru menggunakan prosedur dalam Membuat Distribusi di Panduan CloudFront Pengembang Amazon.

   Saat menyelesaikan prosedur, terapkan pengaturan berikut:

   - Di Nama Domain Asal, pilih bucket S3 Anda.
   - Pilih Ya untuk Batasi Akses Bucket.
   - Pilih Buat Identitas Baru untuk Identitas Akses Asal.
   - Pilih Ya, Perbarui Kebijakan Bucket di bawah Berikan Izin Baca pada Bucket.

   > **Note**
   >
   > Dalam prosedur ini, CloudFront ubah kebijakan bucket Anda agar dapat mengakses objek bucket. Pertimbangkan mengedit kebijakan ini untuk hanya mengizinkan akses ke objek di bawah folder `crl`.

2. Setelah distribusi diinisialisasi, cari nama domainnya di CloudFront konsol dan simpan untuk prosedur selanjutnya.

> **Note**
>
> Jika bucket S3 Anda baru dibuat di Wilayah selain us-east-1, Anda mungkin mendapatkan kesalahan pengalihan sementara HTTP 307 saat mengakses aplikasi yang dipublikasikan melalui. CloudFront Mungkin perlu beberapa jam agar alamat ember menyebar.

Siapkan CA Anda untuk BPA

Saat mengonfigurasi CA baru Anda, sertakan alias ke distribusi Anda CloudFront.

Untuk mengonfigurasi CA Anda dengan CNAME untuk CloudFront

- Buat CA Anda menggunakan Prosedur untuk membuat CA (CLI) .

  Saat Anda melakukan prosedur, file pencabutan `revoke_config.txt` harus menyertakan baris berikut untuk menentukan objek CRL non-publik dan untuk memberikan URL ke titik akhir distribusi di: CloudFront

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

  Setelah itu, ketika Anda mengeluarkan sertifikat dengan CA ini, sertifikat tersebut akan berisi blok seperti berikut:

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

> **Note**
>
> Jika Anda memiliki sertifikat yang lebih lama yang diterbitkan oleh CA ini, sertifikat tersebut akan dapat mengakses CRL.

Mengenkripsi CRL Anda

Anda dapat mengonfigurasi enkripsi secara opsional pada bucket Amazon S3 yang berisi CRL Anda. AWS Private CA mendukung dua mode enkripsi untuk aset di Amazon S3:

- Enkripsi sisi server otomatis dengan kunci AES-256 terkelola Amazon S3.
- Enkripsi terkelola pelanggan menggunakan AWS Key Management Service dan AWS KMS key dikonfigurasi dengan spesifikasi Anda.

> **Note**
>
> AWS Private CA tidak mendukung penggunaan kunci KMS default yang dihasilkan secara otomatis oleh S3.

Prosedur berikut menjelaskan cara menyiapkan setiap opsi enkripsi.

Untuk mengkonfigurasi enkripsi otomatis

Selesaikan langkah-langkah berikut untuk mengaktifkan enkripsi sisi server S3.

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Di tabel Bucket, pilih ember yang akan menampung AWS Private CA aset Anda.
3. Pada halaman untuk bucket Anda, pilih tab Properti.
4. Pilih kartu Enkripsi default.
5. Pilih Aktifkan.
6. Pilih Kunci Amazon S3 (SSE-S3).
7. Pilih Simpan Perubahan.

Untuk mengkonfigurasi enkripsi kustom

Selesaikan langkah-langkah berikut untuk mengaktifkan enkripsi menggunakan kunci kustom.

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Di tabel Bucket, pilih ember yang akan menampung AWS Private CA aset Anda.
3. Pada halaman untuk bucket Anda, pilih tab Properti.
4. Pilih kartu Enkripsi default.

5. Pilih Aktifkan.
6. Pilih AWS Key Management Service kunci (SSE-KMS).
7. Pilih salah satu Pilih dari AWS KMS kunci Anda atau Masukkan AWS KMS key ARN.
8. Pilih Simpan Perubahan.
9. (Opsional) Jika Anda belum memiliki kunci KMS, buat satu menggunakan perintah AWS CLI create-key berikut:

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

Outputnya berisi ID kunci dan Nama Sumber Daya Amazon (ARN) dari kunci KMS. Berikut ini adalah contoh output:

```
{
    "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. Dengan menggunakan langkah-langkah berikut, Anda memberikan izin kepada kepala AWS Private CA layanan untuk menggunakan kunci KMS. Secara default, semua kunci KMS bersifat pribadi; hanya pemilik sumber daya yang dapat menggunakan kunci KMS untuk mengenkripsi dan mendekripsi data. Namun, pemilik sumber daya dapat memberikan izin untuk mengakses kunci KMS ke pengguna dan sumber daya lain. Prinsipal layanan harus berada di Wilayah yang sama dengan tempat kunci KMS disimpan.

    a. Pertama, simpan kebijakan default untuk kunci KMS Anda seperti `policy.json` menggunakan get-key-policyperintah berikut:

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

b. Buka file `policy.json` di editor teks. Pilih salah satu pernyataan kebijakan berikut dan tambahkan ke kebijakan yang ada.

Jika kunci bucket Amazon S3 diaktifkan, gunakan pernyataan berikut:

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

Jika kunci bucket Amazon S3 dinonaktifkan, gunakan pernyataan berikut:

```
{
   "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. Terakhir, terapkan kebijakan yang diperbarui menggunakan put-key-policyperintah berikut:

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

Menentukan URI Titik Distribusi CRL (CDP)

Jika Anda menggunakan bucket S3 sebagai CDP untuk CA Anda, URI CDP dapat berada dalam salah satu format berikut.

- `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`

Jika Anda telah mengonfigurasi CA Anda dengan CNAME kustom, URI CDP akan menyertakan CNAME, misalnya, `http://alternative.example.com/crl/CA-ID.crl`

Mengkonfigurasi URL Kustom untuk AWS Private CA OCSP

> **Note**
>
> Topik ini ditujukan untuk pelanggan yang ingin menyesuaikan URL publik dari endpoint responder OCSP untuk branding atau tujuan lain. Jika Anda berencana untuk menggunakan konfigurasi default OCSP AWS Private CA terkelola, Anda dapat melewati topik ini dan mengikuti petunjuk konfigurasi di Configure revocation.

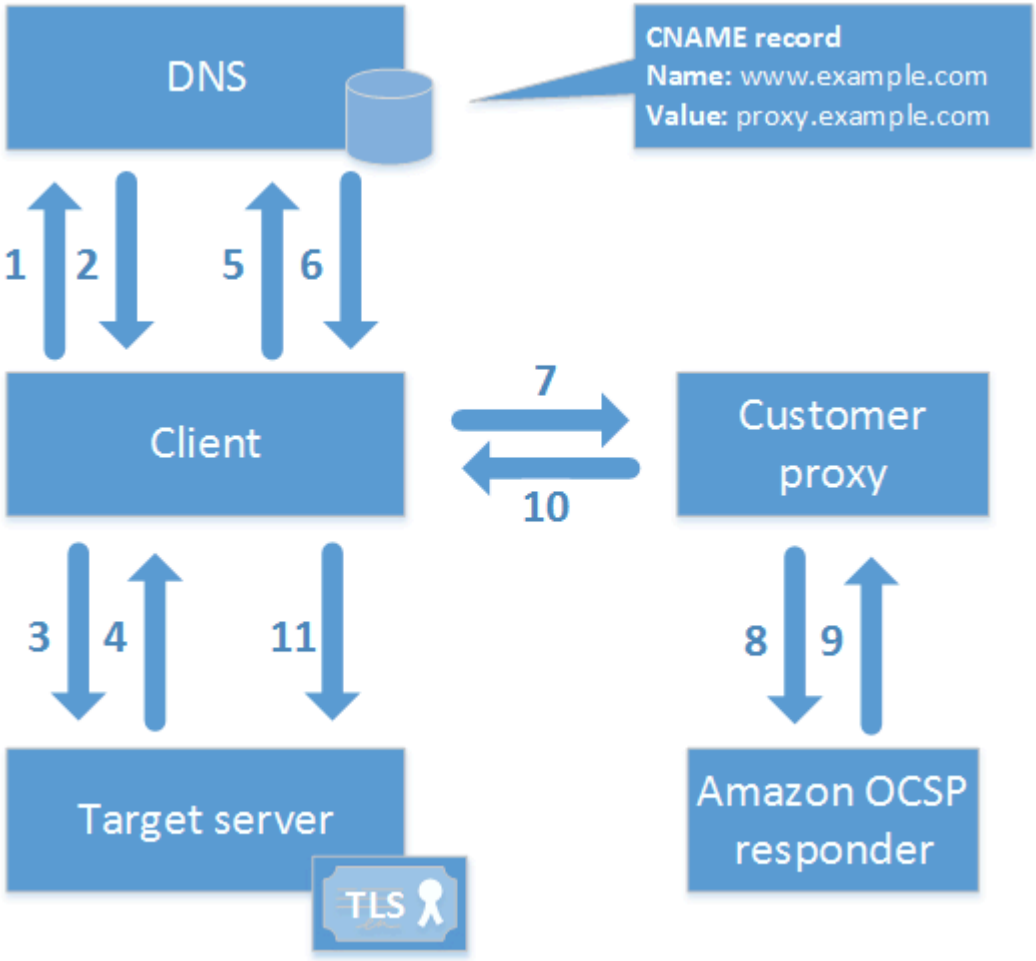
Secara default, saat Anda mengaktifkan OCSP AWS Private CA, setiap sertifikat yang Anda terbitkan berisi URL untuk responden AWS OCSP. Hal ini memungkinkan klien meminta koneksi kriptografis aman untuk mengirim kueri validasi OCSP langsung ke. AWS Namun, dalam beberapa kasus, mungkin lebih baik untuk menyatakan URL yang berbeda di sertifikat Anda sambil tetap mengirimkan kueri OCSP ke. AWS

> **Note**
>
> Untuk informasi tentang menggunakan daftar pencabutan sertifikat (CRL) sebagai alternatif atau tambahan untuk OCSP, lihat Mengkonfigurasi pencabutan dan Merencanakan daftar pencabutan sertifikat (CRL).

Tiga elemen terlibat dalam mengkonfigurasi URL kustom untuk OCSP.

- Konfigurasi CA - Tentukan URL OCSP kustom di `RevocationConfiguration` CA Anda seperti yang dijelaskan Contoh 2: Buat CA dengan OCSP dan CNAME kustom diaktifkan dalamProsedur untuk membuat CA (CLI) .
- DNS — Tambahkan catatan CNAME ke konfigurasi domain Anda untuk memetakan URL yang muncul di sertifikat ke URL server proxy. Untuk informasi selengkapnya, lihat Contoh 2: Buat CA dengan OCSP dan CNAME kustom diaktifkan di Prosedur untuk membuat CA (CLI) .
- Forwarding proxy server - Siapkan server proxy yang dapat secara transparan meneruskan lalu lintas OCSP yang diterimanya ke responder OCSP. AWS

Diagram berikut menggambarkan bagaimana elemen-elemen ini bekerja sama.



Seperti yang ditunjukkan pada diagram, proses validasi OCSP yang disesuaikan melibatkan langkah-langkah berikut:

1. Klien menanyakan DNS untuk domain target.
2. Klien menerima IP target.
3. Klien membuka koneksi TCP dengan target.
4. Klien menerima sertifikat TLS target.
5. Klien menanyakan DNS untuk domain OCSP yang tercantum dalam sertifikat.
6. Klien menerima IP proxy.
7. Klien mengirimkan kueri OCSP ke proxy.
8. Proxy meneruskan kueri ke responder OCSP.
9. Responder mengembalikan status sertifikat ke proxy.
10. Proxy meneruskan status sertifikat ke klien.

11. Jika sertifikat valid, klien memulai jabat tangan TLS.

> **Tip**
>
> Contoh ini dapat diimplementasikan menggunakan Amazon CloudFront dan Amazon Route 53 setelah Anda mengonfigurasi CA seperti yang dijelaskan di atas.
>
> 1. Di CloudFront, buat distribusi dan konfigurasikan sebagai berikut:
>   - Buat nama alternatif yang cocok dengan CNAME kustom Anda.
>   - Ikat sertifikat Anda untuk itu.
>   - Setel ocsp.acm-pca. <region>.amazonaws.com sebagai asal.
>   - Terapkan `Managed-CachingDisabled` kebijakan.
>   - Tetapkan kebijakan protokol Viewer ke HTTP dan HTTPS.
>   - Atur metode HTTP yang Diizinkan untuk GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE.
> 2. Di Route 53, buat catatan DNS yang memetakan CNAME kustom Anda ke URL distribusi. CloudFront

Mode otoritas sertifikat

AWS Private CA mendukung pembuatan CA di salah satu dari dua mode. Mode, GENERAL_PURPOSE dan SHORT_LIVED_CERTIFICATE, mempengaruhi masa berlaku yang diizinkan dari sertifikat yang dikeluarkan oleh CA.

> **Note**
>
> AWS Private CA tidak melakukan pemeriksaan validitas pada sertifikat CA root.

GENERAL_PURPOSE (default)

Mode ini memungkinkan CA untuk menerbitkan sertifikat dengan periode validitas apa pun. Sebagian besar aplikasi menggunakan sertifikat jenis ini. Biasanya, CA juga menentukan mekanisme pencabutan.

SHORT_LIVED_CERTIFICATE

Mode ini mendefinisikan CA yang secara eksklusif mengeluarkan sertifikat dengan masa berlaku maksimum tujuh hari. Sertifikat berumur pendek ini kedaluwarsa begitu cepat sehingga dapat digunakan tanpa mekanisme pencabutan. Untuk beberapa aplikasi, lebih masuk akal untuk sering menggunakan sertifikat berumur pendek daripada mengeluarkan jaringan dan memproses overhead pencabutan.

CA dengan mode SHORT_LIVED_CERTIFICATE harganya lebih murah dari CA tujuan umum. Untuk informasi lebih lanjut, lihat AWS Private Certificate Authority Harga.

Untuk membuat CA yang mengeluarkan sertifikat berumur pendek, setel `UsageMode` parameter ke SHORT_LIVED_CERTIFICATE menggunakan prosedur untuk membuat CA. AWS CLI

> **Note**
>
> AWS Certificate Manager tidak dapat menerbitkan sertifikat yang ditandatangani oleh CA pribadi dengan mode berumur pendek.

Penggunaan sertifikat berumur pendek didukung oleh AWS layanan berikut:

- Amazon AppStream
- Amazon WorkSpaces

Perencanaan ketahanan

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Availability Zone, Anda dapat mendesain dan mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

Redundansi dan pemulihan bencana

Pertimbangkan redundansi dan DR saat merencanakan hierarki CA Anda. AWS Private CA tersedia di beberapa Wilayah, yang memungkinkan Anda membuat CA redundan di beberapa Wilayah. AWS Private CA Layanan ini beroperasi dengan perjanjian tingkat layanan (SLA) dengan ketersediaan 99,9%. Setidaknya ada dua pendekatan yang dapat Anda pertimbangkan untuk redundansi dan pemulihan bencana. Anda dapat mengonfigurasi redundansi di CA akar atau di CA bawahan tertinggi. Setiap pendekatan memiliki pro dan kontra.

1. Anda dapat membuat dua CA root di dua AWS Wilayah berbeda untuk redundansi dan pemulihan bencana. Dengan konfigurasi ini, setiap root CA beroperasi secara independen di suatu AWS Wilayah, melindungi Anda jika terjadi bencana Single-region. Namun, membuat CA akar yang berlebihan memang meningkatkan kompleksitas operasional: Anda perlu mendistribusikan kedua sertifikat CA akar ke penyimpanan tepercaya peramban dan sistem operasi di lingkungan Anda.
2. Anda juga dapat membuat CA bawahan yang berlebihan untuk diterapkan di setiap AWS Wilayah Anda, dan mengikatnya ke CA root unik yang sama di satu Wilayah. AWS Manfaat dari pendekatan ini adalah Anda hanya perlu mendistribusikan satu sertifikat CA akar ke penyimpanan kepercayaan di lingkungan Anda. Batasannya adalah Anda tidak memiliki akar CA yang berlebihan jika terjadi bencana yang memengaruhi AWS Wilayah tempat CA root Anda berada.

AWS Private CA praktik terbaik

Praktik terbaik adalah rekomendasi yang dapat membantu Anda menggunakannya AWS Private CA secara efektif. Praktik terbaik berikut didasarkan pada pengalaman dunia nyata dari saat ini AWS Certificate Manager dan AWS Private CA pelanggan.

Mendokumentasikan struktur dan kebijakan CA

AWS merekomendasikan untuk mendokumentasikan semua kebijakan dan praktik Anda untuk mengoperasikan CA Anda. Ini mungkin termasuk:

- Penalaran untuk keputusan Anda tentang struktur CA
- Diagram yang menunjukkan CA Anda dan hubungannya
- Kebijakan tentang masa validasi CA
- Merencanakan suksesi CA
- Kebijakan pada panjang jalur
- Izin katalog
- Deskripsi struktur kontrol administratif
- Keamanan

Anda dapat menangkap informasi ini dalam dua dokumen, yang dikenal sebagai Kebijakan Sertifikasi (CP) dan Pernyataan Praktik Sertifikasi (CPS). Baca RFC 3647 untuk kerangka kerja untuk mendapatkan informasi penting tentang operasi CA Anda.

Minimalkan penggunaan CA akar jika memungkinkan

CA akar secara umum hanya boleh digunakan untuk menerbitkan sertifikat untuk CA perantara. Hal ini memungkinkan CA akar disimpan dari bahaya sementara CA perantara melakukan tugas harian menerbitkan sertifikat entitas akhir.

Namun, jika praktik organisasi Anda saat ini adalah menerbitkan sertifikat entitas akhir langsung dari root CA, AWS Private CA dapat mendukung alur kerja ini sekaligus meningkatkan keamanan dan kontrol operasional. Menerbitkan sertifikat entitas akhir dalam skenario ini memerlukan kebijakan izin IAM yang mengizinkan CA akar Anda untuk menggunakan templat sertifikat entitas akhir. Untuk

informasi selengkapnya tentang kebijakan IAM, lihat Identity and Access Management (IAM) untuk AWS Private Certificate Authority.

> **Note**
>
> Konfigurasi ini memberlakukan batasan yang dapat mengakibatkan tantangan operasional. Misalnya, jika CA akar Anda disusupi atau hilang, Anda harus membuat CA akar baru dan mendistribusikannya ke semua klien di lingkungan Anda. Sampai proses pemulihan ini selesai, Anda tidak akan dapat menerbitkan sertifikat baru. Penerbitan sertifikat langsung dari CA akar juga mencegah Anda membatasi akses dan membatasi jumlah sertifikat yang dikeluarkan dari akar Anda, yang keduanya dianggap sebagai praktik terbaik untuk mengelola CA akar.

Berikan root CA miliknya sendiri Akun AWS

Membuat CA root dan CA bawahan dalam dua AWS akun berbeda adalah praktik terbaik yang direkomendasikan. Melakukannya dapat memberi Anda perlindungan tambahan dan kontrol akses untuk CA akar Anda. Anda dapat melakukannya dengan mengekspor CSR dari CA bawahan di satu akun, dan menandatanganinya dengan CA akar di akun lain. Manfaat dari pendekatan ini adalah Anda dapat memisahkan kontrol CA Anda berdasarkan akun. Kerugiannya adalah Anda tidak dapat menggunakan AWS Management Console wizard untuk menyederhanakan proses penandatanganan sertifikat CA CA CA bawahan dari CA root Anda.

> **Important**
>
> Kami sangat menyarankan penggunaan otentikasi multi-faktor (MFA) setiap kali Anda mengakses. AWS Private CA

Peran administrator dan penerbit terpisah

Peran administrator CA harus terpisah dari pengguna yang hanya perlu menerbitkan sertifikat entitas akhir. Jika administrator CA dan penerbit sertifikat berada di tempat yang samaAkun AWS, Anda dapat membatasi izin penerbit dengan membuat pengguna IAM khusus untuk tujuan tersebut.

Melaksanakan pencabutan sertifikat yang dikelola

Pencabutan terkelola secara otomatis memberikan pemberitahuan kepada klien sertifikat ketika sertifikat telah dicabut. Anda mungkin perlu mencabut sertifikat jika informasi kriptografinya telah dikompromikan atau jika dikeluarkan karena kesalahan. Klien biasanya menolak untuk menerima sertifikat yang dicabut. AWS Private CAmenawarkan dua opsi standar untuk pencabutan terkelola: Protokol Status Sertifikat Online (OCSP), dan daftar pencabutan sertifikat (CRL). Untuk informasi selengkapnya, lihat Menyiapkan metode pencabutan sertifikat.

Mengaktifkan AWS CloudTrail

Aktifkan CloudTrail pencatatan sebelum Anda membuat dan mulai mengoperasikan CA pribadi. Dengan CloudTrail, Anda dapat mengambil riwayat panggilan AWS API untuk akun Anda untuk memantau AWS penerapan Anda. Riwayat ini mencakup panggilan API yang dibuat dari AWS Management Console, AWSSDK, AWS Command Line Interface, dan layanan AWS tingkat yang lebih tinggi. Anda juga dapat mengidentifikasi pengguna dan akun mana yang disebut operasi API PCA, alamat IP sumber asal panggilan, dan kapan panggilan terjadi. Anda dapat mengintegrasikan CloudTrail ke dalam aplikasi menggunakan API, mengotomatiskan pembuatan jejak untuk organisasi Anda, memeriksa status jejak Anda, dan mengontrol cara administrator mengaktifkan dan menonaktifkan CloudTrail log. Untuk informasi lebih lanjut, lihat Membuat Jejak. Pergi ke Menggunakan CloudTrail untuk melihat contoh jejak untuk AWS Private CA operasi.

Memutar kunci privat CA

Ini adalah praktik terbaik untuk memperbarui kunci privat untuk CA privat Anda secara berkala. Anda dapat memperbarui kunci dengan mengimpor sertifikat CA baru, atau Anda dapat mengganti CA privat dengan CA baru.

> **Note**
>
> Jika Anda mengganti CA itu sendiri, ketahuilah bahwa ARN CA berubah. Ini akan menyebabkan otomatisasi yang mengandalkan ARN hard-code gagal.

Hapus CA yang tidak digunakan

Anda dapat menghapus CA privat secara permanen. Anda mungkin ingin melakukannya jika Anda tidak lagi membutuhkan CA atau jika Anda ingin menggantinya dengan CA yang memiliki kunci privat yang lebih baru. Untuk menghapus CA dengan aman, kami sarankan Anda mengikuti proses yang diuraikan dalam Menghapus CA privat Anda.

> **Note**
>
> AWS menagihkan Anda untuk CA sampai itu dihapus.

Blokir akses publik ke CRL Anda

AWS Private CAmerekomendasikan penggunaan fitur Amazon S3 Block Public Access (BPA) pada bucket yang berisi CRL. Ini menghindari pengungkapan detail PKI pribadi Anda yang tidak perlu kepada musuh potensial. BPA adalah praktik terbaik S3 dan diaktifkan secara default pada bucket baru. Pengaturan tambahan diperlukan dalam beberapa kasus. Untuk informasi selengkapnya, lihat Mengaktifkan S3 Block Public Access (BPA) dengan CloudFront.

Praktik terbaik aplikasi Amazon EKS

Saat menggunakan AWS Private CA untuk menyediakan Amazon EKS dengan sertifikat X.509, ikuti rekomendasi untuk mengamankan lingkungan multi-penyewa di Panduan Praktik Terbaik Amazon EKS. Untuk informasi umum tentang integrasi AWS Private CA dengan Kubernetes, lihat. Mengamankan Kubernetes dengan AWS Private CA

Administrasi CA swasta

Dengan menggunakan AWS Private CA, Anda dapat membuat hierarki root dan subordinate certificate authority (CA) yang sepenuhnya AWS dihosting untuk penggunaan internal oleh organisasi Anda. Untuk mengelola pencabutan sertifikat, Anda dapat mengaktifkan Online Certificate Status Protocol (OCSP), daftar pencabutan sertifikat (CRL), atau keduanya. AWS Private CA menyimpan dan mengelola sertifikat CA, CRL, dan respons OCSP Anda, dan kunci pribadi untuk otoritas root Anda disimpan dengan aman. AWS

> **Note**
>
> Implementasi OCSP di AWS Private CA tidak mendukung ekstensi permintaan OCSP. Jika Anda mengirimkan kueri batch OCSP yang berisi beberapa sertifikat, responder AWS OCSP hanya memproses sertifikat pertama dalam antrian dan menjatuhkan yang lain. Pencabutan mungkin memakan waktu hingga satu jam untuk muncul dalam tanggapan OCSP.

Anda dapat mengakses AWS Private CA menggunakan AWS Management Console, the AWS CLI, dan AWS Private CA API. Topik berikut ini akan menunjukkan kepada Anda cara menggunakan konsol tersebut dan CLI. Untuk mempelajari API selengkapnya, lihat Referensi AWS Private Certificate Authority API. Untuk sampel Java yang menunjukkan cara menggunakan API, lihat Menggunakan AWS Private CA API (contoh Java).

Topik

Membuat CA pribadi

Anda dapat menggunakan prosedur di bagian ini untuk membuat CA root atau CA bawahan, menghasilkan hierarki hubungan kepercayaan yang dapat diaudit yang sesuai dengan kebutuhan organisasi Anda. Anda dapat membuat CA menggunakan AWS Management Console, bagian PCA dari AWS CLI, atau AWS CloudFormation.

Untuk informasi tentang memperbarui konfigurasi CA yang telah Anda buat, lihatMemperbarui CA privat Anda.

Untuk informasi tentang cara menggunakan CA untuk menandatangani sertifikat entitas akhir untuk pengguna, perangkat, dan aplikasi, lihat Menerbitkan sertifikat entitas akhir pribadi.

> **Note**
>
> Akun Anda akan dikenakan harga bulanan untuk setiap CA privat mulai sejak Anda membuatnya.
> Untuk informasi AWS Private CA harga terbaru, lihat AWS Private Certificate Authority Harga. Anda juga dapat menggunakan kalkulator AWS harga untuk memperkirakan biaya.

Topik

Prosedur untuk membuat CA (konsol)

Selesaikan langkah-langkah berikut untuk membuat CA pribadi menggunakan AWS Management Console.

Untuk mulai menggunakan konsol

Masuk ke AWS akun Anda dan buka AWS Private CA konsol di**`https://console.aws.amazon.com/acm-pca/home`**.

- Jika Anda membuka konsol di Wilayah di mana Anda tidak memiliki CA pribadi, halaman pengantar akan muncul. Pilih Buat CA pribadi.

- Jika Anda membuka konsol di Wilayah tempat Anda telah membuat CA, halaman otoritas sertifikat pribadi terbuka dengan daftar CA Anda. Pilih Buat CA.

Opsi mode

Pada bagian Opsi mode konsol, pilih mode kedaluwarsa sertifikat yang dikeluarkan CA Anda.

- Tujuan umum - Mengeluarkan sertifikat yang dapat dikonfigurasi dengan tanggal kedaluwarsa apa pun. Ini adalah opsi default.
- Sertifikat berumur pendek - Menerbitkan sertifikat dengan masa berlaku maksimum tujuh hari. Periode validitas yang singkat dapat menggantikan dalam beberapa kasus untuk mekanisme pencabutan.

Opsi tipe CA

Pada bagian Opsi jenis konsol, pilih jenis otoritas sertifikat pribadi yang ingin Anda buat.

- Memilih Root menetapkan hierarki CA baru. CA ini didukung dengan sertifikat yang ditandatangani sendiri. Sertifikat ini berfungsi sebagai otoritas penandatanganan terakhir untuk CA lain dan sertifikat entitas akhir dalam hierarki tersebut.
- Memilih Bawahan menciptakan CA yang harus ditandatangani oleh CA induk di atasnya dalam hierarki. CA bawahan biasanya digunakan untuk membuat CA bawahan lainnya atau untuk mengeluarkan sertifikat entitas akhir kepada pengguna, komputer, dan aplikasi.

> **Note**
>
> AWS Private CA menyediakan proses penandatanganan otomatis ketika CA induk CA bawahan Anda juga di-host oleh AWS Private CA. Yang Anda lakukan hanyalah memilih CA induk untuk digunakan.
> CA bawahan Anda mungkin perlu ditandatangani oleh penyedia layanan kepercayaan eksternal. Jika demikian, AWS Private CA memberi Anda permintaan penandatanganan sertifikat (CSR) yang harus Anda unduh dan gunakan untuk mendapatkan sertifikat CA yang ditandatangani. Untuk informasi selengkapnya, lihat Memasang sertifikat CA bawahan yang ditandatangani oleh CA induk eksternal.

Opsi nama yang dibedakan subjek

Di bawah Opsi nama yang dibedakan Subjek, konfigurasikan nama subjek CA pribadi Anda. Anda harus memasukkan nilai untuk setidaknya satu dari opsi berikut:

- Organisasi (O) — Misalnya, nama perusahaan
- Unit Organisasi (OU) — Misalnya, divisi dalam perusahaan
- Nama negara (C) — Kode negara dua huruf
- Nama negara bagian atau provinsi — Nama lengkap negara bagian atau provinsi
- Nama lokalitas — Nama kota
- Common Name (CN) — String yang dapat dibaca manusia untuk mengidentifikasi CA.

> **Note**
>
> Anda dapat lebih lanjut menyesuaikan nama subjek sertifikat dengan menerapkan template ApiPassThrough pada saat penerbitan. Untuk informasi lebih lanjut dan contoh terperinci, lihatMenerbitkan sertifikat dengan nama subjek kustom menggunakan template ApiPassThrough.

Karena sertifikat dukungan ditandatangani sendiri, informasi subjek yang Anda berikan untuk CA pribadi mungkin lebih jarang daripada yang dikandung CA publik. Untuk informasi selengkapnya tentang masing-masing nilai yang membentuk nama subjek yang dibedakan, lihat RFC 5280.

Opsi algoritma kunci

Di bawah opsi Algoritma kunci, pilih algoritma kunci dan ukuran bit kunci. Nilai default adalah algoritme RSA dengan panjang kunci 2048 bit. Anda dapat memilih dari algoritma berikut:

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

Opsi pencabutan sertifikat

Di bawah opsi pencabutan sertifikat, Anda dapat memilih dari dua metode berbagi status pencabutan dengan klien yang menggunakan sertifikat Anda:

- Aktifkan distribusi CRL
- Nyalakan OCSP

Anda dapat mengonfigurasi salah satu, keduanya, atau kedua opsi pencabutan ini untuk CA Anda. Meskipun opsional, pencabutan terkelola direkomendasikan sebagai praktik terbaik. Sebelum menyelesaikan langkah ini, lihat Menyiapkan metode pencabutan sertifikat informasi tentang keunggulan masing-masing metode, pengaturan awal yang mungkin diperlukan, dan fitur pencabutan tambahan.

> **Note**
>
> Jika Anda membuat CA tanpa mengonfigurasi pencabutan, Anda selalu dapat mengonfigurasinya nanti. Untuk informasi selengkapnya, lihat Memperbarui CA privat Anda.

Untuk mengkonfigurasi CRL

1. Di bawah opsi pencabutan sertifikat, pilih Aktifkan distribusi CRL.
2. Untuk membuat bucket Amazon S3 untuk entri CRL Anda, pilih Buat bucket S3 baru dan ketikkan nama bucket unik. (Anda tidak perlu menyertakan jalur ke bucket.) Jika tidak, di bawah URI bucket S3, pilih bucket yang ada dari daftar.

   Saat Anda membuat bucket baru melalui konsol, AWS Private CA coba lampirkan kebijakan akses yang diperlukan ke bucket, dan nonaktifkan setelan Blokir Akses Publik (BPA) default S3 di dalamnya. Jika Anda menentukan bucket yang sudah ada, Anda harus memastikan bahwa BPA dinonaktifkan untuk akun dan bucket. Jika tidak, operasi untuk membuat CA gagal. Jika CA berhasil dibuat, Anda harus tetap melampirkan kebijakan secara manual sebelum Anda dapat mulai membuat CRL. Gunakan salah satu pola kebijakan yang dijelaskan dalam Kebijakan akses untuk CRL di Amazon S3 . Untuk informasi selengkapnya, lihat Menambahkan kebijakan bucket menggunakan konsol Amazon S3.

> **Important**
>
> Upaya untuk membuat CA menggunakan AWS Private CA konsol gagal jika semua kondisi berikut berlaku:
>
> - Anda sedang menyiapkan CRL.
> - Anda meminta AWS Private CA untuk membuat bucket S3 secara otomatis.
> - Anda menegakkan pengaturan BPA di S3.
>
> Dalam situasi ini, konsol membuat ember, tetapi mencoba dan gagal membuatnya dapat diakses publik. Periksa pengaturan Amazon S3 Anda jika ini terjadi, nonaktifkan BPA sesuai kebutuhan, lalu ulangi prosedur untuk membuat CA. Untuk informasi lebih lanjut, lihat Memblokir akses publik ke penyimpanan Amazon S3 Anda.

3. Perluas pengaturan CRL untuk opsi konfigurasi tambahan.

   - Tambahkan Nama CRL Kustom untuk membuat alias untuk bucket Amazon S3. Nama ini ada dalam sertifikat yang diterbitkan oleh CA di ekstensi “CRL Distribution Points” yang ditentukan oleh RFC 5280.
   - Ketik Validitas dalam beberapa hari CRL Anda akan tetap valid. Nilai default-nya adalah 7 hari. Untuk CRL online, masa berlaku 2-7 hari adalah umum. AWS Private CA mencoba meregenerasi CRL pada titik tengah periode yang ditentukan.

4. Perluas pengaturan S3 untuk konfigurasi opsional versi Bucket dan pencatatan akses Bucket.

Untuk mengkonfigurasi OCSP

1. Di bawah opsi pencabutan sertifikat, pilih Aktifkan OCSP.
2. Di bidang endpoint OCSP Kustom - opsional, Anda dapat memberikan nama domain yang memenuhi syarat (FQDN) untuk titik akhir OCSP non-Amazon.

   Saat Anda memberikan FQDN di bidang ini, AWS Private CA masukkan FQDN ke ekstensi Akses Informasi Otoritas dari setiap sertifikat yang dikeluarkan sebagai pengganti URL default untuk responden OCSP. AWS Ketika titik akhir menerima sertifikat yang berisi FQDN kustom, ia menanyakan alamat tersebut untuk respons OCSP. Agar mekanisme ini berfungsi, Anda perlu mengambil dua tindakan tambahan:

- Gunakan server proxy untuk meneruskan lalu lintas yang tiba di FQDN kustom Anda ke responder OCSP. AWS
- Tambahkan catatan CNAME yang sesuai ke database DNS Anda.

> **Tip**
>
> Untuk informasi selengkapnya tentang penerapan solusi OCSP lengkap menggunakan CNAME kustom, lihat. Mengkonfigurasi URL Kustom untuk AWS Private CA OCSP

Misalnya, berikut adalah catatan CNAME untuk OCSP yang disesuaikan seperti yang akan muncul di Amazon Route 53.

Nama catatan	Tipe	Kebijakan perutean	Diferensiator	Nilai/Rutekan lalu lintas ke
alternatif.example.com	CNAME	Sederhana	-	proxy.example.com

> **Note**
>
> Nilai CNAME tidak boleh menyertakan awalan protokol seperti “http://” atau “https://”.

Tambahkan tag

Di bawah Tambahkan tag, Anda dapat menandai CA Anda secara opsional. Tag adalah pasangan nilai kunci yang berfungsi sebagai metadata untuk mengidentifikasi dan mengatur sumber daya. AWS Untuk daftar parameter AWS Private CA tag dan petunjuk tentang cara menambahkan tag ke CA setelah pembuatan, lihatMengelola tag untuk CA pribadi Anda.

> **Note**
>
> Untuk melampirkan tag ke CA pribadi selama prosedur pembuatan, administrator CA harus terlebih dahulu mengaitkan kebijakan IAM sebaris dengan

`CreateCertificateAuthority` tindakan dan secara eksplisit mengizinkan penandaan. Untuk informasi selengkapnya, lihat Tag-on-create: Melampirkan tag ke CA pada saat pembuatan.

Opsi izin CA

Di bawah opsi izin CA, Anda dapat secara opsional mendelegasikan izin perpanjangan otomatis ke kepala layanan. AWS Certificate Manager ACM hanya dapat memperbarui sertifikat entitas akhir privat secara otomatis yang dibuat oleh CA ini, jika izin ini diberikan. Anda dapat menetapkan izin perpanjangan kapan saja dengan AWS Private CACreatePermissionAPI atau perintah CLI create-permission.

Dafault-nya adalah untuk mengaktifkan izin ini.

> **Note**
>
> AWS Certificate Manager tidak mendukung pembaruan otomatis sertifikat berumur pendek.

Harga

Di bawah Harga, konfirmasikan bahwa Anda memahami harga untuk CA pribadi.

> **Note**
>
> Untuk informasi AWS Private CA harga terbaru, lihat AWS Private Certificate Authority Harga. Anda juga dapat menggunakan kalkulator AWS harga untuk memperkirakan biaya.

Buat CA

Pilih Buat CA setelah Anda memeriksa semua informasi yang dimasukkan untuk akurasi. Halaman detail untuk CA terbuka dan menampilkan statusnya sebagai sertifikat Tertunda.

> **Note**
>
> Saat berada di halaman detail, Anda dapat menyelesaikan konfigurasi CA Anda dengan memilih Tindakan, Instal sertifikat CA, atau Anda dapat kembali nanti ke daftar otoritas sertifikat pribadi dan menyelesaikan prosedur instalasi yang berlaku dalam kasus Anda:

- Memasang sertifikat CA root
- Memasang sertifikat CA bawahan yang diselenggarakan oleh AWS Private CA
- Memasang sertifikat CA bawahan yang ditandatangani oleh CA induk eksternal

Prosedur untuk membuat CA (CLI)

Gunakan perintah create-certificate-authority untuk membuat CA privat. Anda harus menentukan konfigurasi CA (berisi algoritme dan informasi nama subjek), konfigurasi pencabutan (jika Anda berencana menggunakan OCSP dan/atau CRL), dan jenis CA (root atau bawahan). Rincian konfigurasi konfigurasi dan pencabutan terkandung dalam dua file yang Anda berikan sebagai argumen ke perintah. Secara opsional, Anda juga dapat mengonfigurasi mode penggunaan CA (untuk menerbitkan sertifikat standar atau jangka pendek), melampirkan tag, dan menyediakan token idempotensi.

Jika mengkonfigurasi CRL, Anda harus membuat bucket Amazon S3 aman yang siap digunakan sebelum Anda menerbitkan perintah create-certificate-authority. Untuk informasi lebih lanjut, lihat Kebijakan akses untuk CRL di Amazon S3 .

File konfigurasi CA menentukan informasi berikut:

- Nama algoritme
- Ukuran kunci yang akan digunakan untuk membuat kunci privat CA
- Jenis algoritme penandatanganan yang CA gunakan untuk menandatangan
- Informasi subjek X.500

Konfigurasi pencabutan untuk OCSP mendefinisikan `OcspConfiguration` objek dengan informasi berikut:

- `Enabled`Bendera diatur ke “true”.
- (Opsional) CNAME kustom dideklarasikan sebagai nilai untuk`OcspCustomCname`.

Konfigurasi pencabutan untuk CRL mendefinisikan `CrlConfiguration` objek dengan informasi berikut:

- `Enabled`Bendera diatur ke “true”.

- Periode kedaluwarsa CRL dalam beberapa hari (masa berlaku CRL).
- Bucket Amazon S3 yang akan berisi CRL.
- (Opsional) ObjectAcl Nilai S3 yang menentukan apakah CRL dapat diakses publik. Dalam contoh yang disajikan di sini, akses publik diblokir. Untuk informasi selengkapnya, lihat Mengaktifkan S3 Block Public Access (BPA) dengan CloudFront.
- (Opsional) Alias CNAME untuk bucket S3 yang disertakan dalam sertifikat yang diterbitkan oleh CA. Jika CRL tidak dapat diakses publik, ini akan mengarah ke mekanisme distribusi seperti Amazon. CloudFront
- (Opsional) `CrlDistributionPointExtensionConfiguration` Objek dengan informasi berikut:
  - `OmitExtension`Bendera disetel ke “true” atau “false”. Ini mengontrol apakah nilai default untuk ekstensi CDP akan ditulis ke sertifikat yang dikeluarkan oleh CA. Untuk informasi selengkapnya tentang ekstensi CDP, lihatMenentukan URI Titik Distribusi CRL (CDP) . A CustomCname tidak dapat OmitExtension diatur jika “benar”.

> **Note**
>
> Anda dapat mengaktifkan kedua mekanisme pencabutan pada CA yang sama dengan mendefinisikan `OcspConfiguration` objek dan objek. `CrlConfiguration` Jika Anda tidak memberikan --revocation-configuration parameter, kedua mekanisme dinonaktifkan secara default. Jika Anda memerlukan dukungan validasi pencabutan nanti, lihat. Memperbarui CA (CLI)

Contoh berikut mengasumsikan bahwa Anda telah menyiapkan direktori `.aws` konfigurasi Anda dengan Region, endpoint, dan kredensyal default yang valid. Untuk informasi tentang mengonfigurasi AWS CLI lingkungan Anda, lihat Pengaturan konfigurasi dan file kredensyal. Untuk keterbacaan, kami menyediakan konfigurasi CA dan input pencabutan sebagai file JSON dalam perintah contoh. Ubah file contoh sesuai kebutuhan untuk Anda gunakan.

Semua contoh menggunakan file `ca_config.txt` konfigurasi berikut kecuali dinyatakan lain.

Berkas: ca_config.txt

```
{
   "KeyAlgorithm":"RSA_2048",
   "SigningAlgorithm":"SHA256WITHRSA",
```

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

Contoh 1: Buat CA dengan OCSP diaktifkan

Dalam contoh ini, file pencabutan mengaktifkan dukungan OCSP default, yang menggunakan AWS Private CA responden untuk memeriksa status sertifikat.

File: revoke_config.txt untuk OCSP

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

Perintah

```
$ 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
```

Jika berhasil, perintah ini akan menghasilkan Amazon Resource Name (ARN) dari CA baru.

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

Perintah

```
$ 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
```

Jika berhasil, perintah ini mengeluarkan Amazon Resource Name (ARN) dari CA.

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

Gunakan perintah berikut untuk memeriksa konfigurasi CA Anda.

```
$ 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
```

Deskripsi ini harus berisi bagian berikut.

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

Contoh 2: Buat CA dengan OCSP dan CNAME kustom diaktifkan

Dalam contoh ini, file pencabutan memungkinkan dukungan OCSP yang disesuaikan. `OcspCustomCname`Parameter mengambil nama domain yang sepenuhnya memenuhi syarat (FQDN) sebagai nilainya.

Saat Anda memberikan FQDN di bidang ini, AWS Private CA masukkan FQDN ke ekstensi Akses Informasi Otoritas dari setiap sertifikat yang dikeluarkan sebagai pengganti URL default untuk responden OCSP. AWS Ketika titik akhir menerima sertifikat yang berisi FQDN kustom, ia menanyakan alamat tersebut untuk respons OCSP. Agar mekanisme ini berfungsi, Anda perlu mengambil dua tindakan tambahan:

- Gunakan server proxy untuk meneruskan lalu lintas yang tiba di FQDN kustom Anda ke responder OCSP. AWS
- Tambahkan catatan CNAME yang sesuai ke database DNS Anda.

> **Tip**
>
> Untuk informasi selengkapnya tentang penerapan solusi OCSP lengkap menggunakan CNAME kustom, lihat. Mengkonfigurasi URL Kustom untuk AWS Private CA OCSP

Misalnya, berikut adalah catatan CNAME untuk OCSP yang disesuaikan seperti yang akan muncul di Amazon Route 53.

Nama catatan	Tipe	Kebijakan perutean	Diferensiator	Nilai/Rutekan lalu lintas ke
alternatif.example.com	CNAME	Sederhana	-	proxy.example.com

> **Note**
>
> Nilai CNAME tidak boleh menyertakan awalan protokol seperti “http://” atau “https://”.

File: revoke_config.txt untuk OCSP

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

Perintah

```
$ 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
```

Jika berhasil, perintah ini mengeluarkan Amazon Resource Name (ARN) dari CA.

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

Gunakan perintah berikut untuk memeriksa konfigurasi CA Anda.

```
$ 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
```

Deskripsi ini harus berisi bagian berikut.

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

Contoh 3: Buat CA dengan CRL terlampir

Dalam contoh ini, konfigurasi pencabutan mendefinisikan parameter CRL.

Berkas: revoke_config.txt

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

```
}
```

Perintah

```
$ 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
```

Jika berhasil, perintah ini mengeluarkan Amazon Resource Name (ARN) dari CA.

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

Gunakan perintah berikut untuk memeriksa konfigurasi CA Anda.

```
$ 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
```

Deskripsi ini harus berisi bagian berikut.

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

Contoh 4: Buat CA dengan CRL terlampir dan CNAME kustom diaktifkan

Dalam contoh ini, konfigurasi pencabutan mendefinisikan parameter CRL yang menyertakan CNAME kustom.

Berkas: revoke_config.txt

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

Perintah

```
$ 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
```

Jika berhasil, perintah ini mengeluarkan Amazon Resource Name (ARN) dari CA.

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

Gunakan perintah berikut untuk memeriksa konfigurasi CA Anda.

```
$ 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
```

Deskripsi ini harus berisi bagian berikut.

```
"RevocationConfiguration": {
   ...
   "CrlConfiguration": {
      "Enabled": true,
      "ExpirationInDays": 7,
```

```
      "CustomCname": "alternative.example.com",
      "S3BucketName": "DOC-EXAMPLE-BUCKET",
   ...
   }
}
```

Contoh 5: Buat CA dan tentukan mode penggunaan

Dalam contoh ini, mode penggunaan CA ditentukan saat membuat CA. Jika tidak ditentukan, parameter mode penggunaan default ke GENERAL_PURPOSE. Dalam contoh ini, parameter diatur ke SHORT_LIVED_CERTIFICATE, yang berarti bahwa CA akan mengeluarkan sertifikat dengan masa berlaku maksimum tujuh hari. Dalam situasi di mana tidak nyaman untuk mengonfigurasi pencabutan, sertifikat berumur pendek yang telah dikompromikan dengan cepat kedaluwarsa sebagai bagian dari operasi normal. Akibatnya, contoh CA ini tidak memiliki mekanisme pencabutan.

> **Note**
>
> AWS Private CA tidak melakukan pemeriksaan validitas pada sertifikat CA root.

```
$ 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
```

Gunakan describe-certificate-authorityperintah di AWS CLI untuk menampilkan rincian tentang CA yang dihasilkan, seperti yang ditunjukkan pada perintah berikut:

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

Contoh 6: Buat CA untuk login Active Directory

Anda dapat membuat CA pribadi yang cocok untuk digunakan di toko Enterprise NTAuth Microsoft Active Directory (AD), di mana ia dapat mengeluarkan sertifikat card-logon atau domain-controller. Untuk informasi tentang mengimpor sertifikat CA ke AD, lihat Cara mengimpor sertifikat otoritas sertifikasi pihak ketiga (CA) ke dalam toko Enterprise NTAuth.

Alat Microsoft certutil dapat digunakan untuk mempublikasikan sertifikat CA di AD dengan menjalankan opsi. -dspublish Sertifikat yang diterbitkan untuk AD dengan certutil dipercaya di seluruh hutan. Dengan menggunakan kebijakan grup, Anda juga dapat membatasi kepercayaan pada subset dari seluruh hutan, misalnya, satu domain atau sekelompok komputer dalam domain. Agar logon berfungsi, CA penerbit juga harus dipublikasikan di toko NTAuth. Untuk informasi selengkapnya, lihat Mendistribusikan Sertifikat ke Komputer Klien dengan Menggunakan Kebijakan Grup.

Contoh ini menggunakan file `ca_config_AD.txt` konfigurasi berikut.

Berkas: 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"
         }
      ]
   }
}
```

Perintah

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

Jika berhasil, perintah ini mengeluarkan Amazon Resource Name (ARN) dari CA.

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

Gunakan perintah berikut untuk memeriksa konfigurasi CA Anda.

```
$ 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
```

Deskripsi ini harus berisi bagian berikut.

```
...

"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"
      }
   ]
}
...
```

Contoh 7: Buat CA Materi dengan CRL terlampir dan ekstensi CDP dihilangkan dari sertifikat yang diterbitkan

Anda dapat membuat CA pribadi yang cocok untuk menerbitkan sertifikat untuk standar rumah pintar Matter. Dalam contoh ini, konfigurasi CA dalam `ca_config_PAA.txt` mendefinisikan Matter Product Attestation Authority (PAA) dengan Vendor ID (VID) disetel ke FFF1.

Berkas: 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"
      }
    ]
  }
}
```

Konfigurasi pencabutan memungkinkan CRL, dan mengonfigurasi CA untuk menghilangkan URL CDP default dari sertifikat yang dikeluarkan.

Berkas: revoke_config.txt

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

Perintah

```
$ 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
```

Jika berhasil, perintah ini mengeluarkan Amazon Resource Name (ARN) dari CA.

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

Gunakan perintah berikut untuk memeriksa konfigurasi CA Anda.

```
$ 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
```

Deskripsi ini harus berisi bagian berikut.

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

Menggunakan AWS CloudFormation untuk membuat CA

Untuk informasi tentang membuat CA pribadi menggunakan AWS CloudFormation, lihat Referensi Jenis AWS Private CA Sumber Daya di Panduan AWS CloudFormation Pengguna.

Membuat dan menginstal sertifikat CA

Selesaikan prosedur berikut untuk membuat dan menginstal sertifikat CA privat Anda. CA Anda kemudian akan siap digunakan.

AWS Private CA mendukung tiga skenario untuk menginstal sertifikat CA:

- Menginstal sertifikat untuk root CA yang dihosting oleh AWS Private CA
- Memasang sertifikat CA bawahan yang otoritas induknya dihosting oleh AWS Private CA
- Menginstal sertifikat CA bawahan yang otoritas induknya di-host secara eksternal

Bagian berikut menjelaskan prosedur untuk setiap skenario. Prosedur konsol dimulai pada halaman konsol, CA privat.

Algoritma penandatanganan yang kompatibel

Dukungan algoritma penandatanganan untuk sertifikat CA tergantung pada algoritma penandatanganan CA induk dan pada Wilayah AWS. Kendala berikut berlaku untuk konsol dan operasi. AWS CLI

- CA induk dengan algoritma penandatanganan RSA dapat mengeluarkan sertifikat dengan algoritme berikut:
  - SHA256 RSA
  - SHA384 RSA
  - SHA512 RSA
- Dalam warisan Wilayah AWS, CA induk dengan algoritme penandatanganan EDCSA dapat mengeluarkan sertifikat dengan algoritme berikut:
  - SHA256 ECDSA
  - SHA384 ECDSA
  - SHA512 ECDSA

  Warisan Wilayah AWS meliputi:

Nama Wilayah	Lokasi geografis
eu-north-1	Eropa (Stockholm)
me-south-1	Timur Tengah (Bahrain)
ap-south-1	Asia Pasifik (Mumbai)
eu-west-3	Eropa (Paris)
us-east-2	AS Timur (Ohio)
af-south-1	Afrika (Cape Town)

Nama Wilayah	Lokasi geografis
eu-west-1	Eropa (Irlandia)
eu-central-1	Eropa (Frankfurt)
sa-east-1	Amerika Selatan (Sao Paulo)
ap-east-1	Asia Pasifik (Hong Kong)
us-east-1	AS Timur (Virginia Utara)
ap-northeast-2	Asia Pasifik (Seoul)
eu-west-2	Eropa (London)
ap-northeast-1	Asia Pasifik (Tokyo)
us-gov-east-1	AWS GovCloud (AS-Timur)
us-gov-west-1	AWS GovCloud (AS-Barat)
us-west-2	AS Barat (Oregon)
us-west-1	AS Barat (California Utara)
ap-southeast-1	Asia Pasifik (Singapura)
ap-southeast-2	Asia Pasifik (Sydney)

- Dalam non-warisan Wilayah AWS, aturan berikut berlaku untuk EDCSA:

- CA induk dengan algoritma penandatanganan EC_Prime256v1 dapat mengeluarkan sertifikat dengan ECDSA P256.
- CA induk dengan algoritma penandatanganan EC_Secp384R1 dapat mengeluarkan sertifikat dengan ECDSA P384.

Memasang sertifikat CA root

Anda dapat menginstal sertifikat CA root dari AWS Management Console atau file AWS CLI.

Untuk membuat dan menginstal sertifikat untuk CA root pribadi Anda (konsol)

1. (Opsional) Jika Anda belum berada di halaman detail CA, buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home. Pada halaman Private Certificate Authority, pilih CA root dengan status Sertifikat tertunda atau Aktif.
2. Pilih Tindakan, Instal sertifikat CA untuk membuka halaman sertifikat CA root Install.
3. Di bawah Tentukan parameter sertifikat CA root, tentukan parameter sertifikat berikut:

   - Validitas - Menentukan tanggal kedaluwarsa dan waktu untuk sertifikat CA. Masa berlaku AWS Private CA default untuk sertifikat CA root adalah 10 tahun.
   - Algoritma tanda tangan - Menentukan algoritma penandatanganan yang akan digunakan saat root CA mengeluarkan sertifikat baru. Opsi yang tersedia bervariasi sesuai dengan Wilayah AWS tempat Anda membuat CA. Untuk informasi selengkapnya, lihatAlgoritma penandatanganan yang kompatibel,Algoritme kriptografi yang didukung, dan SigningAlgorithmdi CertificateAuthorityKonfigurasi.
     - SHA256 RSA
     - SHA384 RSA
     - SHA512 RSA

   Tinjau pengaturan Anda untuk kebenaran, lalu pilih Konfirmasi dan instal. AWS Private CA mengekspor CSR untuk CA Anda, menghasilkan sertifikat menggunakan templat sertifikat CA root, dan menandatangani sendiri sertifikat tersebut. AWS Private CA kemudian mengimpor sertifikat CA root yang ditandatangani sendiri.
4. Halaman detail untuk CA menampilkan status instalasi (sukses atau gagal) di bagian atas. Jika penginstalan berhasil, CA root yang baru selesai menampilkan status Aktif di panel Umum.

Untuk membuat dan menginstal sertifikat untuk root pribadi CA (AWS CLI)

1. Buat permintaan penandatanganan sertifikat (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
```

File yang dihasilkan`ca.csr`, file PEM yang dikodekan dalam format base64, memiliki tampilan sebagai berikut.

```
-----BEGIN CERTIFICATE REQUEST-----
MIIC1DCCAbwCAQAwbTELMAkGA1UEBhMCVVMxFTATBgNVBAoMDEV4YW1wbGUgQ29y
cDEOMAwGA1UECwwFU2FsZXMxCzAJBgNVBAgMAldBMRgwFgYDVQQDDA93d3cuZXhh
bXBsZS5jb20xEDAOBgNVBAcMB1NlYXR0bGUwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQDD+7eQChWUO2m6pHslI7AVSFkWvbQofKIHvbvy7wm8VO9/BuI7
LE/jrnd1jGoyI7jaMHKXPtEP3uNlCzv+oEza07OjgjqPZVehtA6a3/3vdQ1qCoD2
rXpv6VIzcq2onx2X7m+Zixwn2oY1l1ELXP7I5g0GmUStymq+pY5VARPy3vTRMjgC
JEiz8w7VvC15uIsHFAWa2/NvKyndQMPaCNft238wesV5s2cXOUS173jghIShg99o
ymf0TRUgvAGQMCXvsW07MrP5VDmBU7k/AZ9ExsUfMe2OB++fhfQWr2N7/lpC4+DP
qJTfXTEexLfRTLeLuGEaJL+c6fMyG+Yk53tZAgMBAAGgIjAgBgkqhkiG9w0BCQ4x
EzARMA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEBAA7xxLVI5s1B
qmXMMT44y1DZtQx3RDPanMNGLGO1TmLtyqqnUH49Tla+2p7nrl0tojUf/3PaZ52F
QN09SrFk8qtYSKnMGd5PZL0A+NFsNW+w4BAQNKlg9m617YEsnkztbfKRloaJNYoA
HZaRvbA0lMQ/tU2PKZR2vnao444Ugm0O/t3jx5rj817b31hQcHHQ0lQuXV2kyTrM
ohWeLf2fL+K0xJ9ZgXD4KYnY0zarpreA5RBeO5xs3Ms+oGWc13qQfMBx33vrrz2m
dw5iKjg71uuUUmtDV6ewwGa/VO5hNinYAfogdu5aGuVbnTFT3n45B8WHz2+9r0dn
bA7xUel1SuQ=
-----END CERTIFICATE REQUEST-----
```

Anda dapat menggunakan OpenSSL untuk melihat dan memverifikasi isi CSR.

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

Ini menghasilkan output yang mirip dengan yang berikut ini.

```
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. Menggunakan CSR dari langkah sebelumnya sebagai argumen untuk `--csr` parameter, keluarkan sertifikat root.

> **Note**
>
> Jika Anda menggunakan AWS CLI versi 1.6.3 atau yang lebih baru, gunakan awalan `fileb://` saat menentukan file input yang diperlukan. Ini memastikan bahwa AWS Private CA mem-parsing data yang dikodekan Base64 dengan benar.

```
$ 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. Ambil sertifikat akar.

```
$ 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
```

File yang dihasilkan`cert.pem`, file PEM yang dikodekan dalam format base64, memiliki tampilan sebagai berikut.

```
-----BEGIN CERTIFICATE-----
MIIDpzCCAo+gAwIBAgIRAIIuOarlQETlUQEOZJGZYdIwDQYJKoZIhvcNAQELBQAw
bTELMAkGA1UEBhMCVVMxFTATBgNVBAoMDEV4YW1wbGUgQ29ycDEOMAwGA1UECwwF
U2FsZXMxCzAJBgNVBAgMAldBMRgwFgYDVQQDDA93d3cuZXhhbXBsZS5jb20xEDAO
BgNVBAcMB1NlYXR0bGUwHhcNMjEwMzA4MTU0NjI3WhcNMjIwMzA4MTY0NjI3WjBt
MQswCQYDVQQGEwJVUzEVMBMGA1UECgwMRXhhbXBsZSBDb3JwMQ4wDAYDVQQLDAVT
YWxlczELMAkGA1UECAwCV0ExGDAWBgNVBAMMD3d3dy5leGFtcGxlLmNvbTEQMA4G
A1UEBwwHU2VhdHRsZTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMP7
```

```
t5AKFZQ7abqkeyUjsBVIWRa9tCh8oge9u/LvCbxU738G4jssT+Oud3WMajIjuNow
cpc+0Q/e42ULO/6gTNrTs6OCOo9lV6G0Dprf/e91DWoKgPatem/pUjNyraifHZfu
b5mLHCfahjWXUQtc/sjmDQaZRK3Kar6ljlUBE/Le9NEyOAIkSLPzDtW8LXm4iwcU
BZrb828rKd1Aw9oI1+3bfzB6xXmzZxc5RLXveOCEhKGD32jKZ/RNFSC8AZAwJe+x
bTsys/lUOYFTuT8Bn0TGxR8x7Y4H75+F9BavY3v+WkLj4M+olN9dMR7Et9FMt4u4
YRokv5zp8zIb5iTne1kCAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4E
FgQUaW3+r328uTLokog2TklmoBK+yt4wDgYDVR0PAQH/BAQDAgGGMA0GCSqGSIb3
DQEBCwUAA4IBAQAXjd/7UZ8RDE+PLWSDNGQdLemOBTcawF+tK+PzA4Evlmn9VuNc
g+x3oZvVZSDQBANUz0b9oPeo54aE38dW1zQm2qfTab8822aqeWMLyJ1dMsAgqYX2
t9+u6w3NzRCw8Pvz18V69+dFE5AeXmNP0Z5/gdz8H/NSpctjlzopbScRZKCSlPid
Rf3ZOPm9QP92YpWyYDkfAU04xdDo1vR0MYjKPkl4LjRqSU/tcCJnPMbJiwq+bWpX
2WJoEBXB/p15Kn6JxjI0ze2SnSI48JZ8it4fvxrhOo0VoLNIuCuNXJOwU17Rdl1W
YJidaq7je6k18AdgPA0Kh8y1XtfUH3fTaVw4
-----END CERTIFICATE-----
```

Anda dapat menggunakan OpenSSL untuk melihat dan memverifikasi isi sertifikat.

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

Ini menghasilkan output yang mirip dengan yang berikut ini.

```
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. Impor sertifikat CA root untuk menginstalnya di CA.

> **Note**
>
> Jika Anda menggunakan AWS CLI versi 1.6.3 atau yang lebih baru, gunakan awalan `fileb://` saat menentukan file input yang diperlukan. Ini memastikan bahwa AWS Private CA mem-parsing data yang dikodekan Base64 dengan benar.

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

Periksa status baru 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
```

Status sekarang muncul sebagai AKTIF.

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

Memasang sertifikat CA bawahan yang diselenggarakan oleh AWS Private CA

Anda dapat menggunakan AWS Management Console untuk membuat dan menginstal sertifikat untuk CA bawahan Anda yang AWS Private CA dihosting.

Untuk membuat dan menginstal sertifikat untuk CA bawahan Anda yang AWS Private CA dihosting

1. (Opsional) Jika Anda belum berada di halaman detail CA, buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home. Pada halaman Otoritas sertifikat pribadi, pilih CA bawahan dengan status Sertifikat tertunda atau Aktif.
2. Pilih Tindakan, Instal Sertifikat CA untuk membuka halaman Instal sertifikat CA bawahan.
3. Pada halaman Instal sertifikat CA bawahan, di bawah Pilih jenis CA, pilih AWS Private CAuntuk menginstal sertifikat yang dikelola oleh AWS Private CA.
4. Di bawah Pilih CA induk, pilih CA dari daftar CA pribadi Induk. Daftar ini disaring untuk menampilkan CA yang memenuhi kriteria berikut:
   - Anda memiliki izin untuk menggunakan CA.
   - CA tidak akan menandatangani sendiri.
   - CA berada di negara bagian`ACTIVE`.

- Mode CA adalah`GENERAL_PURPOSE`.

5. Di bawah Tentukan parameter sertifikat CA bawahan, tentukan parameter sertifikat berikut:

   - Validitas - Menentukan tanggal kedaluwarsa dan waktu untuk sertifikat CA.
   - Algoritma tanda tangan - Menentukan algoritma penandatanganan yang akan digunakan saat root CA mengeluarkan sertifikat baru. Pilihannya adalah:
     - SHA256 RSA
     - SHA384 RSA
     - SHA512 RSA
   - Panjang jalur — Jumlah lapisan kepercayaan yang dapat ditambahkan CA bawahan saat menandatangani sertifikat baru. Panjang jalur nol (default) berarti hanya sertifikat entitas akhir, dan bukan sertifikat CA, yang dapat dibuat. Panjang jalur satu atau lebih berarti bahwa CA bawahan dapat mengeluarkan sertifikat untuk membuat CA tambahan di bawahnya.
   - Template ARN - Menampilkan ARN dari template konfigurasi untuk sertifikat CA ini. Templat berubah jika Anda mengubah Panjang jalur yang ditentukan. Jika Anda membuat sertifikat menggunakan perintah CLI issue-certificate atau IssueCertificatetindakan API, Anda harus menentukan ARN secara manual. Untuk informasi tentang templat sertifikat CA yang tersedia, lihat Memahami templat sertifikat.

6. Tinjau pengaturan Anda untuk kebenaran, lalu pilih Konfirmasi dan instal. AWS Private CA mengekspor CSR, menghasilkan sertifikat menggunakan templat sertifikat CA bawahan, dan menandatangani sertifikat dengan CA induk yang dipilih. AWS Private CA kemudian mengimpor sertifikat CA bawahan yang ditandatangani.
7. Halaman detail untuk CA menampilkan status instalasi (sukses atau gagal) di bagian atas. Jika penginstalan berhasil, CA bawahan yang baru selesai menampilkan status Aktif di panel Umum.

Memasang sertifikat CA bawahan yang ditandatangani oleh CA induk eksternal

Setelah Anda membuat CA privat bawahan seperti yang dijelaskan dalam Prosedur untuk membuat CA (konsol) atau Prosedur untuk membuat CA (CLI) , Anda dapat mengaktifkannya dengan menginstal sertifikat CA yang ditandatangani oleh otoritas penandatanganan eksternal. Menandatangani sertifikat CA bawahan Anda dengan CA eksternal mengharuskan Anda terlebih

dahulu menyiapkan penyedia layanan kepercayaan eksternal sebagai otoritas penandatanganan Anda, atau mengatur penggunaan penyedia pihak ketiga.

> **Note**
>
> Prosedur untuk membuat atau memperoleh penyedia layanan kepercayaan eksternal berada di luar cakupan panduan ini.

Setelah Anda membuat CA bawahan dan Anda memiliki akses ke otoritas penandatanganan eksternal, selesaikan tugas-tugas berikut:

1. Dapatkan permintaan penandatanganan sertifikat (CSR) dari AWS Private CA.
2. Kirimkan CSR ke otoritas penandatanganan eksternal Anda dan dapatkan sertifikat CA yang ditandatangani bersamaan dengan sertifikat rantai apa pun.
3. Impor sertifikat CA dan rantai ke AWS Private CA untuk mengaktifkan CA bawahan Anda.

Untuk prosedur terperinci, lihat Sertifikat CA pribadi yang ditandatangani secara eksternal .

Mengontrol akses ke CA privat

Setiap pengguna dengan izin yang diperlukan pada CA pribadi AWS Private CA dapat menggunakan CA tersebut untuk menandatangani sertifikat lain. Pemilik CA dapat menerbitkan sertifikat atau mendelegasikan izin yang diperlukan untuk menerbitkan sertifikat kepada pengguna AWS Identity and Access Management (IAM) yang berada di dalamnya. Akun AWS Pengguna yang berada di AWS akun lain juga dapat mengeluarkan sertifikat jika diizinkan oleh pemilik CA melalui kebijakan berbasis sumber daya.

Pengguna yang berwenang, baik akun tunggal atau lintas akun, dapat menggunakan AWS Private CA atau AWS Certificate Manager sumber daya saat mengeluarkan sertifikat. Sertifikat yang dikeluarkan dari AWS Private CA IssueCertificateAPI atau perintah CLI issue-certificate tidak dikelola. Sertifikat tersebut memerlukan instalasi manual pada perangkat target dan pembaruan manual ketika mereka kedaluwarsa. Sertifikat yang dikeluarkan dari konsol ACM, ACM RequestCertificateAPI, atau perintah CLI request-certificate dikelola. Sertifikat semacam itu dapat dengan mudah dipasang di layanan yang terintegrasi dengan ACM. Jika administrator CA mengizinkannya dan akun penerbit menyediakan peran tertaut layanan untuk ACM, sertifikat terkelola diperbarui secara otomatis saat kedaluwarsa.

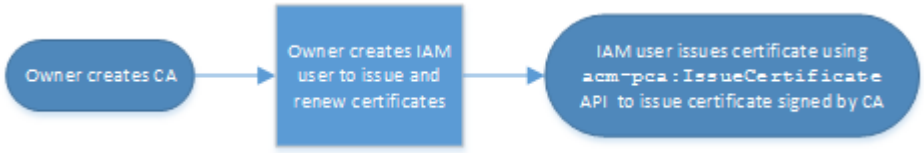
Topik

Membuat izin akun tunggal untuk pengguna IAM

Ketika administrator CA (yaitu, pemilik CA) dan penerbit sertifikat berada dalam satu AWS akun, praktik terbaik adalah memisahkan peran penerbit dan administrator dengan membuat pengguna AWS Identity and Access Management (IAM) dengan izin terbatas. Untuk informasi tentang menggunakan IAM dengan AWS Private CA, bersama dengan izin contoh, lihat. Identity and Access Management (IAM) untuk AWS Private Certificate Authority

Kasus akun tunggal 1: Menerbitkan sertifikat yang tidak dikelola

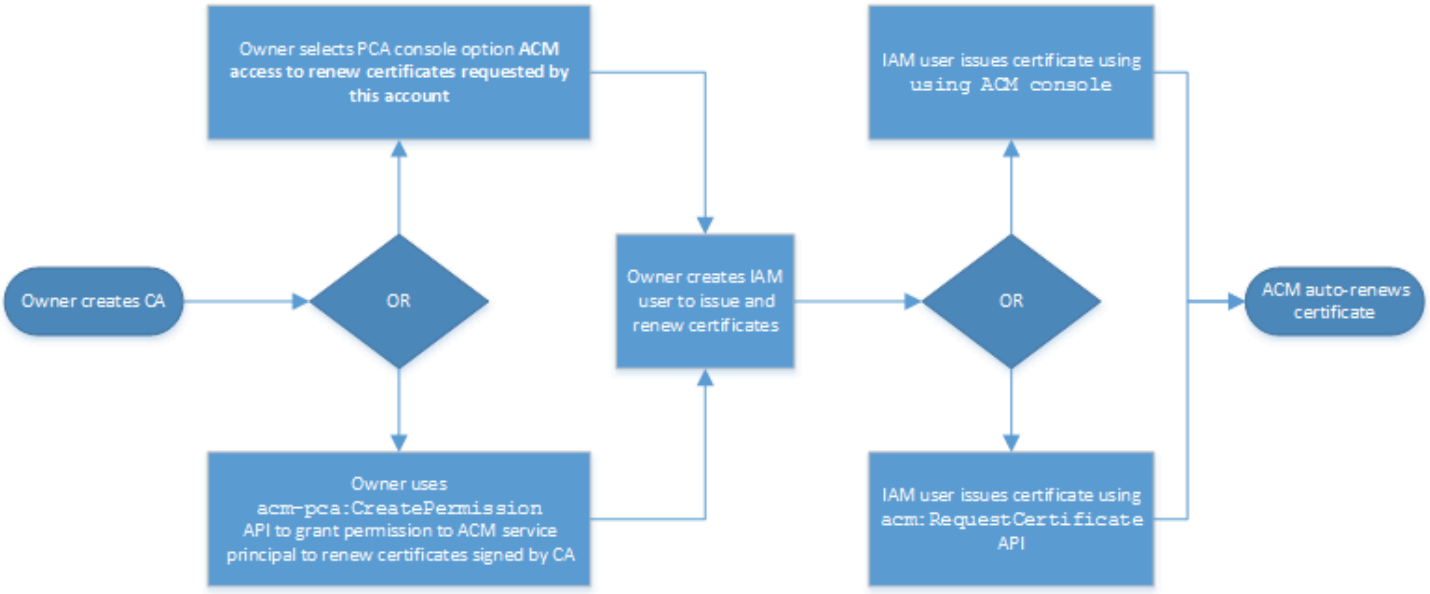
Dalam hal ini, pemilik akun membuat CA pribadi dan kemudian membuat pengguna IAM dengan izin untuk mengeluarkan sertifikat yang ditandatangani oleh CA pribadi. Pengguna IAM mengeluarkan sertifikat dengan memanggil AWS Private CA `IssueCertificate` API.



Sertifikat yang diterbitkan dengan cara ini tidak dikelola, yang berarti bahwa administrator harus mengekspornya dan menginstalnya di perangkat yang akan digunakan. Sertifikat tersebut juga harus diperbarui secara manual saat kedaluwarsa. Menerbitkan sertifikat menggunakan API ini memerlukan permintaan penandatanganan sertifikat (CSR) dan key pair yang dihasilkan di luar oleh AWS Private CA OpenSSL atau program serupa. Untuk informasi selengkapnya, lihat `IssueCertificate`dokumentasi https://docs.aws.amazon.com/privateca/latest/APIReference/API_IssueCertificate.html.

Kasus akun tunggal 2: Menerbitkan sertifikat terkelola melalui ACM

Kasus kedua ini melibatkan operasi API dari ACM dan PCA. Pemilik akun membuat pengguna CA dan IAM pribadi seperti sebelumnya. Pemilik akun kemudian memberikan izin kepada kepala layanan ACM untuk memperbarui secara otomatis sertifikat apa pun yang ditandatangani oleh CA ini. Pengguna IAM kembali mengeluarkan sertifikat, tetapi kali ini dengan memanggil ACM `RequestCertificate` API, yang menangani CSR dan pembuatan kunci. Ketika sertifikat kedaluwarsa, ACM mengotomatiskan alur kerja perpanjangan.



Pemilik akun memiliki opsi untuk memberikan izin perpanjangan melalui konsol manajemen selama atau setelah pembuatan CA atau menggunakan `CreatePermission` PCA API. Sertifikat terkelola yang dibuat dari alur kerja ini tersedia untuk digunakan dengan AWS layanan yang terintegrasi dengan ACM.

Bagian berikut berisi prosedur untuk memberikan izin perpanjangan.

Menetapkan izin perpanjangan sertifikat untuk ACM

Dengan perpanjangan terkelola di AWS Certificate Manager (ACM), Anda dapat mengotomatiskan proses perpanjangan sertifikat untuk sertifikat publik dan swasta. Agar ACM secara otomatis memperbarui sertifikat yang dihasilkan oleh CA privat, entitas keamanan layanan ACM harus diberikan semua kemungkinan izin oleh CA itu sendiri. Jika izin perpanjangan ini tidak tersedia untuk ACM, pemilik CA (atau perwakilan yang sah) harus menerbitkan ulang secara manual setiap sertifikat privat saat kedaluwarsa.

> **Important**
>
> Prosedur untuk menetapkan izin perpanjangan ini hanya berlaku ketika pemilik CA dan penerbit sertifikat berada di akun yang sama. AWS Untuk skenario lintas akun, lihat Lampirkan kebijakan untuk akses lintas akun.

Izin perpanjangan dapat didelegasikan selama pembuatan CA privat atau diubah kapan saja setelah selama CA berada di negara bagian `ACTIVE`.

Anda dapat mengelola izin CA pribadi dari AWS Private CA Konsol, AWS Command Line Interface (AWS CLI), atau AWS Private CA API:

Untuk menetapkan izin CA privat untuk ACM (konsol)

1. Masuk ke AWS akun Anda dan buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home.
2. Pada halaman Otoritas sertifikat pribadi, pilih CA pribadi Anda dari daftar.
3. Pilih Tindakan, Konfigurasikan izin CA.
4. Pilih Otorisasi akses ACM untuk memperbarui sertifikat yang diminta oleh akun ini.
5. Pilih Simpan.

Untuk mengelola izin ACM di AWS Private CA ()AWS CLI

Gunakan perintah create-permission untuk menentapkan izin ke ACM. Anda harus menetapkan izin yang diperlukan (`IssueCertificate`, `GetCertificate`, dan `ListPermissions`) agar ACM otomatis memperpanjang sertifikat Anda.

```
$ 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
```

Gunakan perintah list-permissions untuk membuat daftar izin yang didelegasikan oleh CA.

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

Gunakan perintah delete-permission untuk mencabut izin yang ditetapkan oleh CA ke kepala layanan. AWS

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

Lampirkan kebijakan untuk akses lintas akun

Saat administrator CA dan penerbit sertifikat berada d akun AWS yang berbeda, administrator CA harus berbagi akses CA. Hal ini dilakukan dengan melampirkan kebijakan berbasis sumber daya ke CA. Kebijakan ini memberikan izin penerbitan kepada prinsipal tertentu, yang dapat berupa pemilik AWS akun, pengguna IAM, AWS Organizations ID, atau ID unit organisasi.

Administrator CA dapat melampirkan dan mengelola kebijakan dengan cara berikut:

- Di konsol manajemen, menggunakan AWS Resource Access Manager (RAM), yang merupakan metode standar untuk berbagi AWS sumber daya di seluruh akun. Saat Anda membagikan sumber daya CA AWS RAM dengan prinsipal di akun lain, kebijakan berbasis sumber daya yang diperlukan akan dilampirkan ke CA secara otomatis. Untuk informasi lebih lanjut tentang RAM, lihat Panduan Pengguna AWS RAM.

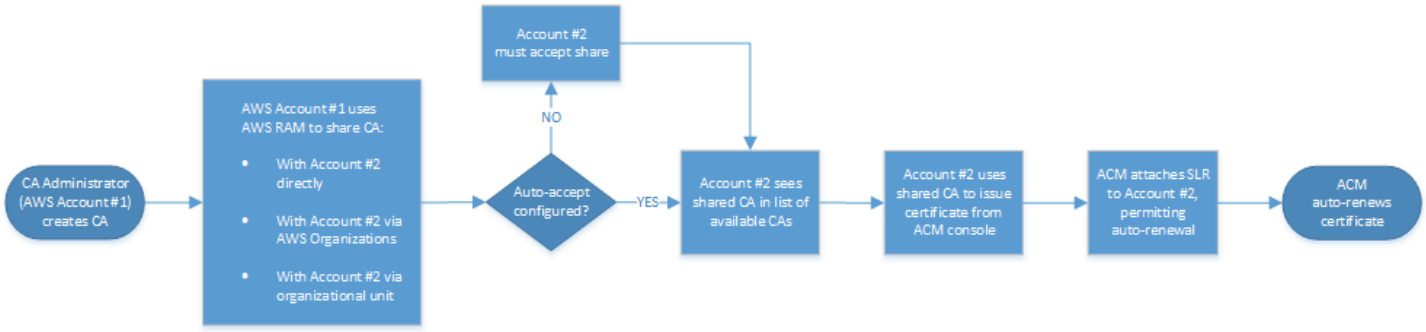
  > **Note**
  >
  > Anda dapat dengan mudah membuka konsol RAM dengan memilih CA dan kemudian memilih Tindakan, Kelola berbagi sumber daya.

- Secara terprogram, menggunakan API PCA PutPolicy,, GetPolicydan. DeletePolicy
- Secara manual, menggunakan perintah PCA put-policy, get-policy, dan delete-kebijakan di AWS CLI.

Hanya metode konsol yang memerlukan akses RAM.

Kasus lintas akun 1: Menerbitkan sertifikat terkelola dari konsol

Dalam hal ini, administrator CA menggunakan AWS Resource Access Manager (AWS RAM) untuk berbagi akses CA dengan AWS akun lain, yang memungkinkan akun tersebut mengeluarkan sertifikat ACM terkelola. Diagram menunjukkan bahwa AWS RAM dapat berbagi CA secara langsung dengan akun, atau secara tidak langsung melalui AWS Organizations ID di mana akun tersebut adalah anggota.

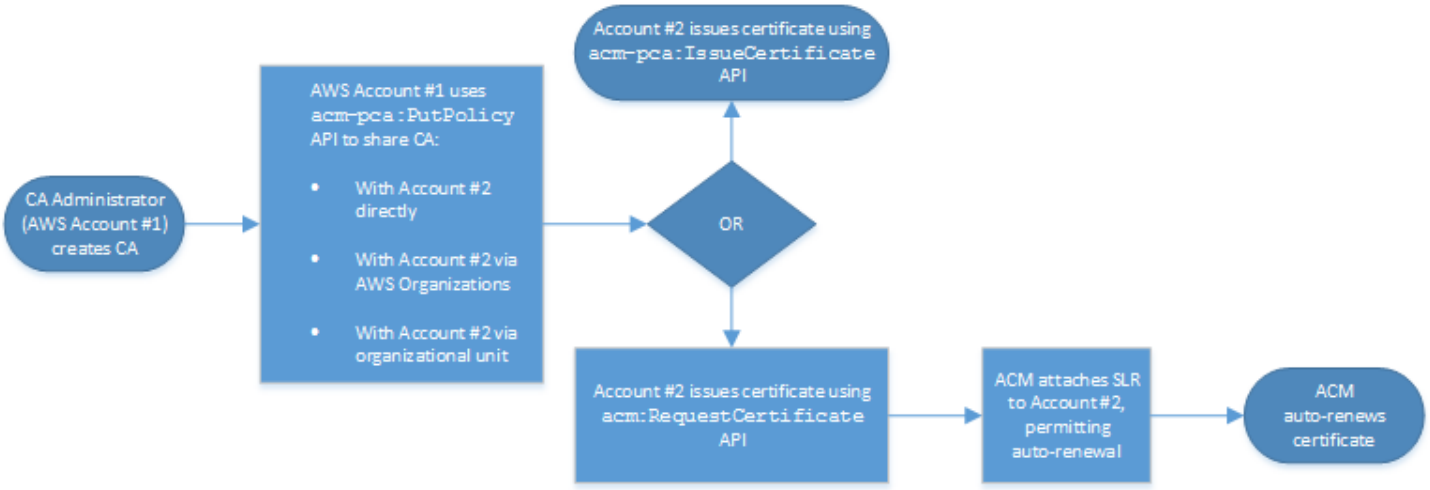


Setelah RAM berbagi sumber daya AWS Organizations, prinsipal penerima harus menerima sumber daya agar dapat diterapkan. Penerima dapat mengonfigurasi AWS Organizations untuk menerima saham yang ditawarkan secara otomatis.

> **Note**
>
> Akun penerima bertanggung jawab untuk mengonfigurasi autorenewal di ACM. Biasanya, pada kesempatan pertama CA bersama digunakan, ACM menginstal peran terkait layanan yang memungkinkannya melakukan panggilan sertifikat tanpa pengawasan. AWS Private CA Jika ini gagal (biasanya karena izin yang hilang), sertifikat dari CA tidak diperbarui secara otomatis. Hanya pengguna ACM yang dapat menyelesaikan masalah, bukan administrator CA. Untuk informasi selengkapnya, lihat Menggunakan Peran Tertaut Layanan (SLR) dengan ACM.

Kasus lintas akun 2: Menerbitkan sertifikat terkelola dan tidak terkelola menggunakan API atau CLI

Kasus kedua ini menunjukkan opsi berbagi dan penerbitan yang dimungkinkan menggunakan API AWS Certificate Manager dan AWS Private CA . Semua operasi ini juga dapat dilakukan dengan menggunakan AWS CLI perintah yang sesuai.



Karena operasi API digunakan secara langsung dalam contoh ini, penerbit sertifikat memiliki pilihan dua operasi API untuk mengeluarkan sertifikat. Tindakan API PCA `IssueCertificate` menghasilkan sertifikat tidak terkelola yang tidak akan diperpanjang secara otomatis dan harus diekspor dan diinstal secara manual. Tindakan ACM API RequestCertificatemenghasilkan sertifikat terkelola yang dapat dengan mudah diinstal pada layanan terintegrasi ACM dan diperbarui secara otomatis.

> **Note**
>
> Akun penerima bertanggung jawab untuk mengkonfigurasi perpanjangan otomatis di ACM. Biasanya, pada kesempatan pertama CA bersama digunakan, ACM menginstal peran terkait layanan yang memungkinkannya melakukan panggilan sertifikat tanpa pengawasan. AWS Private CA Jika ini gagal (biasanya karena izin hilang), sertifikat dari CA tidak akan memperbarui secara otomatis, dan hanya pengguna ACM yang dapat menyelesaikan masalah tersebut, bukan administrator CA. Untuk informasi selengkapnya, lihat Menggunakan Peran Tertaut Layanan (SLR) dengan ACM.

Mencantumkan CA privat

Anda dapat menggunakan AWS Private CA konsol atau AWS CLI untuk membuat daftar CA pribadi yang Anda miliki atau memiliki akses ke.

Untuk daftar CA yang tersedia menggunakan konsol

1. Masuk ke AWS akun Anda dan buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home.
2. Tinjau informasi dalam daftar otoritas sertifikat swasta. Anda dapat menavigasi melalui beberapa halaman CA menggunakan nomor halaman di kanan atas. Setiap CA menempati baris dengan beberapa atau semua kolom berikut ditampilkan untuk masing-masing kolom:

- Subjek - Ringkasan informasi nama terhormat untuk CA.
- Id — pengidentifikasi unik heksadesimal 32-byte dari CA.
- Status — Status CA. Nilai yang mungkin adalah Membuat, Sertifikat tertunda, Aktif, Dihapus, Dinonaktifkan, Kedaluwarsa, dan Gagal.
- Tipe — Jenis CA. Nilai yang mungkin adalah Root dan Subordinate.
- Mode — Mode CA. Nilai yang mungkin adalah Tujuan umum (mengeluarkan sertifikat yang dapat dikonfigurasi dengan tanggal kedaluwarsa apa pun) dan Sertifikat berumur pendek (menerbitkan sertifikat dengan masa berlaku maksimum tujuh hari). Periode validitas yang singkat dapat menggantikan dalam beberapa kasus untuk mekanisme pencabutan. Defaultnya adalah tujuan umum.
- Pemilik — AWS Akun yang memiliki CA. Ini mungkin akun Anda atau akun yang telah mendelegasikan izin manajemen CA kepada Anda.
- Algoritma kunci — Algoritma kunci publik yang didukung oleh CA. Nilai yang mungkin adalah RSA_2048, RSA_4096, EC_Prime256v1, dan EC_Secp384R1.
- Algoritma penandatanganan — Algoritma yang digunakan CA untuk menandatangani permintaan sertifikat. (Jangan bingung dengan `SigningAlgorithm` parameter yang digunakan untuk menandatangani sertifikat saat dikeluarkan.) Nilai yang mungkin adalah SHA256WITHECDSA, SHA384WITHECDSA, SHA512WITHECDSA, SHA256WITHRSA, SHA384WITHRSA, dan SHA512WITHRSA.

> **Note**
>
> Anda dapat menyesuaikan kolom yang ingin Anda tampilkan, serta pengaturan lainnya, dengan memilih ikon pengaturan di sudut kanan atas konsol.

Untuk daftar CA yang tersedia menggunakan AWS CLI

Gunakan perintah list-certificate-authority untuk daftar CA yang tersedia seperti yang ditunjukkan pada contoh berikut:

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

Perintah mengembalikan informasi yang mirip dengan berikut ini:

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

Melihat CA pribadi

Anda dapat menggunakan konsol ACM atau AWS CLI untuk melihat metadata terperinci tentang CA pribadi dan mengubah beberapa nilai sesuai kebutuhan. Untuk informasi rinci tentang memperbarui CA, lihatMemperbarui CA privat Anda.

Untuk melihat detail CA di konsol

1. Masuk ke AWS akun Anda dan buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home.
2. Tinjau daftar otoritas sertifikat swasta. Anda dapat menavigasi melalui beberapa halaman CA menggunakan nomor halaman di kanan atas.
3. Untuk menampilkan metadata terperinci untuk CA yang terdaftar, pilih tombol radio oleh CA yang ingin Anda periksa. Ini membuka panel detail dengan tampilan tab berikut:

   - Tab subjek — Informasi tentang nama yang dibedakan untuk CA. Untuk informasi selengkapnya, lihat Opsi nama yang dibedakan subjek. Bidang yang ditampilkan meliputi:
     - Subjek - Ringkasan bidang informasi nama yang disediakan
     - Organisasi (O) — Misalnya, nama perusahaan
     - Unit Organisasi (OU) — Misalnya, divisi dalam perusahaan
     - Nama negara (C) — Kode negara dua huruf
     - Nama negara bagian atau provinsi — Nama lengkap negara bagian atau provinsi
     - Nama lokalitas — Nama kota
     - Common Name (CN) — String yang dapat dibaca manusia untuk mengidentifikasi CA.
   - Tab sertifikat CA — Informasi tentang validitas sertifikat CA
     - Berlaku hingga - Tanggal dan waktu hingga sertifikat CA valid
     - Kedaluwarsa dalam — Jumlah hari sampai kedaluwarsa
   - Tab konfigurasi pencabutan — Pilihan Anda saat ini untuk opsi pencabutan sertifikat. Pilih Edit untuk memperbarui.
     - Distribusi Daftar Pencabutan Sertifikat (CRL) - Status Diaktifkan atau Dinonaktifkan
     - Protokol Status Sertifikat Online (OCSP) - Status Diaktifkan atau Dinonaktifkan
   - Tab izin — Pilihan izin perpanjangan sertifikat Anda saat ini untuk CA melalui (ACM) ini. AWS Certificate Manager Pilih Edit untuk memperbarui.
   - Otorisasi ACM untuk pembaruan - Status resmi atau tidak sah
   - Tab tag — Penugasan Anda saat ini untuk label yang dapat disesuaikan untuk CA ini. Pilih tag Kelola untuk diperbarui.
   - Tab berbagi sumber daya — Penugasan pembagian sumber daya Anda saat ini untuk CA melalui AWS Resource Access Manager (RAM) ini. Pilih Kelola pembagian sumber daya untuk diperbarui.

- Nama — Nama pembagian sumber daya
- Status - status pembagian sumber daya

4. Pilih bidang ID CA yang ingin Anda periksa untuk membuka panel Umum. Pengidentifikasi unik heksadesimal 32-byte CA muncul di bagian atas. Panel menyediakan informasi tambahan berikut:

   - Status — Status CA. Nilai yang mungkin adalah Membuat, Sertifikat tertunda, Aktif, Dihapus, Dinonaktifkan, Kedaluwarsa, dan Gagal.
   - ARN — Nama Sumber Daya Amazon untuk CA.
   - Pemilik — AWS Akun yang memiliki CA. Ini mungkin akun Anda (Self) atau akun yang telah mendelegasikan izin manajemen CA kepada Anda.
   - Tipe CA — Jenis CA. Nilai yang mungkin adalah Root dan Subordinate.
   - Dibuat pada — Tanggal dan waktu ketika CA dibuat.
   - Tanggal kedaluwarsa - Tanggal dan waktu ketika sertifikat CA kedaluwarsa.
   - Mode — Mode CA. Nilai yang mungkin adalah Tujuan umum (sertifikat yang dapat dikonfigurasi dengan tanggal kedaluwarsa apa pun) dan Sertifikat berumur pendek (sertifikat dengan masa berlaku maksimum tujuh hari). Periode validitas yang singkat dapat menggantikan dalam beberapa kasus untuk mekanisme pencabutan. Defaultnya adalah tujuan umum.
   - Algoritma kunci — Algoritma kunci publik yang didukung oleh CA. Nilai yang mungkin adalah RSA 2048, RSA 4096, ECDSA P2567, dan ECDSA P384.
   - Algoritma penandatanganan — Algoritma yang digunakan CA untuk menandatangani permintaan sertifikat. (Jangan bingung dengan `SigningAlgorithm` parameter yang digunakan untuk menandatangani sertifikat saat dikeluarkan.) Nilai yang mungkin adalah SHA256 ECDSA, SHA384 ECDSA, SHA512 ECDSA, SHA256 RSA, SHA384 RSA, dan SHA512 RSA
   - Standar keamanan penyimpanan utama — Tingkat kesesuaian Standar Pemrosesan Informasi Federal. Nilai yang mungkin adalah FIPS 140-2 level 3 atau lebih tinggi dan FIPS 140-2 level 3 atau lebih tinggi. Parameter ini bervariasi menurut AWS Wilayah.

Untuk melihat dan memodifikasi detail CA menggunakan AWS CLI

Gunakan describe-certificate-authorityperintah dalam AWS CLI untuk menampilkan rincian tentang CA, seperti yang ditunjukkan dalam perintah berikut:

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

Perintah mengembalikan informasi yang mirip dengan berikut ini:

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

Untuk informasi tentang memperbarui CA pribadi dari baris perintah, lihatMemperbarui CA (CLI).

Mengelola tag untuk CA pribadi Anda

Tanda adalah kata atau frasa yang bertindak sebagai metadata untuk mengidentifikasi dan mengatur sumber daya AWS . Setiap tanda terdiri dari kunci dan nilai opsional. Anda dapat menggunakan AWS Private CA konsol, AWS Command Line Interface (AWS CLI), atau PCA API untuk menambahkan, melihat, atau menghapus tag untuk CA pribadi.

Anda dapat menambahkan atau menghapus tag khusus untuk CA pribadi Anda kapan saja. Misalnya, Anda dapat menandai CA pribadi dengan pasangan nilai kunci seperti

`Environment=Prod` atau `Environment=Beta` untuk mengidentifikasi lingkungan mana CA dimaksudkan. Untuk informasi selengkapnya, lihat Membuat CA Pribadi.

> **Note**
>
> Untuk melampirkan tag ke CA pribadi selama prosedur pembuatan, administrator CA harus terlebih dahulu mengaitkan kebijakan IAM sebaris dengan `CreateCertificateAuthority` tindakan dan secara eksplisit mengizinkan penandaan. Untuk informasi selengkapnya, lihat Tag-on-create: Melampirkan tag ke CA pada saat pembuatan.

AWS Sumber daya lain juga mendukung penandaan. Anda dapat menetapkan tag yang sama ke sumber daya yang berbeda untuk menunjukkan bahwa sumber daya tersebut terkait. Misalnya, Anda dapat menetapkan tanda seperti `Website=example.com` ke CA Anda, penyeimbang beban Elastic Load Balancing, dan sumber daya terkait lainnya. Untuk informasi selengkapnya tentang menandai AWS sumber daya, lihat Menandai Sumber Daya Amazon EC2 Anda di Panduan Pengguna Amazon EC2.

Pembatasan dasar berikut berlaku untuk AWS Private CA tag:

- Jumlah maksimum tanda per CA privat adalah 50.
- Panjang maksimal kunci tanda adalah 128 karakter.
- Panjang maksimal nilai tanda adalah 256 karakter.
- Kunci dan nilai tanda dapat berisi karakter berikut: A-Z, a-z, and .:+=@_%-(tanda hubung).
- Kunci dan nilai tag peka huruf besar dan kecil.
- Awalan `aws:` dan `rds:` dicadangkan untuk digunakan AWS ; Anda tidak dapat menambahkan, mengedit, atau menghapus tanda yang kuncinya dimulai dengan `aws:` atau `rds:`. Tag default yang dimulai dengan `aws:` dan `rds:` tidak dihitung terhadap tags-per-resource kuota Anda.
- Jika Anda berencana untuk menggunakan skema penandaan di beberapa layanan dan sumber daya, ingatlah bahwa layanan lain mungkin memiliki batasan berbeda untuk karakter yang diizinkan. Baca dokumentasi ini untuk layanan tersebut.
- AWS Private CA tag tidak tersedia untuk digunakan di Resource Groups dan Tag Editor di AWS Management Console.

Anda dapat menandai CA pribadi dari AWS Private CA Konsol, AWS Command Line Interface (AWS CLI), atau AWS Private CA API.

Untuk menandai CA privat (konsol)

1. Masuk ke AWS akun Anda dan buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home.
2. Pada halaman Otoritas sertifikat pribadi, pilih CA pribadi Anda dari daftar.
3. Di area detail di bawah daftar, pilih tab Tag. Daftar tag yang ada ditampilkan.
4. Pilih Kelola tanda.
5. Pilih Tambahkan tag baru.
6. Ketik pasangan kunci dan nilai.
7. Pilih Simpan.

Untuk menandai CA privat (AWS CLI)

Gunakan perintah tag-certificate-authority untuk menambahkan tanda ke CA privat Anda.

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

Gunakan perintah list-tag guna membuat daftar tanda untuk CA privat.

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

Gunakan perintah untag-certificate-authority untuk menghapus tanda dari CA privat.

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

Memperbarui CA privat Anda

Anda dapat memperbarui status CA pribadi atau mengubah konfigurasi pencabutan setelah membuatnya. Topik ini memberikan detail tentang status CA dan siklus hidup CA, bersama dengan contoh pembaruan konsol dan CLI ke CA.

Memperbarui status CA

Status CA yang dikelola oleh AWS Private CA hasil dari tindakan pengguna atau, dalam beberapa kasus, dari tindakan layanan. Misalnya, status CA berubah saat kedaluwarsa. Opsi status yang tersedia untuk CA administrator bervariasi, bergantung pada status CA saat ini.

AWS Private CA dapat melaporkan nilai status berikut. Tabel menunjukkan kemampuan CA yang tersedia di setiap negara bagian.

> **Note**
>
> Untuk semua nilai status kecuali `DELETED` dan`FAILED`, Anda ditagih untuk CA.

Status	Sertifikat penerbitan	Validasi sertifikat dengan OCSP	Menghakan CRL	Hasilkan audit	Anda dapat memperbarui sertifikat CA	Sertifikat dapat dicabut	Anda ditagih untuk CA
`CREATING`CA sedang dibuat.	Tidak	Tidak	Tidak	Tidak	Tidak	Tidak	Ya
`PENDING_CERTIFICATE` — CA telah dibuat dan membutuhkan sertifikat untuk beroperasi. *	Tidak	Tidak	Tidak	Tidak	Tidak	Tidak	Ya
`ACTIVE`	Ya	Ya	Ya	Ya	Ya	Ya	Ya

Status	Sertifikat penerbitan	Validasi sertifikat dengan OCSP	Menghasilkan CRL	Hasilkan audit	Anda dapat memperbarui sertifikat CA	Sertifikat dapat dicabut	Anda ditagih untuk CA
`DISABLED`— Anda telah menonaktifkan CA secara manual.	Tidak	Ya	Ya	Ya	Tidak	Ya	Ya
`EXPIRED`— Sertifikat CA telah kedaluwarsa. **	Tidak	Tidak	Tidak	Tidak	Ya	Tidak	Ya
`FAILED`	`CreateCertificateAuthority` Tindakan itu gagal. Hal ini dapat terjadi karena pemadaman jaringan, AWS kegagalan backend, atau kesalahan lainnya. CA yang gagal tidak dapat dipulihkan. Hapus CA dan buat yang baru.						Tidak
`DELETED`	CA Anda berada dalam periode pemulihan, yang dapat memiliki panjang 7-30 hari. Setelah periode ini, CA akan dihapus secara permanen. • Jika Anda memanggil API `RestoreCertificateAuthority` pada CA dengan status `DELETED` dan sertifikat yang kedaluwarsa, CA akan diatur ke `EXPIRED`. • Untuk informasi lebih lanjut tentang penghapusan CA, lihat Menghapus CA privat Anda.						Tidak

* Untuk menyelesaikan aktivasi, Anda perlu membuat CSR, mendapatkan sertifikat CA yang ditandatangani dari CA, dan mengimpor sertifikat ke dalam AWS Private CA. CSR dapat dikirimkan ke CA baru Anda (untuk penandatanganan sendiri), atau ke root lokal atau CA bawahan. Untuk informasi selengkapnya, lihat Membuat dan menginstal sertifikat CA.

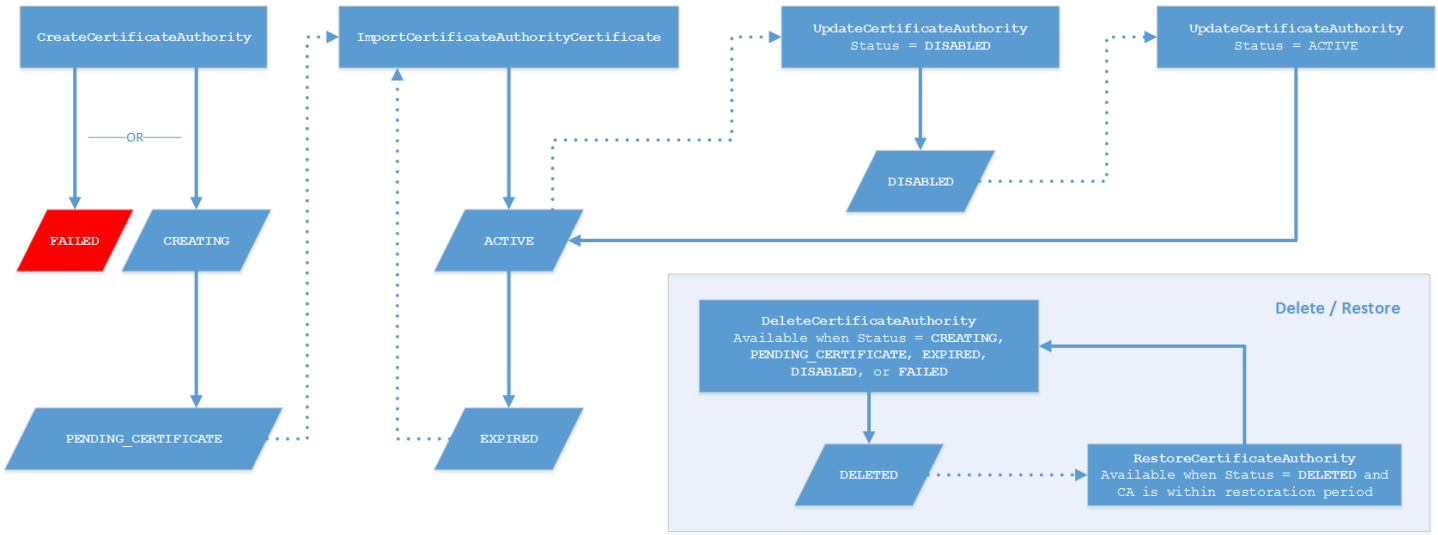
** Anda tidak dapat secara langsung mengubah status CA yang kedaluwarsa. Jika Anda mengimpor sertifikat baru untuk CA, AWS Private CA setel ulang statusnya `ACTIVE` kecuali telah disetel ke `DISABLED` sebelum sertifikat kedaluwarsa.

Pertimbangan tambahan tentang sertifikat CA yang kedaluwarsa:

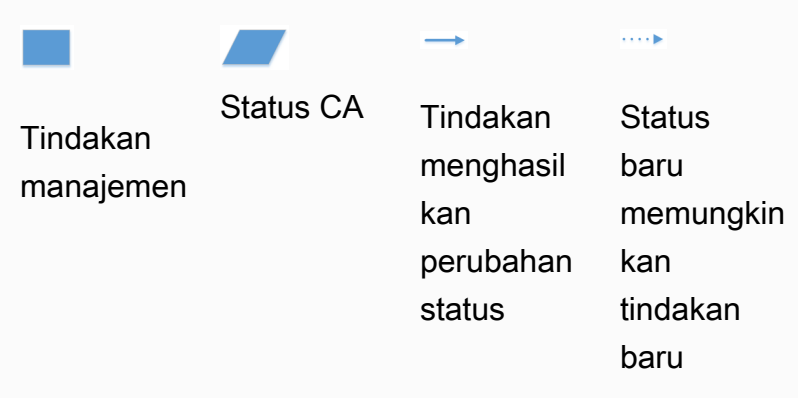
- Sertifikat CA tidak diperbarui secara otomatis. Untuk informasi tentang mengotomatisasi pembaruan melalui AWS Certificate Manager, lihat. Menetapkan izin perpanjangan sertifikat untuk ACM
- Jika Anda mencoba untuk menerbitkan sertifikat baru dengan CA yang kedaluwarsa, API `IssueCertificate` mengembalikan `InvalidStateException`. CA akar yang kedaluwarsa harus menandatangani sendiri sertifikat CA akar yang baru sebelum dapat menerbitkan sertifikat bawahan baru.
- API `The ListCertificateAuthorities` dan `DescribeCertificateAuthority` mengembalikan status `EXPIRED` jika sertifikat CA kedaluwarsa, terlepas dari apakah CA status diatur ke `ACTIVE` atau `DISABLED`. Namun, jika CA kedaluwarsa telah ditetapkan ke `DELETED`, status yang dikembalikan adalah `DELETED`.
- API `UpdateCertificateAuthority` tidak dapat memperbarui status CA kedaluwarsa.
- `RevokeCertificate`API tidak dapat digunakan untuk mencabut sertifikat yang kedaluwarsa, termasuk sertifikat CA.

Status CA dan siklus hidup CA

Diagram berikut menggambarkan CA siklus hidup sebagai interaksi tindakan manajemen dengan status CA.



Kunci diagram



Di bagian atas diagram, tindakan manajemen diterapkan melalui AWS Private CA konsol, CLI, atau API. Tindakan ini membawa CA melalui proses pembuatan, aktivasi, kedaluwarsa, dan perpanjangan. Status CA berubah sebagai respons (seperti yang ditunjukkan oleh garis solid) untuk tindakan manual atau pembaruan otomatis. Di sebagian besar kasus, status baru mengarah ke potensi tindakan yang baru (ditunjukkan oleh garis putus-putus) yang dapat diterapkan oleh administrator CA. Inset kanan bawah menunjukkan kemungkinan nilai status yang mengizinkan penghapusan dan pengembalian tindakan.

Memperbarui CA (konsol)

Prosedur berikut menunjukkan cara memperbarui konfigurasi CA yang ada menggunakan. AWS Management Console

Perbarui status CA (konsol)

Dalam contoh ini, status CA yang diaktifkan diubah menjadi dinonaktifkan.

Untuk memperbarui status CA

1. Masuk ke AWS akun Anda dan buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home
2. Pada halaman Private Certificate Authority, pilih CA pribadi yang saat ini aktif dari daftar.
3. Pada menu Tindakan, pilih Nonaktifkan untuk menonaktifkan CA pribadi.

Memperbarui konfigurasi pencabutan CA (konsol)

Anda dapat memperbarui konfigurasi pencabutan untuk CA pribadi Anda, misalnya, dengan menambahkan atau menghapus dukungan OCSP atau CRL, atau dengan memodifikasi pengaturannya.

> **Note**
>
> Perubahan pada konfigurasi pencabutan CA tidak memengaruhi sertifikat yang sudah diterbitkan. Agar pencabutan terkelola berfungsi, sertifikat yang lebih lama harus diterbitkan kembali.

Untuk OCSP, Anda mengubah pengaturan berikut:

- Aktifkan atau nonaktifkan OCSP.
- Mengaktifkan atau menonaktifkan nama domain OCSP yang sepenuhnya memenuhi syarat (FQDN) kustom.
- Ubah FQDN.

Untuk CRL, Anda dapat mengubah salah satu pengaturan berikut:

- Apakah CA privat menghasilkan daftar pencabutan sertifikat (CRL)
- Jumlah hari sebelum CRL kedaluwarsa. Perhatikan bahwa AWS Private CA mulai mencoba meregenerasi CRL pada ½ jumlah hari yang Anda tentukan.
- Nama bucket Amazon S3 tempat CRL Anda disimpan.

- Alias untuk menyembunyikan nama bucket Amazon S3 Anda dari tampilan publik.

> **Important**
>
> Mengubah salah satu parameter sebelumnya dapat memiliki efek negatif. Contohnya termasuk menonaktifkan pembuatan CRL, mengubah masa berlaku, atau mengubah bucket S3 setelah Anda menempatkan CA pribadi Anda dalam produksi. Perubahan tersebut dapat merusak sertifikat yang ada yang bergantung pada CRL dan konfigurasi CRL saat ini. Mengubah alias dapat dilakukan dengan aman, selama alias lama tetap terhubung ke bucket yang benar.

Untuk memperbarui pengaturan pencabutan

1. Masuk ke AWS akun Anda dan buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home.
2. Pada halaman Otoritas sertifikat pribadi, pilih CA pribadi dari daftar. Ini membuka panel detail untuk CA.
3. Pilih tab konfigurasi Pencabutan, lalu pilih Edit.
4. Di bawah opsi pencabutan Sertifikat, dua opsi ditampilkan:

   - Aktifkan distribusi CRL
   - Nyalakan OCSP

   Anda dapat mengonfigurasi salah satu, keduanya, atau kedua mekanisme pencabutan ini untuk CA Anda. Meskipun opsional, pencabutan terkelola direkomendasikan sebagai praktik terbaik. Sebelum menyelesaikan langkah ini, lihat Menyiapkan metode pencabutan sertifikat informasi tentang keunggulan masing-masing metode, pengaturan awal yang mungkin diperlukan, dan fitur pencabutan tambahan.

Untuk mengkonfigurasi CRL

1. Pilih Aktifkan distribusi CRL.
2. Untuk membuat bucket Amazon S3 untuk entri CRL Anda, pilih Buat bucket S3 baru. Berikan nama ember yang unik. (Anda tidak perlu menyertakan jalur ke bucket.) Jika tidak, biarkan opsi ini tidak dipilih dan pilih bucket yang ada dari daftar nama bucket S3.

Jika Anda membuat bucket baru, AWS Private CA buat dan lampirkan kebijakan akses yang diperlukan padanya. Jika Anda memutuskan untuk menggunakan bucket yang sudah ada, Anda harus melampirkan kebijakan akses itu sebelum Anda dapat mulai membuat CRL. Gunakan salah satu pola kebijakan yang dijelaskan dalam Kebijakan akses untuk CRL di Amazon S3 . Untuk informasi tentang melampirkan kebijakan, lihat Menambahkan kebijakan bucket menggunakan konsol Amazon S3.

> **Note**
>
> Saat Anda menggunakan AWS Private CA konsol, upaya untuk membuat CA gagal jika kedua kondisi berikut berlaku:
>
> - Anda menerapkan pengaturan Blokir Akses Publik di bucket atau akun Amazon S3 Anda.
> - Anda diminta AWS Private CA untuk membuat bucket Amazon S3 secara otomatis.
>
> Dalam situasi ini, konsol mencoba, secara default, untuk membuat bucket yang dapat diakses publik, dan Amazon S3 menolak tindakan ini. Periksa pengaturan Amazon S3 Anda jika hal ini terjadi. Untuk informasi lebih lanjut, lihat Memblokir akses publik ke penyimpanan Amazon S3.

3. Perluas Lanjutan untuk opsi konfigurasi tambahan.

   - Tambahkan Nama CRL Kustom untuk membuat alias untuk bucket Amazon S3. Nama ini ada dalam sertifikat yang diterbitkan oleh CA di ekstensi “CRL Distribution Points” yang ditentukan oleh RFC 5280.
   - Ketik jumlah hari CRL Anda akan tetap valid. Nilai default-nya adalah 7 hari. Untuk CRL online, masa berlaku 2-7 hari adalah umum. AWS Private CA mencoba meregenerasi CRL pada titik tengah periode yang ditentukan.

4. Pilih Simpan perubahan setelah selesai.

Untuk mengkonfigurasi OCSP

1. Pada halaman Pencabutan sertifikat, pilih Aktifkan OCSP.
2. (Opsional) Di bidang titik akhir OCSP Kustom, berikan nama domain yang memenuhi syarat (FQDN) untuk titik akhir OCSP Anda.

Saat Anda memberikan FQDN di bidang ini, AWS Private CA masukkan FQDN ke ekstensi Akses Informasi Otoritas dari setiap sertifikat yang dikeluarkan sebagai pengganti URL default untuk responden OCSP. AWS Ketika titik akhir menerima sertifikat yang berisi FQDN kustom, ia menanyakan alamat tersebut untuk respons OCSP. Agar mekanisme ini berfungsi, Anda perlu mengambil dua tindakan tambahan:

- Gunakan server proxy untuk meneruskan lalu lintas yang tiba di FQDN kustom Anda ke responder OCSP. AWS
- Tambahkan catatan CNAME yang sesuai ke database DNS Anda.

> **Tip**
>
> Untuk informasi selengkapnya tentang penerapan solusi OCSP lengkap menggunakan CNAME kustom, lihat. Mengkonfigurasi URL Kustom untuk AWS Private CA OCSP

Misalnya, berikut adalah catatan CNAME untuk OCSP yang disesuaikan seperti yang akan muncul di Amazon Route 53.

Nama catatan	Tipe	Kebijakan perutean	Diferensiator	Nilai/Rutekan lalu lintas ke
alternatif.example.com	CNAME	Sederhana	-	proxy.example.com

> **Note**
>
> Nilai CNAME tidak boleh menyertakan awalan protokol seperti “http://” atau “https://”.

3. Pilih Simpan perubahan setelah selesai.

Memperbarui CA (CLI)

Prosedur berikut menunjukkan cara memperbarui status dan konfigurasi pencabutan CA yang ada menggunakan. AWS CLI

> **Note**
>
> Perubahan pada konfigurasi pencabutan CA tidak memengaruhi sertifikat yang sudah diterbitkan. Agar pencabutan terkelola berfungsi, sertifikat yang lebih lama harus diterbitkan kembali.

Untuk memperbarui status CA (AWS CLI) pribadi Anda

Gunakan perintah update-certificate-authority.

Ini berguna ketika Anda memiliki CA yang sudah ada dengan status `DISABLED` yang ingin Anda atur`ACTIVE`. Untuk memulai, konfirmasikan status awal CA dengan perintah berikut.

```
$ 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
```

Ini menghasilkan output yang mirip dengan yang berikut ini.

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

Perintah berikut menetapkan status CA pribadi keACTIVE. Ini hanya mungkin jika sertifikat yang valid diinstal pada CA.

```
$ aws acm-pca update-certificate-authority \
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
      --status "ACTIVE"
```

Periksa status baru 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
```

Status sekarang muncul sebagaiACTIVE.

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

Dalam beberapa kasus, Anda mungkin memiliki CA aktif tanpa mekanisme pencabutan yang dikonfigurasi. Jika Anda ingin mulai menggunakan daftar pencabutan sertifikat (CRL), gunakan prosedur berikut.

Untuk menambahkan CRL ke CA ()AWS CLI yang ada

1. Gunakan perintah berikut untuk memeriksa status CA saat ini.

```
$ 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
```

Output mengkonfirmasi bahwa CA memiliki status `ACTIVE` tetapi tidak dikonfigurasi untuk menggunakan CRL.

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

2. Buat dan simpan file dengan nama seperti `revoke_config.txt` untuk menentukan parameter konfigurasi CRL Anda.

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

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

> **Note**
>
> Saat memperbarui CA pengesahan perangkat Matter untuk mengaktifkan CRL, Anda harus mengonfigurasinya untuk menghilangkan ekstensi CDP dari sertifikat yang dikeluarkan untuk membantu menyesuaikan dengan standar Matter saat ini. Untuk melakukan ini, tentukan parameter konfigurasi CRL Anda seperti yang diilustrasikan di bawah ini:
>
> ```
> {
>    "CrlConfiguration":{
>       "Enabled": true,
>       "ExpirationInDays": 7,
>       "S3BucketName": "bucket-name"
>       "CrlDistributionPointExtensionConfiguration":{
>          "OmitExtension": true
>       }
>    }
> }
> ```

3. Gunakan perintah update-certificate-authority dan file konfigurasi revocation untuk memperbarui CA.

```
$ aws acm-pca update-certificate-authority \
      --certificate-authority-arn arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566 \
      --revocation-configuration file://revoke_config.txt
```

4. Sekali lagi periksa status 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
```

Output mengkonfirmasi bahwa CA sekarang dikonfigurasi untuk menggunakan CRL.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-08T14:36:26.449000-08:00",
        "LastStateChangeAt": "2021-03-08T14:50:52.224000-08:00",
        "Type": "ROOT",
        "Serial": "serial_numbner",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T13:46:50-08:00",
        "NotAfter": "2022-03-08T14:46:50-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "S3BucketName": "DOC-EXAMPLE-BUCKET1",
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

Dalam beberapa kasus, Anda mungkin ingin menambahkan dukungan pencabutan OCSP alih-alih mengaktifkan CRL seperti pada prosedur sebelumnya. Dalam hal ini, gunakan langkah-langkah berikut.

Untuk menambahkan dukungan OCSP ke CA ()AWS CLI yang ada

1. Buat dan simpan file dengan nama seperti `revoke_config.txt` untuk menentukan parameter OCSP Anda.

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

2. Gunakan perintah update-certificate-authority dan file konfigurasi revocation untuk memperbarui CA.

```
$ aws acm-pca update-certificate-authority \
      --certificate-authority-arn arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566 \
      --revocation-configuration file://revoke_config.txt
```

3. Sekali lagi periksa status CA.

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

Output menegaskan bahwa CA sekarang dikonfigurasi untuk menggunakan OCSP.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-08T14:36:26.449000-08:00",
        "LastStateChangeAt": "2021-03-08T14:50:52.224000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T13:46:50-08:00",
        "NotAfter": "2022-03-08T14:46:50-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
```

```
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": false
            },
            "OcspConfiguration": {
                "Enabled": true
            }
        }
    }
}
```

> **Note**
>
> Anda juga dapat mengonfigurasi dukungan CRL dan OCSP pada CA.

Menghapus CA privat Anda

Anda dapat menghapus CA pribadi dari AWS Management Console atau secara AWS CLI permanen. Anda mungkin ingin menghapus satu, misalnya, untuk menggantinya dengan CA baru yang memiliki kunci privat baru. Untuk menghapus CA dengan aman, ikuti langkah berikut:

1. Buat CA pengganti.
2. Setelah CA privat baru dalam produksi, nonaktifkan yang lama, tetapi jangan langsung menghapusnya.
3. Menjaga CA lama dinonaktifkan sampai semua sertifikat yang diterbitkan olehnya telah kedaluwarsa.
4. Hapus CA lama.

AWS Private CA tidak memeriksa bahwa semua sertifikat yang dikeluarkan telah kedaluwarsa sebelum memproses permintaan penghapusan. Anda dapat menghasilkan laporan audit untuk menentukan certifikat mana yang telah kedaluwarsa. Saat CA dinonaktifkan, Anda dapat mencabut sertifikat, tetapi Anda tidak dapat menerbitkan yang baru.

Jika Anda harus menghapus CA privat sebelum semua sertifikat yang diterbitkan kedaluwarsa, sebaiknya Anda juga mencabut sertifikat CA tersebut. Sertifikat CA akan dicantumkan dalam CRL CA induk, dan CA privat tidak akan dipercaya oleh klien.

> **Important**
>
> CA privat dapat dihapus jika berada di negara bagian `PENDING_CERTIFICATE`, `CREATING`, `EXPIRED`, `DISABLED`, atau `FAILED`. Untuk menghapus CA di negara bagian `ACTIVE`, Anda harus terlebih dahulu menonaktifkannya, atau menghapus permintaan hasil dalam pengecualian. Jika Anda menghapus CA pribadi di `DISABLED` negara bagian `PENDING_CERTIFICATE` atau, Anda dapat mengatur panjang periode restorasi dari 7-30 hari, dengan 30 menjadi default. Selama itu, status diatur ke `DELETED` dan CA dapat dipulihkan. CA pribadi yang dihapus saat berada di `FAILED` negara bagian `CREATING` atau tidak memiliki periode pemulihan yang ditetapkan dan tidak dapat dipulihkan. Untuk informasi selengkapnya, lihat Memulihkan CA pribadi.
> Anda tidak akan ditagihkan CA privat setelah dihapus. Namun, jika CA yang dihapus dipulihkan, masa antara penghapusan dan pemulihan akan ditagihkan. Untuk informasi selengkapnya, lihat Harga.

Untuk menghapus CA privat (konsol)

1. Masuk ke AWS akun Anda dan buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home.
2. Pada halaman Otoritas sertifikat pribadi, pilih CA pribadi Anda dari daftar.
3. Jika CA Anda berada di `ACTIVE` negara bagian, Anda harus menonaktifkannya terlebih dahulu. Pada menu Tindakan, pilih Nonaktif. Ketika diminta, pilih Saya mengerti risikonya, lanjutkan.
4. Untuk CA yang tidak berada dalam `ACTIVE` status, pilih Tindakan, Hapus.
5. Jika CA Anda berada dalam`DISABLED`,`EXPIRED`, atau `PENDING_CERTIFICATE` status, halaman Hapus CA memungkinkan Anda menentukan periode pemulihan 7-30 hari. Jika CA pribadi Anda tidak berada di salah satu negara bagian ini, CA tidak dapat dipulihkan nanti dan penghapusan bersifat permanen.

6. Pilih Hapus.
7. Jika Anda yakin ingin menghapus CA privat, pilih Hapus secara permanen saat diminta. Status CA privat berubah menjadi `DELETED`. Namun, Anda dapat memulihkan CA privat sebelum akhir masa pemulihan. Untuk memeriksa periode pemulihan CA pribadi di `DELETED` negara bagian, hubungi operasi DescribeCerticateOtoritas atau ListCertificate Otoritas API.

Untuk menghapus CA privat (AWS CLI)

Gunakan perintah delete-certificate-authority untuk menghapus CA privat.

```
$ 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
```

Memulihkan CA pribadi

Anda dapat memulihkan CA privat yang telah dihapus selama CA tetap dalam periode pemulihan yang Anda tentukan saat dihapus. Periode restorasi adalah 7-30 hari. Pada akhir periode tersebut, CA privat akan dihapus secara permanen. Untuk informasi lebih lanjut, lihat Menghapus CA privat Anda. Anda tidak dapat memulihkan CA privat yang telah dihapus secara permanen.

> **Note**
>
> Anda tidak akan ditagihkan CA privat setelah dihapus. Namun, jika CA yang dihapus dipulihkan, masa antara penghapusan dan pemulihan akan ditagihkan. Untuk informasi lebih lanjut, lihat Harga.

Memulihkan CA privat (konsol)

Anda dapat menggunakan AWS Management Console untuk memulihkan CA pribadi.

Untuk memulihkan CA privat (konsol)

1. Masuk ke AWS akun Anda dan buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home.
2. Pada halaman Otoritas sertifikat pribadi, pilih CA pribadi yang dihapus dari daftar.

3. Pada menu Tindakan, pilih Pemulihan.
4. Pada halaman Restore CA, pilih Restore again.
5. Jika berhasil, status CA privat diatur ke negara bagian pra-penghapusan. Pilih Tindakan, Aktifkan, dan Aktifkan lagi untuk mengubah statusnya menjadi`ACTIVE`. Jika CA privat berada di negara bagian `PENDING_CERTIFICATE` pada saat penghapusan, Anda harus mengimpor sertifikat CA ke CA privat sebelum Anda dapat mengaktifkannya.

Memulihkan CA privat (AWS CLI)

Gunakan perintah restore-certificate-authority untuk memulihkan CA privat yang dihapus yang berada di negara bagian `DELETED`. Langkah-langkah berikut membahas seluruh proses yang diperlukan untuk menghapus, memulihkan, lalu mengaktifkan kembali CA privat.

Untuk menghapus, memulihkan, dan mengaktifkan kembali CA privat (AWS CLI)

1. Hapus CA privat.

   Jalankan perintah delete-certificate-authority untuk menghapus CA privat. Jika status CA pribadi adalah `DISABLED` atau`PENDING_CERTIFICATE`, Anda dapat mengatur `--permanent-deletion-time-in-days` parameter untuk menentukan periode pemulihan CA pribadi dari 7 -30 hari. Jika Anda tidak menentukan masa pemulihan, defaultnya adalah 30 hari. Jika berhasil, perintah ini menetapkan status CA privat ke `DELETED`.

   > **Note**
   >
   > Agar dapat dipulihkan, status CA privat pada saat penghapusan harus `DISABLED` atau `PENDING_CERTIFICATE`.

   ```
   $ aws acm-pca delete-certificate-authority \
        --certificate-authority-arn arn:aws:acm-pca:region:account:certificate-authority/CA_ID \
        --permanent-deletion-time-in-days 16
   ```

2. Memulihkan CA privat.

   Jalankan perintah restore-certificate-authority untuk memulihkan CA privat. Anda harus menjalankan perintah sebelum masa pemulihan yang Anda atir dengan kedaluwarsa perintah

delete-certificate-authority. Jika berhasil, perintah mengatur status CA privat ke status penghapusan sebelumnya.

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

3. Membuat ACTIVE CA privat.

   Jalankan perintah update-certificate-authority untuk mengubah status CA privat menjadi ACTIVE.

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

Administrasi sertifikat

Setelah Anda membuat dan mengaktifkan Private Certificate Authority (CA) dan mengonfigurasi akses ke sana, Anda atau pengguna yang sah dapat melakukan tugas yang dibahas di bagian ini. Jika Anda belum menyiapkan kebijakan (IAM) AWS Identity and Access Management untuk CA, Anda dapat pelajari selengkapnya tentang mengonfigurasinya di bagian Identity and Access Management pada panduan ini. Untuk informasi tentang mengkonfigurasi akses CA dalam skenario akun tunggal dan lintas akun, lihat Mengontrol akses ke CA privat.

Topik

Menerbitkan sertifikat entitas akhir pribadi

Dengan adanya CA pribadi, Anda dapat meminta sertifikat entitas akhir pribadi dari AWS Certificate Manager (ACM) atau. AWS Private CA Kemampuan kedua layanan dibandingkan dalam tabel berikut.

Kemampuan	ACM	AWS Private CA
Menerbitkan sertifikat entitas akhir	✓ (menggunakan RequestCertificateatau konsol)	✓ (menggunakan IssueCertificate)
Asosiasi dengan penyeimbang beban dan layanan yang menghadap ke internet AWS	✓	Tidak didukung
Perpanjangan sertifikat terkelola	✓	Didukung secara tidak langsung melalui ACM

Kemampuan	ACM	AWS Private CA
Dukungan konsol	✓	Tidak didukung
Dukungan API	✓	✓
Dukungan CLI	✓	✓

Saat AWS Private CA membuat sertifikat, ia mengikuti template yang menentukan jenis sertifikat dan panjang jalur. Jika tidak ada template ARN yang diberikan ke API atau pernyataan CLI yang membuat sertifikat, template EndEntityCertificate/V1 diterapkan secara default. Untuk informasi selengkapnya tentang templat sertifikat yang tersedia, lihat Memahami templat sertifikat.

Sementara sertifikat ACM dirancang di sekitar kepercayaan publik, AWS Private CA melayani kebutuhan PKI pribadi Anda. Akibatnya, Anda dapat mengonfigurasi sertifikat menggunakan AWS Private CA API dan CLI dengan cara yang tidak diizinkan oleh ACM. Sumber daya yang dimaksud meliputi:

- Membuat sertifikat dengan nama Subjek.
- Menggunakan salah satu algoritma kunci privat yang didukung dan panjang kunci.
- Menggunakan salah satu algoritma penandatanganan yang didukung.
- Menentukan periode validitas apa pun untuk CA privat dan sertifikat privat Anda.

Setelah membuat sertifikat TLS pribadi menggunakanAWS Private CA, Anda dapat mengimpornya ke ACM dan menggunakannya dengan layanan yang didukungAWS.

> **Note**
>
> Sertifikat yang dibuat dengan prosedur di bawah ini, menggunakan issue-certificate perintah, atau dengan tindakan IssueCertificateAPI, tidak dapat langsung diekspor untuk digunakan di luarAWS. Namun, Anda dapat menggunakan CA pribadi Anda untuk menandatangani sertifikat yang dikeluarkan melalui ACM, dan sertifikat tersebut dapat diekspor bersama dengan kunci rahasianya. Untuk informasi selengkapnya, lihat Meminta sertifikat pribadi dan Mengekspor sertifikat pribadi di Panduan Pengguna ACM.

Menerbitkan sertifikat standar () AWS CLI

Anda dapat menggunakan sertifikat penerbitan perintah AWS Private CA CLI atau tindakan API IssueCertificateuntuk meminta sertifikat entitas akhir. Perintah ini memerlukan Amazon Resource Name (ARN) dari CA privat yang ingin Anda gunakan untuk menerbitkan sertifikat. Anda juga harus membuat permintaan penandatanganan sertifikat (CSR) menggunakan program seperti OpenSSL.

Jika Anda menggunakan AWS Private CA API atau AWS CLI menerbitkan sertifikat pribadi, sertifikat tidak dikelola, artinya Anda tidak dapat menggunakan konsol ACM, ACM CLI, atau ACM API untuk melihat atau mengekspornya, dan sertifikat tidak diperpanjang secara otomatis. Namun, Anda dapat menggunakan perintah get-certificate PCA untuk mengambil detail sertifikat, dan jika Anda memiliki CA, Anda dapat membuat laporan audit.

Pertimbangan saat membuat sertifikat

- Sesuai dengan RFC 5280, panjang nama domain (secara teknis, Nama Umum) yang Anda berikan tidak boleh melebihi 64 oktet (karakter), termasuk periode. Untuk menambahkan nama domain yang lebih panjang, tentukan di bidang Nama Alternatif Subjek, yang mendukung panjang nama hingga 253 oktet.
- Jika Anda menggunakan AWS CLI versi 1.6.3 atau yang lebih baru, gunakan awalan `fileb://` saat menentukan file input yang disandikan base64 seperti CSR. Ini memastikan bahwa AWS Private CA mem-parsing data dengan benar.

Perintah OpenSSL berikut menghasilkan CSR dan kunci pribadi untuk sertifikat:

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

Anda dapat memeriksa isi CSR sebagai berikut:

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

Output yang dihasilkan harus menyerupai contoh singkat berikut:

```
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
```

Perintah berikut membuat sertifikat. Karena tidak ada templat yang ditentukan, sertifikat entitas akhir dasar dikeluarkan secara default.

```
$ 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"
```

ARN dari sertifikat yang diterbitkan dikembalikan:

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

> **Note**
>
> AWS Private CAsegera mengembalikan ARN dengan nomor seri ketika menerima perintah. issue-certificate Namun, pemrosesan sertifikat terjadi secara asinkron dan masih bisa gagal. Jika ini terjadi, get-certificate perintah yang menggunakan ARN baru juga akan gagal.

Menerbitkan sertifikat dengan nama subjek kustom menggunakan template ApiPassThrough

Dalam contoh ini, sertifikat dikeluarkan yang berisi elemen nama subjek yang disesuaikan. Selain memasok CSR seperti yang ada di[Menerbitkan sertifikat standar () AWS CLI](), Anda meneruskan dua argumen tambahan ke issue-certificate perintah: ARN dari template APIPassThrough, dan file konfigurasi JSON yang menentukan atribut kustom dan pengidentifikasi objek (OID) mereka. Anda tidak dapat menggunakan `StandardAttributes` bersama dengan`CustomAttributes`. Namun, Anda dapat melewati OID standar sebagai bagian dari. `CustomAttributes` Nama subjek default OID tercantum dalam tabel berikut (informasi dari RFC 4519 dan database referensi Global OID):

Nama subjek	Singkatan	ID Objek
Nama negara	c	2.5.4.6
commonName	cn	2.5.4.3
DNQualifier [kualifikasi nama terhormat]		2.5.4.46
GenerationQualifier		2.5.4.44
givenName		2.5.4.42
inisial		2.5.4.43
lokalitas	l	2.5.4.7
Nama Organisasi	o	2.5.4.10
organizationalUnitName	ou	2.5.4.11

Nama subjek	Singkatan	ID Objek
nama samaran		2.5.4.65
SerialNumber		2.5.4.5
st [negara]		2.5.4.8
nama keluarga	sn	2.5.4.4
title		2.5.4.12
DomainComponent	dc	0.9.2342.19200300.100.1.25
userid		0.9.2342.19200300.100.1.1

File konfigurasi sampel `api_passthrough_config.txt` berisi kode berikut:

```
{
  "Subject": {
    "CustomAttributes": [
      {
        "ObjectIdentifier": "2.5.4.6",
        "Value": "US"
      },
      {
        "ObjectIdentifier": "1.3.6.1.4.1.37244.1.1",
        "Value": "BCDABCDA12341234"
      },
      {
        "ObjectIdentifier": "1.3.6.1.4.1.37244.1.5",
        "Value": "CDABCDAB12341234"
      }
    ]
  }
}
```

Gunakan perintah berikut untuk mengeluarkan sertifikat:

```
$ aws acm-pca issue-certificate \
      --validity Type=DAYS,Value=10
```

```
      --signing-algorithm "SHA256WITHRSA" \
      --csr fileb://csr.pem \
      --api-passthrough file://api_passthrough_config.txt \
      --template-arn arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APIPassthrough/V1 \
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
```

ARN dari sertifikat yang diterbitkan dikembalikan:

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

Ambil sertifikat secara lokal sebagai berikut:

```
$ 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
```

Anda dapat memeriksa konten sertifikat menggunakan OpenSSL:

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

> **Note**
>
> Dimungkinkan juga untuk membuat CA pribadi yang meneruskan atribut khusus ke setiap sertifikat yang dikeluarkannya.

Mengeluarkan sertifikat dengan ekstensi kustom menggunakan template ApiPassThrough

Dalam contoh ini, sertifikat dikeluarkan yang berisi ekstensi yang disesuaikan. Untuk ini, Anda perlu meneruskan tiga argumen ke issue-certificate perintah: ARN dari template ApiPassThrough,

dan file konfigurasi JSON yang menentukan ekstensi kustom, dan CSR seperti yang ditunjukkan. Menerbitkan sertifikat standar () AWS CLI

File konfigurasi sampel `api_passthrough_config.txt` berisi kode berikut:

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

Sertifikat yang disesuaikan dikeluarkan sebagai berikut:

```
$ aws acm-pca issue-certificate \
      --validity Type=DAYS,Value=10
      --signing-algorithm "SHA256WITHRSA" \
      --csr fileb://csr.pem \
      --api-passthrough file://api_passthrough_config.txt \
      --template-arn arn:aws:acm-pca:::template/EndEntityCertificate_APIPassthrough/V1
 \
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566
```

ARN dari sertifikat yang diterbitkan dikembalikan:

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

Ambil sertifikat secara lokal sebagai berikut:

```
$ 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
```

Anda dapat memeriksa konten sertifikat menggunakan OpenSSL:

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

Mengambil sertifikat privat

Anda dapat menggunakan AWS Private CA API dan AWS CLI mengeluarkan sertifikat pribadi. Jika ya, Anda dapat menggunakan AWS Private CA API AWS CLI atau untuk mengambil sertifikat tersebut. Jika Anda menggunakan ACM untuk membuat CA privat dan meminta sertifikat, Anda harus menggunakan ACM untuk mengekspor sertifikat dan kunci privat terenkripsi. Untuk informasi selengkapnya, lihat Mengekspor sertifikat pribadi.

Untuk mengambil sertifikat entitas akhir

Gunakan perintah get-certificate untuk mengambil sertifikat entitas akhir privat. Anda juga dapat menggunakan operasi GetCertificateAPI. Kami merekomendasikan memformat output dengan jq, parser sed-like.

> **Note**
>
> Jika Anda ingin mencabut sertifikat, Anda dapat menggunakan perintah get-certificate untuk mengambil nomor seri dalam format heksadesimal. Anda juga dapat membuat laporan audit untuk mengambil nomor seri heks. Untuk informasi selengkapnya, lihat Menggunakan laporan audit dengan CA privat Anda.

```
$ 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'
```

Perintah ini mengeluarkan sertifikat dan rantai sertifikat dalam format standar berikut.

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

Untuk mengambil sertifikat CA

Anda dapat menggunakan AWS Private CA API dan AWS CLI mengambil sertifikat otoritas sertifikat (CA) untuk CA pribadi Anda. Jalankan perintah get-certificate-authority-certificate. Anda juga dapat menghubungi operasi GetCertificateAuthorityCertificate. Kami merekomendasikan memformat output dengan jq, parser sed-like.

```
$ 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'
```

Perintah ini mengeluarkan sertifikat CA dalam format standar berikut.

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

Daftar sertifikat privat

Untuk membuat daftar sertifikat privat Anda, buat laporan audit, ambil dari bucket S3-nya, dan parser konten laporan sesuai kebutuhan. Untuk informasi tentang membuat laporan AWS Private CA audit, lihatMenggunakan laporan audit dengan CA privat Anda. Untuk informasi tentang mengambil objek dari bucket S3, lihat Mengunduh objek di Panduan Pengguna Amazon Simple Storage Service.

Contoh berikut mengilustrasikan pendekatan untuk membuat laporan audit dan melakukan parser untuk data yang berguna. Hasil diformat dalam JSON, dan data difilter menggunakan jq, parser sed-like.

1. Buat laporan audit.

Perintah berikut menghasilkan laporan audit untuk CA tertentu.

```
$ 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
```

Ketika berhasil, perintah mengembalikan ID dan lokasi laporan audit baru.

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

2. Ambil dan format laporan audit.

Perintah ini mengambil laporan audit, menampilkan isinya dalam output standar, serta menyaring hasil untuk menampilkan hanya sertifikat yang diterbitkan pada atau setelah 01-12-2020.

```
$ aws s3api get-object \
     --region region \
     --bucket bucket_name \
     --key audit-report/CA_ID/audit_report_ID.json \
     /dev/stdout | jq '.[] | select(.issuedAt >= "2020-12-01")'
```

Item yang dikembalikan menyerupai berikut ini:

```
{
   "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. Simpan laporan audit secara lokal.

Jika Anda ingin melakukan beberapa kueri, akan lebih mudah untuk menyimpan laporan audit ke file lokal.

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

Filter yang sama seperti sebelumnya menghasilkan output yang sama:

```
$ 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. Kueri di dalam rentang tanggal

Anda dapat kueri untuk sertifikat yang diterbitkan dalam rentang tanggal sebagai berikut:

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

Konten yang difilter ditampilkan dalam output standar:

```
{
   "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. Cari sertifikat mengikuti templat yang ditentukan.

Perintah berikut memfilter konten laporan menggunakan templat ARN:

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

Output menampilkan catatan sertifikat yang cocok:

```
{
   "awsAccountId": "account",
   "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
   "serial": "serial_number",
   "subject": "CN=pca.alpha.root2.rsa2048sha256",
   "notBefore": "2020-11-06T19:15:46+0000",
   "notAfter": "9999-12-31T23:59:59+0000",
   "issuedAt": "2020-11-06T20:15:46+0000",
   "templateArn": "arn:aws:acm-pca:::template/RootCACertificate/V1"
```

```
}
```

6. Filter untuk sertifikat yang dicabut

Untuk menemukan semua sertifikat yang dicabut, gunakan perintah berikut:

```
$ cat my_local_audit_report.json | jq '.[] | select(.revokedAt != null)'
```

Sertifikat yang dicabut ditampilkan sebagai berikut:

```
{
   "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. Filter menggunakan ekspresi reguler.

Perintah berikut mencari nama subjek yang berisi string "leaf":

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

Catatan sertifikat yang cocok dikembalikan sebagai berikut:

```
{
   "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"
}
```

Mengekspor sertifikat pribadi dan kunci rahasianya

AWS Private CAtidak dapat langsung mengekspor sertifikat pribadi yang telah ditandatangani dan diterbitkan. Namun, Anda dapat menggunakan AWS Certificate Manager untuk mengekspor sertifikat tersebut bersama dengan kunci rahasia terenkripsi. Sertifikat ini kemudian sepenuhnya portabel untuk penyebaran di mana saja di PKI pribadi Anda. Untuk informasi selengkapnya, lihat Mengekspor sertifikat pribadi di Panduan AWS Certificate Manager Pengguna.

Sebagai manfaat tambahan, AWS Certificate Manager menyediakan perpanjangan terkelola untuk sertifikat pribadi yang dikeluarkan menggunakan konsol ACM, `RequestCertificate` tindakan ACM API, atau request-certificate perintah di bagian ACM. AWS CLI Untuk informasi selengkapnya tentang perpanjangan, lihat Memperbarui sertifikat di PKI pribadi.

Mencabut sertifikat privat

Anda dapat mencabut AWS Private CA sertifikat menggunakan AWS CLI perintah revoke-certificate atau tindakan API. RevokeCertificate Sertifikat mungkin perlu dicabut sebelum kedaluwarsa yang

dijadwalkan jika, misalnya, kunci rahasianya dikompromikan atau domain terkaitnya menjadi tidak valid. Agar pencabutan efektif, klien yang menggunakan sertifikat memerlukan cara untuk memeriksa status pencabutan setiap kali mencoba membangun koneksi jaringan yang aman.

AWS Private CAmenyediakan dua mekanisme yang dikelola sepenuhnya untuk mendukung pemeriksaan status pencabutan: Protokol Status Sertifikat Online (OCSP) dan daftar pencabutan sertifikat (CRL). Dengan OCSP, klien menanyakan database pencabutan otoritatif yang mengembalikan status secara real-time. Dengan CRL, klien memeriksa sertifikat terhadap daftar sertifikat yang dicabut yang diunduh dan disimpan secara berkala. Klien menolak untuk menerima sertifikat yang telah dicabut.

Baik OCSP dan CRL bergantung pada informasi validasi yang tertanam dalam sertifikat. Untuk alasan ini, CA penerbitan harus dikonfigurasi untuk mendukung salah satu atau kedua mekanisme ini sebelum penerbitan. Untuk informasi tentang memilih dan menerapkan pencabutan terkelola melaluiAWS Private CA, lihat. Menyiapkan metode pencabutan sertifikat

Sertifikat yang dicabut selalu dicatat dalam laporan AWS Private CA audit.

> **Note**
>
> Penerbit sertifikat lintas akun memerlukan izin tambahan untuk mencabut sertifikat yang mereka terbitkan; jika tidak, pemilik CA harus melakukan pencabutan. Untuk mengaktifkan pencabutan oleh penerbit lintas akun, administrator CA harus membuat dua saham RAM, keduanya menunjuk pada CA yang sama:
>
> 1. Berbagi dengan `AWSRAMRevokeCertificateCertificateAuthority` izin.
> 2. Berbagi dengan `AWSRAMDefaultPermissionCertificateAuthority` izin.

Untuk mencabut sertifikat

Gunakan tindakan RevokeCertificateAPI atau perintah pencabutan sertifikat untuk mencabut sertifikat PKI pribadi. Nomor seri harus dalam format heksadesimal. Anda dapat mengambil nomor seri dengan memanggil perintah get-certificate. Perintah `revoke-certificate` tidak mengembalikan respons.

```
$ 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"
```

Sertifikat dan OCSP yang dicabut

Tanggapan OCSP dapat memakan waktu hingga 60 menit untuk mencerminkan status baru saat Anda mencabut sertifikat. Secara umum, OCSP cenderung mendukung distribusi informasi pencabutan yang lebih cepat karena, tidak seperti CRL yang dapat di-cache oleh klien selama berhari-hari, respons OCSP biasanya tidak di-cache oleh klien.

Sertifikat yang dicabut di CRL

CRL biasanya diperbarui sekitar 30 menit setelah sertifikat dicabut. Jika karena alasan apa pun pembaruan CRL gagal, lakukan AWS Private CA upaya lebih lanjut setiap 15 menit.

Dengan Amazon CloudWatch, Anda dapat membuat alarm untuk metrik `CRLGenerated` dan. `MisconfiguredCRLBucket` Untuk informasi selengkapnya, lihat CloudWatchMetrik yang Didukung. Untuk informasi selengkapnya tentang membuat dan mengonfigurasi CRL, lihat Merencanakan daftar pencabutan sertifikat (CRL).

Contoh berikut menunjukkan sertifikat yang dicabut dalam daftar pencabutan sertifikat (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
```

Sertifikat yang dicabut dalam laporan audit

Semua sertifikat, termasuk sertifikat yang dicabut, disertakan dalam laporan audit untuk CA privat. Contoh berikut menunjukkan laporan audit dengan satu diterbitkan dan satu sertifikat dicabut. Untuk informasi selengkapnya, lihat Menggunakan laporan audit dengan CA privat Anda.

```
[
   {
      "awsAccountId":"account",
      "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
      "serial":"serial_number",

 "Subject":"1.2.840.113549.1.9.1=#161173616c6573406578616d706c652e636f6d,CN=www.example1.com,OU
 Company,L=Seattle,ST=Washington,C=US",
      "notBefore":"2018-02-26T18:39:57+0000",
      "notAfter":"2019-02-26T19:39:57+0000",
      "issuedAt":"2018-02-26T19:39:58+0000",
      "revokedAt":"2018-02-26T20:00:36+0000",
      "revocationReason":"KEY_COMPROMISE"
   },
   {
      "awsAccountId":"account",
      "certificateArn":"arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID",
      "serial":"serial_number",
```

```
 "Subject":"1.2.840.113549.1.9.1=#161970726f64407777772e70616c6f75736573616c65732e636f6d,CN=www
 Company,L=Seattle,ST=Washington,C=US",
      "notBefore":"2018-01-22T20:10:49+0000",
      "notAfter":"2019-01-17T21:10:49+0000",
      "issuedAt":"2018-01-22T21:10:49+0000"
   }
]
```


Mengotomatiskan ekspor sertifikat yang diperbarui

Saat Anda menggunakan AWS Private CA untuk membuat CA, Anda dapat mengimpor CA tersebut ke dalam AWS Certificate Manager dan membiarkan ACM mengelola penerbitan dan perpanjangan sertifikat. Jika sertifikat yang diperbarui dikaitkan dengan layanan terintegrasi, layanan tersebut menerapkan sertifikat baru dengan mulus. Namun, jika sertifikat awalnya diekspor untuk digunakan di tempat lain di lingkungan PKI Anda (misalnya, di server atau perangkat lokal), Anda perlu mengekspornya lagi setelah perpanjangan.

Untuk contoh solusi yang mengotomatiskan proses ekspor ACM menggunakan Amazon dan EventBridge AWS Lambda, lihat Mengotomatiskan ekspor sertifikat yang diperbarui.

Memahami templat sertifikat

AWS Private CAmenggunakan templat konfigurasi untuk menerbitkan sertifikat CA dan sertifikat entitas akhir. Saat Anda menerbitkan sertifikat CA dari konsol PCA, templat sertifikat CA akar atau bawahan yang sesuai diterapkan secara otomatis.

Jika Anda menggunakan CLI atau API untuk menerbitkan sertifikat, Anda dapat menyediakan templat ARN sebagai parameter untuk tindakan `IssueCertificate`. Jika Anda tidak memberikan ARN, maka templat `EndEntityCertificate/V1` secara default. Untuk informasi selengkapnya, lihat dokumentasi perintah IssueCertificateAPI dan issue-certificate.

> **Note**
>
> AWS Certificate Manager(ACM) pengguna dengan akses bersama lintas akun ke CA pribadi dapat menerbitkan sertifikat terkelola yang ditandatangani oleh CA. Penerbit lintas akun dibatasi oleh kebijakan berbasis sumber daya dan hanya memiliki akses ke templat sertifikat entitas akhir berikut:

- EndEntityCertificate/V1
- EndEntityClientAuthCertificate/V1
- EndEntityServerAuthCertificate/V1
- BlankEndEntityCertificate_apiPassthrough/v1
- BlankEndEntityCertificate_apicsrPassthrough/v1
- Subordinatecacertificate_ 0/V1 PathLen

Untuk informasi selengkapnya, lihat Kebijakan berbasis sumber daya.

Topik

Variasi templat

AWS Private CAmendukung empat jenis template.

- Template dasar

  Templat yang telah ditentukan sebelumnya tempat tidak ada parameter passthrough yang diizinkan.

- Templat CSRPassThrough

  Templat yang memperpanjang versi templat dasar yang sesuai dengan mengizinkan passthrough CSR. Ekstensi dalam CSR yang digunakan untuk menerbitkan sertifikat disalin ke sertifikat yang diterbitkan. Dalam kasus tempat CSR berisi nilai ekstensi yang bertentangan dengan definisi templat, definisi templat akan selalu memiliki prioritas lebih tinggi. Untuk detail lebih lanjut tentang prioritas, lihat Urutan templat operasi.

- Templat ApiPassThrough

  Templat yang memperluas versi templat dasar yang sesuai dengan mengizinkan passthrough API. Nilai dinamis yang diketahui oleh administrator atau sistem perantara lainnya mungkin

tidak diketahui oleh entitas yang meminta sertifikat, mungkin tidak mungkin ditentukan dalam templat, dan mungkin tidak tersedia di CSR. Namun, Administrator CA, dapat mengambil informasi tambahan dari sumber data lain, seperti Direktori Aktif, untuk menyelesaikan permintaan. Misalnya, jika mesin tidak mengetahui unit organisasi apa yang dimilikinya, administrator dapat mencari informasi di Direktori Aktif dan menambahkannya ke permintaan sertifikat dengan menyertakan informasi dalam struktur JSON.

Nilai dalam parameter `ApiPassthrough` tindakan `IssueCertificate` disalin ke sertifikat yang diterbitkan. Dalam kasus tempat parameter `ApiPassthrough` berisi informasi yang bertentangan dengan definisi templat, definisi templat akan selalu memiliki prioritas lebih tinggi. Untuk detail lebih lanjut tentang prioritas, lihat Urutan templat operasi.

- ApicsRPassThrough template

  Templat yang memperluas versi templat dasar yang sesuai dengan mengizinkan passthrough API dan CSR. Ekstensi dalam CSR yang digunakan untuk menerbitkan sertifikat disalin ke sertifikat yang diterbitkan, dan nilai dalam parameter `ApiPassthrough` tindakan`IssueCertificate` juga disalin. Jika definisi templat, nilai passthrough API, dan ekstensi passthrough CSR menunjukkan konflik, definisi templat memiliki prioritas tertinggi, diikuti oleh nilai passthrough API, diikuti oleh ekstensi passthrough CSR. Untuk detail lebih lanjut tentang prioritas, lihat Urutan templat operasi.

Tabel di bawah ini mencantumkan semua jenis templat AWS Private CA yang didukung oleh tautan ke definisinya.

> **Note**
>
> Untuk informasi tentang ARN templat di GovCloud wilayah, lihat AWS Private Certificate Authoritydi Panduan AWS GovCloud (US) Pengguna.

Templat Dasar

Nama Templat	ARN Templat	Jenis Sertifikat
CodeSigningCertificate/V1	`arn:aws:acm-pca:::template/CodeSigningCertificate/V1`	Penandatanganan kode

Nama Templat	ARN Templat	Jenis Sertifikat
EndEntityCertificate/V1	`arn:aws:acm-pca:::template/EndEntityCertificate/V1`	Entitas akhir
EndEntityClientAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate/V1`	Entitas akhir
EndEntityServerAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate/V1`	Entitas akhir
OCSP/V1 SigningCertificate	`arn:aws:acm-pca:::template/OCSPSigningCertificate/V1`	Penandatanganan OCSP
Rootcacertificate/V1	`arn:aws:acm-pca:::template/RootCACertificate/V1`	CA
Subordinatecacertificate_ 0/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1`	CA
Subordinatecacertificate_ 1/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1/V1`	CA
Subordinatecacertificate_ 2/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2/V1`	CA

Nama Templat	ARN Templat	Jenis Sertifikat
Subordinatecacertificate_ 3/V1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3/V1	CA

Templat CSDRPassThrough

Nama Templat	ARN Templat	Jenis Sertifikat
BlankEndEntityCertificate_CSRPassthrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CSRPassthrough/V1	Entitas akhir
BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassthrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassthrough/V1	Entitas akhir
BlankSubordinateCacertificate_0_CSRPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
BlankSubordinateCacertificate_1_CSRPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
BlankSubordinateCacertificate_2_CSRPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubo	CA

Nama Templat	ARN Templat	Jenis Sertifikat
	rdinateCACertificate_PathLen2_CSRPassthrough/V1	
BlankSubordinateCacertificate_3_CSRPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA
CodeSigningCertificate_CSRPassthrough/v1	arn:aws:acm-pca:::template/CodeSigningCertificate_CSRPassthrough/V1	Penandatanganan kode
EndEntityCertificate_CSRPassthrough/v1	arn:aws:acm-pca:::template/EndEntityCertificate_CSRPassthrough/V1	Entitas akhir
EndEntityClientAuthCertificate_CSRPassthrough/v1	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_CSRPassthrough/V1	Entitas akhir
EndEntityServerAuthCertificate_CSRPassthrough/v1	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_CSRPassthrough/V1	Entitas akhir
OCSP SigningCertificate_CSRPassthrough/v1	arn:aws:acm-pca:::template/OCSPSigningCertificate_CSRPassthrough/V1	Penandatanganan OCSP

Nama Templat	ARN Templat	Jenis Sertifikat
PathLenSubordinatecacertificate_0_CSRPassthrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
PathLenSubordinatecacertificate_1_CSRPassthrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
PathLenSubordinatecacertificate_2_CSRPassthrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
PathLenSubordinatecacertificate_3_CSRPassthrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA

Templat APIPassthrough

Nama Templat	ARN Templat	Jenis Sertifikat
BlankEndEntityCertificate_apiPassthrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_APIPassthrough/V1	Entitas akhir

Nama Templat	ARN Templat	Jenis Sertifikat
BlankEndEntityCertificate_CriticalBasicConstraints _apiPassthrough/v1	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough/V1`	Entitas akhir
CodeSigningCertificate_apiPassthrough/v1	`arn:aws:acm-pca:::template/CodeSigningCertificate_APIPassthrough/V1`	Penandatanganan kode
EndEntityCertificate_apiPassthrough/v1	`arn:aws:acm-pca:::template/EndEntityCertificate_APIPassthrough/V1`	Entitas akhir
EndEntityClientAuthCertificate_apiPassthrough/v1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_APIPassthrough/V1`	Entitas akhir
EndEntityServerAuthCertificate_apiPassthrough/v1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APIPassthrough/V1`	Entitas akhir
OCSP SigningCertificate _apiPassthrough/v1	`arn:aws:acm-pca:::template/OCSPSigningCertificate_APIPassthrough/V1`	Penandatanganan OCSP
RootCacertificate_apiPassthrough/v1	`arn:aws:acm-pca:::template/RootCACertificate_APIPassthrough/V1`	CA

Nama Templat	ARN Templat	Jenis Sertifikat
BlankRootCacertificate_apiPassthrough/v1	arn:aws:acm-pca:::template/BlankRootCACertificate_APIPassthrough/V1	CA
BlankRootCacertificate_0_apiPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen0_APIPassthrough/V1	CA
BlankRootCacertificate_1_apiPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen1_APIPassthrough/V1	CA
BlankRootCacertificate_2_apiPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen2_APIPassthrough/V1	CA
BlankRootCacertificate_3_apiPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen3_APIPassthrough/V1	CA
PathLenSubordinatecacertificate_0_apiPassthrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APIPassthrough/V1	CA
BlankSubordinateCacertificate_0_apiPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1	CA

Nama Templat	ARN Templat	Jenis Sertifikat
Subordinatecacertificate_ PathLen 1_ApiPassthrough/v1	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen1_APIPassthrough/ V1	CA
BlankSubordinateCacertificate_ 1_apiPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen1_APIPas sthrough/V1	CA
Subordinatecacertificate_ PathLen 2_ApiPassthrough/v1	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen2_APIPassthrough/ V1	CA
BlankSubordinateCacertificate_ 2_apiPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen2_APIPas sthrough/V1	CA
PathLenSubordinatecacertificate_ 3_apiPassthrough/v1	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen3_APIPassthrough/ V1	CA
BlankSubordinateCacertificate_ 3_apiPassthrough/v1 PathLen	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen3_APIPas sthrough/V1	CA

Templat APICSRPassthrough

Nama Templat	ARN Templat	Jenis Sertifikat
BlankEndEntityCertificate_apicsrPassthrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_APICSRPassthrough/V1	Entitas akhir
BlankEndEntityCertificate_CriticalBasicConstraints _apicsrPassthrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APICSRPassthrough/V1	Entitas akhir
CodeSigningCertificate_apicsrPassthrough/v1	arn:aws:acm-pca:::template/CodeSigningCertificate_APICSRPassthrough/V1	Penandatanganan kode
EndEntityCertificate_apicsrPassthrough/v1	arn:aws:acm-pca:::template/EndEntityCertificate_APICSRPassthrough/V1	Entitas akhir
EndEntityClientAuthCertificate_apicsrPassthrough/v1	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_APICSRPassthrough/V1	Entitas akhir
EndEntityServerAuthCertificate_apicsrPassthrough/v1	arn:aws:acm-pca:::template/EndEntityServerAuthCertific	Entitas akhir

Nama Templat	ARN Templat	Jenis Sertifikat
	ate_APICSRPassthrough/V1	
OCSP SigningCertificate_apicsrPassthrough/v1	arn:aws:acm-pca:::template/OCSPSigningCertificate_APICSRPassthrough/V1	Penandatanganan OCSP
PathLenSubordinatecacertificate_0_APICSRPassthrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
BlankSubordinateCacertificate_ 0_APICSRPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
PathLenSubordinatecacertificate_1_APICSRPassthrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA
BlankSubordinateCacertificate_ 1_APICSRPassthrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA

Nama Templat	ARN Templat	Jenis Sertifikat
SubordinateCacertificate_ 2_APICSRPassThrough/ PathLen 3_ApiPassThroughV1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APICSRPassthrough/V1`	CA
BlankSubordinateCacertificate_ 2_APICSRPassthrough/v1 PathLen	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1`	CA
PathLenSubordinatecacertificate_ 3_APICSRPassthrough/v1	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA
BlankSubordinateCacertificate_ 3_APICSRPassthrough/v1 PathLen	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA

Urutan templat operasi

Informasi yang ada dalam sertifikat yang diterbitkan dapat berasal dari empat sumber: definisi templat, passthrough API, passthrough CSR, dan konfigurasi CA.

Nilai passthrough API hanya dipatuhi saat Anda menggunakan passthrough API atau templat passthrough APICSR. Passthrough CSR hanya dihormati saat Anda menggunakan templat passthrough CSRPassthrough atau APICSR. Ketika sumber informasi ini bertentangan, aturan umum biasanya berlaku: Untuk setiap nilai ekstensi, definisi templat memiliki prioritas tertinggi, diikuti oleh nilai passthrough API, diikuti oleh ekstensi passthrough CSR.

Contoh

1. Definisi template untuk EndEntityClientAuthCertificate_apiPassThrough mendefinisikan ExtendedKeyUsage ekstensi dengan nilai “otentikasi server web TLS, otentikasi klien web TLS”. Jika ExtendedKeyUsage didefinisikan dalam CSR atau `IssueCertificate` `ApiPassthrough` parameter, `ApiPassthrough` nilai untuk ExtendedKeyUsage akan diabaikan karena definisi template diprioritaskan, dan nilai CSR untuk ExtendedKeyUsage nilai akan diabaikan karena template bukan variasi passthrough CSR.

   > **Note**
   >
   > Definisi templat tetap menyalin nilai-nilai lain dari CSR, seperti Nama Alternatif Subjek dan Subjek. Nilai-nilai ini tetap diambil dari CSR meskipun templat bukan merupakan variasi passthrough CSR, karena definisi templat selalu menjadi prioritas tertinggi.

2. Definisi template untuk EndEntityClientAuthCertificate_apicSrPassThrough mendefinisikan ekstensi Subject Alternative Name (SAN) sebagai disalin dari API atau CSR. Jika ekstensi SAN didefinisikan dalam CSR dan disediakan dalam parameter `IssueCertificate` `ApiPassthrough`, nilai passthrough API akan diprioritaskan karena nilai passthrough API lebih diprioritaskan daripada nilai passthrough CSR.

Definisi templat

Bagian berikut memberikan rincian konfigurasi tentang template AWS Private CA sertifikat yang didukung.

BlankEndEntityCertificateDefinisi _apiPassthrough/v1

Dengan templat sertifikat entitas akhir kosong, Anda dapat menerbitkan sertifikat entitas akhir hanya dengan batasan X.509 Basic yang ada. Ini adalah sertifikat entitas akhir paling sederhana yang AWS Private CA dapat diterbitkan, tetapi dapat disesuaikan menggunakan struktur API. Ekstensi batasan dasar menentukan apakah sertifikat tersebut merupakan sertifikat CA atau bukan. Templat sertifikat entitas akhir kosong memberlakukan nilai FALSE untuk batasan Dasar untuk memastikan bahwa sertifikat entitas akhir diterbitkan dan bukan sertifikat CA.

Anda dapat menggunakan templat passthrough kosong untuk mengeluarkan sertifikat kartu pintar yang memerlukan nilai khusus untuk penggunaan Kunci (KU) dan Penggunaan kunci yang diperluas (EKU). Misalnya, penggunaan kunci yang Diperpanjang mungkin memerlukan Autentikasi Klien dan Masuk Kartu Cerdas, dan penggunaan Kunci mungkin memerlukan Tanda Tangan Digital, Non Repudiation, dan Enkripsi Kunci. Tidak seperti templat passthrough lainnya, templat sertifikat

entitas akhir kosong memungkinkan konfigurasi ekstensi KU dan EKU, di mana KU dapat berupa salah satu dari sembilan nilai yang didukung (DigitalSignature, NonRepudiation, KeyEncipherment, DataEncipherment, KeyAgreement,, CRLSign, EncipherOnly, dan DecipherOnly) dan EKU dapat berupa salah satu nilai yang didukung (ServerAuth, ClientAuth keyCertSign, codesign, EmailProtection, timestamping, dan OcspSigning) ditambah ekstensi khusus.

BlankEndEntityCertificate_apiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	CA:FALSE
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankEndEntityCertificate_apicsrPassThrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankEndEntityCertificate_apicsrPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	CA:FALSE
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]

Parameter X509v3	Nilai
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankEndEntityCertificate_ CriticalBasicConstraints _apicsrPassThrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _apicsrPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Kritis, CA: SALAH
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA, API, atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankEndEntityCertificate_ CriticalBasicConstraints _ApiPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _apiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Kritis, CA: SALAH
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau API]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankEndEntityCertificate_ CriticalBasicConstraints _CSRPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankEndEntityCertificate_ CriticalBasicConstraints _CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Kritis, CA: SALAH
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankEndEntityCertificate_CSRPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankEndEntityCertificate_CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	CA:FALSE
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinatePathLenCacertificate_ 0_CSRPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 0_CSRPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 0`

Parameter X509v3	Nilai
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinatePathLenCacertificate_ 0_apicsrPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 0_APICSRPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 0`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinatePathLenCacertificate_ 0_apiPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 0_apiPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 0`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

BlankSubordinateDefinisi PathLen cacertificate_ 1_apiPassthrough/v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 1_apiPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 1`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinatePathLenCacertificate_ 1_CSRPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 1_CSRPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 1`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinatePathLenCacertificate_ 1_APICSRPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 1_APICSRPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 1`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]

Parameter X509v3	Nilai
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinateDefinisi PathLen cacertificate_ 2_apiPassthrough/v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 2_apiPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 2`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinatePathLenCacertificate_ 2_CSRPassthrough/definisi V1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 2_CSRPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 2`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinatePathLenCacertificate_ 2_apicsrPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 2_APICSRPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 2`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinateDefinisi PathLen cacertificate_ 3_apiPassthrough/v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 3_apiPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 3`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinatePathLenCacertificate_ 3_CSRPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 3_CSRPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 3`

Parameter X509v3	Nilai
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinatePathLenCacertificate_ 3_APICSRPassthrough/definisi v1

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificateDefinisi _apiPassthrough/v1.

BlankSubordinateCacertificate_ 3_APICSRPassThrough PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 3`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

CodeSigningCertificate/V1 definisi

Templat ini digunakan untuk membuat sertifikat untuk penandatanganan kode. Anda dapat menggunakan sertifikat penandatanganan kode dari AWS Private CA solusi penandatanganan kode apa pun yang didasarkan pada infrastruktur CA pribadi. Misalnya, pelanggan yang menggunakan

Penandatanganan Kode untuk AWS IoT dapat menghasilkan sertifikat penandatanganan kode dengan AWS Private CA dan mengimpornya ke. AWS Certificate Manager Untuk informasi selengkapnya, lihat Untuk apa Penandatanganan KodeAWS IoT? dan Dapatkan dan Impor Sertifikat Penandatanganan Kode.

CodeSigningCertificate/V1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	`CA:FALSE`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital
Penggunaan kunci yang diperpanjang	Penting, penandatanganan kode
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

*Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

CodeSigningCertificate_apicsrPassThrough/definisi v1

Template ini memperluas CodeSigningCertificate /V1 untuk mendukung nilai passthrough API dan CSR.

CodeSigningCertificate_apicsrPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]

Parameter X509v3	Nilai
Kendala Dasar	`CA:FALSE`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital
Penggunaan kunci yang diperpanjang	Penting, penandatanganan kode
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

CodeSigningCertificateDefinisi _apiPassthrough/v1

Template ini identik dengan `CodeSigningCertificate` template dengan satu perbedaan: Dalam template ini, AWS Private CA meneruskan ekstensi tambahan melalui API ke sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di API.

CodeSigningCertificate_apiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	`CA:FALSE`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital
Penggunaan kunci yang diperpanjang	Penting, penandatanganan kode

Parameter X509v3	Nilai
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

CodeSigningCertificate_CSRPassthrough/definisi v1

Template ini identik dengan `CodeSigningCertificate` template dengan satu perbedaan: Dalam template ini, AWS Private CA meneruskan ekstensi tambahan dari permintaan penandatanganan sertifikat (CSR) ke dalam sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di CSR.

CodeSigningCertificate_CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	`CA:FALSE`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital
Penggunaan kunci yang diperpanjang	Penting, penandatanganan kode
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

*Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

EndEntityCertificate/V1 definisi

Templat ini digunakan untuk membuat sertifikat untuk entitas akhir seperti sistem operasi atau server web.

EndEntityCertificate/V1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	CA:FALSE
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi server web TLS, autentikasi klien web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

*Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

EndEntityCertificate_apicsrPassThrough/definisi v1

Template ini memperluas EndEntityCertificate /V1 untuk mendukung nilai passthrough API dan CSR.

EndEntityCertificate_apicsrPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	CA:FALSE

Parameter X509v3	Nilai
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi server web TLS, autentikasi klien web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

EndEntityCertificateDefinisi _apiPassthrough/v1

Template ini identik dengan `EndEntityCertificate` template dengan satu perbedaan: Dalam template ini, AWS Private CA meneruskan ekstensi tambahan melalui API ke sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di API.

EndEntityCertificate_apiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	CA:`FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi server web TLS, autentikasi klien web TLS

Parameter X509v3	Nilai
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

EndEntityCertificate_CSRPassthrough/definisi v1

Template ini identik dengan `EndEntityCertificate` template dengan satu perbedaan: Dalam template ini, AWS Private CA meneruskan ekstensi tambahan dari permintaan penandatanganan sertifikat (CSR) ke dalam sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di CSR.

EndEntityCertificate_CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	CA:`FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi server web TLS, autentikasi klien web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

*Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

EndEntityClientAuthCertificate/V1 definisi

Templat ini berbeda dari satu-satunya `EndEntityCertificate` dalam Nilai penggunaan kunci yang diperpanjang, yang membatasinya untuk autentikasi klien web TLS.

EndEntityClientAuthCertificate/V1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	CA:`FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi klien web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

*Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

EndEntityClientAuthCertificate_apicsrPassThrough/definisi v1

Template ini memperluas EndEntityClientAuthCertificate /V1 untuk mendukung nilai passthrough API dan CSR.

EndEntityClientAuthCertificate_apicsrPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	CA:`FALSE`

Parameter X509v3	Nilai
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi klien web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

EndEntityClientAuthCertificateDefinisi _apiPassthrough/v1

Templat ini identik dengan templat `EndEntityClientAuthCertificate` dengan satu perbedaan. Dalam template ini, AWS Private CA meneruskan ekstensi tambahan melalui API ke dalam sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di API.

EndEntityClientAuthCertificate_apiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	CA:`FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi klien web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

EndEntityClientAuthCertificate_CSRPassthrough/definisi v1

Templat ini identik dengan templat `EndEntityClientAuthCertificate` dengan satu perbedaan. Dalam template ini, AWS Private CA meneruskan ekstensi tambahan dari permintaan penandatanganan sertifikat (CSR) ke dalam sertifikat jika ekstensi tidak ditentukan dalam templat. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di CSR.

EndEntityClientAuthCertificate_CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	CA:`FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi klien web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

*Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

EndEntityServerAuthCertificate/V1 definisi

Templat ini berbeda dari satu-satunya `EndEntityCertificate` dalam Nilai penggunaan kunci yang diperpanjang, yang membatasinya pada autentikasi server web TLS.

EndEntityServerAuthCertificate/V1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	CA:FALSE
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi server web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

*Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

EndEntityServerAuthCertificate_apicsrPassThrough/definisi v1

Template ini memperluas EndEntityServerAuthCertificate /V1 untuk mendukung nilai passthrough API dan CSR.

EndEntityServerAuthCertificate_apicsrPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	CA:FALSE
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]

Parameter X509v3	Nilai
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi server web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

EndEntityServerAuthCertificateDefinisi _apiPassthrough/v1

Templat ini identik dengan templat `EndEntityServerAuthCertificate` dengan satu perbedaan. Dalam template ini, AWS Private CA meneruskan ekstensi tambahan melalui API ke dalam sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di API.

EndEntityServerAuthCertificate_apiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	CA:`FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi server web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

EndEntityServerAuthCertificate_CSRPassthrough/definisi v1

Templat ini identik dengan templat `EndEntityServerAuthCertificate` dengan satu perbedaan. Dalam template ini, AWS Private CA meneruskan ekstensi tambahan dari permintaan penandatanganan sertifikat (CSR) ke dalam sertifikat jika ekstensi tidak ditentukan dalam templat. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di CSR.

EndEntityServerAuthCertificate_CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	CA:`FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, penyandian kunci, tanda tangan digital
Penggunaan kunci yang diperpanjang	Autentikasi server web TLS
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

*Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

Definisi SigningCertificate OCSP/V1

Templat ini digunakan untuk membuat sertifikat untuk menandatangani tanggapan OCSP. Templat identik dengan templat `CodeSigningCertificate`, kecuali bahwa Nilai penggunaan kunci yang diperpanjang menentukan penandatanganan OCSP, bukan penandatanganan kode.

OCSP/V1 SigningCertificate

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]

Parameter X509v3	Nilai
Subjek	[Passthrough dari CSR]
Kendala Dasar	`CA:FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital
Penggunaan kunci yang diperpanjang	Penting, penandatanganan OCSP
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

*Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

Definisi OCSP SigningCertificate _apicSrPassthrough/v1

Template ini memperluas SigningCertificate OCSP/V1 untuk mendukung nilai passthrough API dan CSR.

OCSP SigningCertificate _apicsrPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	`CA:FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital
Penggunaan kunci yang diperpanjang	Penting, penandatanganan OCSP

Parameter X509v3	Nilai
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

Definisi OCSP SigningCertificate _apiPassthrough/v1

Templat ini identik dengan templat `OCSPSigningCertificate` dengan satu perbedaan. Dalam template ini, AWS Private CA meneruskan ekstensi tambahan melalui API ke dalam sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di API.

OCSP SigningCertificate _apiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	`CA:FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital
Penggunaan kunci yang diperpanjang	Penting, penandatanganan OCSP
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

Definisi OCSP SigningCertificate _CSRPassthrough/v1

Templat ini identik dengan templat `OCSPSigningCertificate` dengan satu perbedaan. Dalam template ini, AWS Private CA meneruskan ekstensi tambahan dari permintaan penandatanganan sertifikat (CSR) ke dalam sertifikat jika ekstensi tidak ditentukan dalam templat. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di CSR.

OCSP SigningCertificate _CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	`CA:FALSE`
Pengidentifikasi kunci otoritas	[SKI dari sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital
Penggunaan kunci yang diperpanjang	Penting, penandatanganan OCSP
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

*Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

Definisi RootCACertificate/V1

Templat ini digunakan untuk menerbitkan sertifikat CA akar yang ditandatangani sendiri. Sertifikat CA menyertakan ekstensi kendala dasar kritis dengan bidang CA yang disetel `TRUE` untuk menetapkan bahwa sertifikat dapat digunakan untuk menerbitkan sertifikat CA. Template tidak menentukan panjang jalur (pathLenConstraint) karena ini dapat menghambat perluasan hierarki di masa depan. Penggunaan kunci yang diperpanjang dikecualikan untuk mencegah penggunaan sertifikat CA sebagai klien TLS atau sertifikat server. Tidak ada informasi CRL yang ditentukan karena sertifikat yang ditandatangani sendiri tidak dapat dicabut.

RootCACertificate/V1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Kritis, tanda tangan digital, keyCertSign, tanda CRL
Titik distribusi CRL	N/A

Definisi RootCACertificate_APIPassthrough/V1

Templat ini memperpanjang RootCACertificate/V1 untuk mendukung nilai passthrough API dan CSR.

RootCACertificate_APIPassthrough/V1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`
Pengidentifikasi kunci otoritas	[Passthrough dari API]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Kritis, tanda tangan digital, keyCertSign, tanda CRL
Titik distribusi CRL*	N/A

BlankRootDefinisi Cacertificate_apiPassthrough/v1

Dengan templat sertifikat root kosong, Anda dapat menerbitkan root certificate hanya dengan batasan dasar X.509. Ini adalah sertifikat root paling sederhana yang AWS Private CA dapat diterbitkan, tetapi dapat disesuaikan menggunakan struktur API. Ekstensi kendala dasar menentukan apakah sertifikat tersebut adalah sertifikat CA atau tidak. Sebuah template sertifikat root kosong memberlakukan nilai `TRUE` untuk kendala dasar untuk memastikan bahwa sertifikat CA root dikeluarkan.

Anda dapat menggunakan templat root passthrough kosong untuk mengeluarkan sertifikat root yang memerlukan nilai khusus untuk penggunaan kunci (KU). Misalnya, penggunaan kunci mungkin memerlukan `keyCertSign` dan`cRLSign`, tetapi tidak`digitalSignature`. Tidak seperti template sertifikat passthrough root non-blank lainnya, templat sertifikat root kosong memungkinkan konfigurasi ekstensi KU, di mana KU dapat berupa salah satu dari sembilan nilai yang didukung (`digitalSignature`,,`nonRepudiation`,`keyEncipherment`,`dataEncipherment`,`keyAgreement`, `keyCertSign` `cRLSign`encipherOnly, dan`decipherOnly`).

BlankRootCacertificate_apiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`
Pengidentifikasi kunci subjek	[Berasal dari CSR]

BlankRootPathLenCacertificate_ 0_apiPassthrough/definisi v1

Untuk informasi umum tentang templat CA root kosong, lihat???.

BlankRootCacertificate_ 0_apiPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 0`

Parameter X509v3	Nilai
Pengidentifikasi kunci subjek	[Berasal dari CSR]

BlankRootDefinisi PathLen cacertificate_ 1_apiPassthrough/v1

Untuk informasi umum tentang templat CA root kosong, lihat???.

BlankRootCacertificate_ 1_apiPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 1`
Pengidentifikasi kunci subjek	[Berasal dari CSR]

BlankRootDefinisi PathLen cacertificate_ 2_apiPassthrough/v1

Untuk informasi umum tentang templat CA root kosong, lihat???.

BlankRootCacertificate_ 2_apiPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 2`
Pengidentifikasi kunci subjek	[Berasal dari CSR]

BlankRootDefinisi PathLen cacertificate_ 3_apiPassthrough/v1

Untuk informasi umum tentang templat CA root kosong, lihat???.

BlankRootCacertificate_ 3_apiPassthrough/v1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 3`
Pengidentifikasi kunci subjek	[Berasal dari CSR]

SubordinateCacertificate_ 0/V1 definisi PathLen

Template ini digunakan untuk menerbitkan sertifikat CA bawahan dengan panjang jalur. `0` Sertifikat CA menyertakan ekstensi kendala dasar kritis dengan bidang CA yang disetel `TRUE` untuk menetapkan bahwa sertifikat dapat digunakan untuk menerbitkan sertifikat CA. Penggunaan kunci yang diperpanjang tidak disertakan, sehingga sertifikat CA tidak digunakan sebagai klien TLS atau sertifikat server.

Untuk informasi selengkapnya tentang jalur sertifikat, lihat Menetapkan Batasan Panjang di Jalur Sertifikat.

Subordinatecacertificate_ 0/V1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 0`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam sertifikat yang dikeluarkan dengan templat ini hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

SubordinateCacertificate_ PathLen 0_apicsrPassthrough/definisi v1

Template ini memperluas SubordinateCacertificate_ PathLen 0/V1 untuk mendukung nilai passthrough API dan CSR.

PathLenSubordinatecacertificate_ 0_APICSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 0`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

SubordinateCacertificate_ PathLen 0_ApiPassthrough/definisi v1

Template ini memperluas SubordinateCacertificate_ 0/V1 PathLen untuk mendukung nilai passthrough API.

PathLenSubordinatecacertificate_ 0_apiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]

Parameter X509v3	Nilai
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 0`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

SubordinateCacertificate_ PathLen 0_CSRPassthrough/definisi v1

Template ini identik dengan `SubordinateCACertificate_PathLen0` template dengan satu perbedaan: Dalam template ini, AWS Private CA meneruskan ekstensi tambahan dari permintaan penandatanganan sertifikat (CSR) ke dalam sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di CSR.

> **Note**
>
> CSR yang berisi ekstensi tambahan khusus harus dibuat di luar. AWS Private CA

PathLenSubordinatecacertificate_ 0_CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 0`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]

Parameter X509v3	Nilai
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

*Titik distribusi CRL disertakan dalam sertifikat yang diterbitkan dengan templat ini hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

SubordinateCacertificate_ 1/V1 definisi PathLen

Template ini digunakan untuk menerbitkan sertifikat CA bawahan dengan panjang jalur. 1 Sertifikat CA menyertakan ekstensi batasan Dasar yang penting dengan bidang CA yang diatur ke TRUE untuk menunjukkan bahwa sertifikat dapat digunakan untuk menerbitkan sertifikat CA. Penggunaan kunci yang diperpanjang tidak disertakan, sehingga sertifikat CA tidak digunakan sebagai klien TLS atau sertifikat server.

Untuk informasi selengkapnya tentang jalur sertifikat, lihat Menetapkan Batasan Panjang di Jalur Sertifikat.

Subordinatecacertificate_ 1/V1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 1`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL

Parameter X509v3	Nilai
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

*Titik distribusi CRL disertakan dalam sertifikat yang diterbitkan dengan templat ini hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

SubordinateCacertificate_ PathLen 1_APICSRPassthrough/definisi v1

Template ini memperluas subordinateCacertificate_ PathLen 1/V1 untuk mendukung nilai passthrough API dan CSR.

PathLenSubordinatecacertificate_ 1_APICSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 1`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

SubordinateCacertificate_ PathLen 1_ApiPassthrough/definisi v1

Template ini memperluas SubordinateCacertificate_ 0/V1 PathLen untuk mendukung nilai passthrough API.

Subordinatecacertificate_ PathLen 1_ApiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 1`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

SubordinateCacertificate_ PathLen 1_CSRPassthrough/definisi v1

Template ini identik dengan `SubordinateCACertificate_PathLen1` template dengan satu perbedaan: Dalam template ini, AWS Private CA meneruskan ekstensi tambahan dari permintaan penandatanganan sertifikat (CSR) ke dalam sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di CSR.

> **Note**
>
> CSR yang berisi ekstensi tambahan khusus harus dibuat di luar. AWS Private CA

PathLenSubordinatecacertificate_ 1_CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]

Parameter X509v3	Nilai
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 1`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

*Titik distribusi CRL disertakan dalam sertifikat yang diterbitkan dengan templat ini hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

SubordinateCacertificate_ 2/V1 definisi PathLen

Templat ini digunakan untuk menerbitkan sertifikat CA bawahan dengan panjang jalur 2. Sertifikat CA menyertakan ekstensi batasan Dasar yang penting dengan bidang CA yang diatur ke `TRUE` untuk menunjukkan bahwa sertifikat dapat digunakan untuk menerbitkan sertifikat CA. Penggunaan kunci yang diperpanjang tidak disertakan, sehingga sertifikat CA tidak digunakan sebagai klien TLS atau sertifikat server.

Untuk informasi selengkapnya tentang jalur sertifikat, lihat Menetapkan Batasan Panjang di Jalur Sertifikat.

Subordinatecacertificate_ 2/V1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 2`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]

Parameter X509v3	Nilai
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

*Titik distribusi CRL disertakan dalam sertifikat yang diterbitkan dengan templat ini hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

SubordinateCacertificate_ PathLen 2_APICSRPassthrough/definisi v1

Template ini memperluas subordinateCacertificate_ PathLen 2/V1 untuk mendukung nilai passthrough API dan CSR.

PathLenSubordinatecacertificate_ 2_APICSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 2`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

SubordinateCacertificate_ PathLen 2_ApiPassthrough/definisi v1

Template ini memperluas subordinateCacertificate_ 2/V1 PathLen untuk mendukung nilai passthrough API.

Subordinatecacertificate_ PathLen 2_ApiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 2`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

SubordinateCacertificate_ PathLen 2_CSRPassthrough/definisi v1

Template ini identik dengan `SubordinateCACertificate_PathLen2` template dengan satu perbedaan: Dalam template ini, AWS Private CA meneruskan ekstensi tambahan dari permintaan penandatanganan sertifikat (CSR) ke dalam sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di CSR.

> **Note**
>
> CSR yang berisi ekstensi tambahan khusus harus dibuat di luar. AWS Private CA

PathLenSubordinatecacertificate_ 2_CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 2`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

*Titik distribusi CRL disertakan dalam sertifikat yang diterbitkan dengan templat ini hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

SubordinateCacertificate_ 3/V1 definisi PathLen

Templat ini digunakan untuk menerbitkan sertifikat CA bawahan dengan panjang jalur 3. Sertifikat CA menyertakan ekstensi batasan Dasar yang penting dengan bidang CA yang diatur ke TRUE untuk menunjukkan bahwa sertifikat dapat digunakan untuk menerbitkan sertifikat CA. Penggunaan kunci yang diperpanjang tidak disertakan, sehingga sertifikat CA tidak digunakan sebagai klien TLS atau sertifikat server.

Untuk informasi selengkapnya tentang jalur sertifikat, lihat Menetapkan Batasan Panjang di Jalur Sertifikat.

Subordinatecacertificate_ 3/V1 PathLen

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]

Parameter X509v3	Nilai
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 3`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

*Titik distribusi CRL disertakan dalam sertifikat yang diterbitkan dengan templat ini hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

SubordinateCacertificate_ PathLen 3_APICSRPassthrough/definisi v1

Template ini memperluas SubordinateCacertificate_ PathLen 3/V1 untuk mendukung nilai passthrough API dan CSR.

PathLenSubordinatecacertificate_ 3_APICSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 3`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

SubordinateCacertificate_ PathLen 3_ApiPassthrough/definisi v1

Template ini memperluas SubordinateCacertificate_ 3/V1 PathLen untuk mendukung nilai passthrough API.

PathLenSubordinatecacertificate_ 3_apiPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari API atau CSR]
Subjek	[Passthrough dari API atau CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 3`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

SubordinateCacertificate_ PathLen 3_CSRPassthrough/definisi v1

Template ini identik dengan `SubordinateCACertificate_PathLen3` template dengan satu perbedaan: Dalam template ini, AWS Private CA meneruskan ekstensi tambahan dari permintaan penandatanganan sertifikat (CSR) ke dalam sertifikat jika ekstensi tidak ditentukan dalam template. Ekstensi yang ditentukan dalam templat selalu menimpa ekstensi di CSR.

> **Note**
>
> CSR yang berisi ekstensi tambahan khusus harus dibuat di luar. AWS Private CA

PathLenSubordinatecacertificate_ 3_CSRPassthrough/v1

Parameter X509v3	Nilai
Nama alternatif subjek	[Passthrough dari CSR]
Subjek	[Passthrough dari CSR]
Kendala Dasar	Penting, `CA:TRUE`, `pathlen: 3`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Penggunaan kunci	Penting, tanda tangan digital, `keyCertSign` , tanda tangan CRL
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

*Titik distribusi CRL disertakan dalam sertifikat yang diterbitkan dengan templat ini hanya jika CA dikonfigurasi dengan pembuatan CRL diaktifkan.

Menggunakan AWS Private CA API (contoh Java)

Anda dapat menggunakan AWS Private Certificate Authority API untuk berinteraksi secara terprogram dengan layanan dengan mengirimkan permintaan HTTP. Layanan tersebut mengembalikan tanggapan HTTP. Untuk informasi selengkapnya, lihat Referensi AWS Private Certificate Authority API.

Selain HTTP API, Anda dapat menggunakan AWS SDK dan alat baris perintah untuk berinteraksiAWS Private CA. Hal ini direkomendasikan melalui API HTTP. Untuk informasi lebih lanjut, lihat Alat untuk Layanan Web Amazon. Topik berikut menunjukkan cara menggunakan AWS SDK for Javauntuk memprogram AWS Private CA API.

Itu GetCertificateAuthorityCsr, GetCertificate, dan DescribeCertificateAuthorityAuditReportoperasi mendukung pelayan. Anda dapat menggunakan penunggu untuk mengontrol perkembangan kode Anda berdasarkan kehadiran atau keadaan sumber daya tertentu. Untuk informasi selengkapnya, lihat topik berikut, serta Pelayan AWS SDK for Java di Blog AWS Pengembang.

Topik

- Membuat dan mengaktifkan CA akar secara terprogram
- Membuat dan mengaktifkan CA bawahan secara terprogram
- CreateCertificateAuthority
- Menggunakan CreateCertificateAuthority untuk mendukung Active Directory
- CreateCertificateAuthorityAuditReport
- CreatePermission
- DeleteCertificateAuthority
- DeletePermission
- DeletePolicy
- DescribeCertificateAuthority
- DescribeCertificateAuthorityAuditReport
- GetCertificate
- GetCertificateAuthorityCertificate
- GetCertificateAuthorityCsr
- GetPolicy
- ImportCertificateAuthorityCertificate

- IssueCertificate
- ListCertificateAuthorities
- ListPermissions
- ListTags
- PutPolicy
- RestoreCertificateAuthority
- RevokeCertificate
- TagCertificateAuthorities
- UntagCertificateAuthority
- UpdateCertificateAuthority
- Buat CA dan sertifikat dengan nama subjek khusus
- Membuat sertifikat dengan ekstensi khusus

Membuat dan mengaktifkan CA akar secara terprogram

Contoh Java ini menunjukkan cara mengaktifkan CA akar menggunakan berikutAWS Private CATindakan API:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

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

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;
```

```
import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

public class RootCAActivation {
    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = "region"; // Substitute your region here, e.g. "us-
west-2"

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

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

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

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.ROOT;

        // ** Execute core code samples for Root CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCAArn = CreateCertificateAuthority(configCA, crlConfigure, CAtype,
 client);
        String csr = GetCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = IssueCertificate(rootCAArn, csr, client);
```

```
        String rootCertificate = GetCertificate(rootCertificateArn, rootCAArn, client);
        ImportCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

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

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

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

        return client;
    }

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

        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
```

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

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

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

        return rootCAArn;
    }

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

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

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

```
        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

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

        return csr;
    }

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

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

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

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

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

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

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(3650L);
```

```
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

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

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

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

        return rootCertificateArn;
    }

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

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

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

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

        // Create waiter to wait on successful creation of the certificate file.
```

```
        Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
        try {
            getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
        } catch (WaiterUnrecoverableException e) {
            //Explicit short circuit when the recourse transitions into
            //an undesired state.
        } catch (WaiterTimedOutException e) {
            //Failed to transition into desired state even after polling.
        } catch (AWSACMPCAException e) {
            //Unexpected service exception.
        }

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

        // Get the certificate and certificate chain and display the result.
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
    }

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

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

        ByteBuffer certByteBuffer = stringToByteBuffer(rootCertificate);
```

```
        importRequest.setCertificate(certByteBuffer);

        importRequest.setCertificateChain(null);

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Root CA certificate successfully imported.");
        System.out.println("Root CA activated successfully.");
    }

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

Membuat dan mengaktifkan CA bawahan secara terprogram

Contoh Java ini menunjukkan cara mengaktifkan CA bawahan menggunakan tindakan AWS Private CA API berikut:

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

```
package com.amazonaws.samples;

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


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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

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

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateResult;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
```

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

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

public class SubordinateCAActivation {

    public static void main(String[] args) throws Exception {
        // Place your own Root CA ARN here.
        String rootCAArn = "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566";

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

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setOrganization("Example Organization");
        subject.setOrganizationalUnit("Example");
        subject.setCountry("US");
```

```
        subject.setState("Virginia");
        subject.setLocality("Arlington");
        subject.setCommonName("www.example.com");

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

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

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.SUBORDINATE;

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

    }

    private static AWSACMPCA ClientBuilder(String endpointRegion) {
        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load the credentials from the credential profiles file. " +
```

```
                    "Please make sure that your credentials file is at the correct " +
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                    e);
        }

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

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

        return client;
    }

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

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

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

        // Retrieve and display the certificate information.
```

```
        String rootCertificate = getCACertificateResult.getCertificate();
        System.out.println("Root CA Certificate / Certificate Chain:");
        System.out.println(rootCertificate);

        return rootCertificate;
    }

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

        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withRevocationConfiguration(revokeConfig);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);

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

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

        return subordinateCAArn;
    }

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

        // Create the CSR request object and set the CA ARN.
```

```
        GetCertificateAuthorityCsrRequest csrRequest = new
GetCertificateAuthorityCsrRequest();
        csrRequest.withCertificateAuthorityArn(subordinateCAArn);

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

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

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

        return csr;
    }

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

        // Create a certificate request:
```

```
        IssueCertificateRequest issueRequest = new IssueCertificateRequest();

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

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/
SubordinateCACertificate_PathLen0/V1");

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

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

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(730L); // Approximately two years
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

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

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

        // Retrieve and display the certificate ARN.
        String subordinateCertificateArn = issueResult.getCertificateArn();
```

```
        System.out.println("Subordinate Certificate Arn: " +
subordinateCertificateArn);

        return subordinateCertificateArn;
    }

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

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

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

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

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

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

```
        } catch (InvalidStateException ex) {
            throw ex;
        }

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

        return subordinateCertificate;
    }

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

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

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

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

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
```

```
        } catch (RequestFailedException ex) {
            throw ex;
        }
        System.out.println("Subordinate CA certificate successfully imported.");
        System.out.println("Subordinate CA activated successfully.");
    }

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

CreateCertificateAuthority

Contoh Java berikut menunjukkan cara menggunakan CreateCerticateAuthorityoperasi.

Operasi ini membuat otoritas sertifikasi (CA) bawahan privat. Anda harus menentukan konfigurasi CA, konfigurasi pencabutan, jenis CA, dan token idempotensi opsional.

Konfigurasi CA menentukan hal berikut:

- Nama algoritme dan ukuran kunci yang akan digunakan untuk membuat kunci privat CA
- Jenis algoritme penandatanganan yang digunakan CA untuk menandatangani
- Informasi subjek X.500

Konfigurasi CRL menentukan hal berikut:

- Masa kedaluwarsa CRL dalam hari (masa berlaku CRL)
- Bucket Amazon S3 yang akan berisi CRL
- Alias CNAME untuk bucket S3 yang disertakan dalam sertifikat yang diterbitkan oleh CA

Jika berhasil, fungsi ini mengembalikan Amazon Resource Name (ARN) CA.

```
package com.amazonaws.samples;
```

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


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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.util.ArrayList;
import java.util.Objects;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;


public class CreateCertificateAuthority {

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

        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                   "Cannot load the credentials from the credential profiles file. " +
                   "Please make sure that your credentials file is at the correct " +
```

```
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
 format.",
                    e);
        }

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

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

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

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

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

        RevocationConfiguration revokeConfig = new RevocationConfiguration();
```

```
        revokeConfig.setCrlConfiguration(crlConfigure);

        // Define a certificate authority type: ROOT or SUBORDINATE
        CertificateAuthorityType CAtype = CertificateAuthorityType.<<SUBORDINATE>>;

        // Create a tag - method 1
        Tag tag1 = new Tag();
        tag1.withKey("PrivateCA");
        tag1.withValue("Sample");

        // Create a tag - method 2
        Tag tag2 = new Tag()
            .withKey("Purpose")
            .withValue("WebServices");

        // Add the tags to a collection.
        ArrayList<Tag> tags = new ArrayList<Tag>();
        tags.add(tag1);
        tags.add(tag2);

        // Create the request object.
        CreateCertificateAuthorityRequest req = new
CreateCertificateAuthorityRequest();
        req.withCertificateAuthorityConfiguration(configCA);
        req.withRevocationConfiguration(revokeConfig);
        req.withIdempotencyToken("123987");
        req.withCertificateAuthorityType(CAtype);
        req.withTags(tags);


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

        // Retrieve the ARN of the private CA.
        String arn = result.getCertificateAuthorityArn();
```

```
        System.out.println(arn);
    }
}
```

Output Anda harus serupa dengan berikut ini:

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

Menggunakan CreateCertificateAuthority untuk mendukung Active Directory

Contoh Java berikut menunjukkan cara menggunakan CreateCerticateAuthorityoperasi untuk membuat CA yang dapat diinstal di toko Enterprise NTAuth Microsoft Active Directory (AD).

Operasi membuat otoritas sertifikat root pribadi (CA) menggunakan pengidentifikasi objek kustom (OID). Untuk informasi selengkapnya dan AWS CLI contoh operasi yang setara, lihat Membuat CA untuk login Active Directory.

Jika berhasil, fungsi ini mengembalikan Amazon Resource Name (ARN) CA.

```
package com.amazonaws.samples.appstream;

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

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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CustomAttribute;
```

```
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
```

```
import com.amazonaws.waiters.WaiterUnrecoverableException;

import org.bouncycastle.asn1.x509.SubjectPublicKeyInfo;
import org.bouncycastle.cert.jcajce.JcaX509ExtensionUtils;
import org.bouncycastle.openssl.PEMParser;
import org.bouncycastle.pkcs.PKCS10CertificationRequest;
import org.bouncycastle.util.io.pem.PemReader;

import lombok.SneakyThrows;

public class RootCAActivation {
    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "ap-
southeast-2"

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("2.5.4.3") // OID for Common Name
                .withValue("root CA"),
            new CustomAttribute()
                .withObjectIdentifier("0.9.2342.19200300.100.1.25") // OID for Domain
 Component
                .withValue("example"),
            new CustomAttribute()
                .withObjectIdentifier("0.9.2342.19200300.100.1.25") // OID for Domain
 Component
                .withValue("com")

        );

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
 CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.EC_prime256v1);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
        configCA.withSubject(subject);

        // Define a certificate authority type
```

```
        CertificateAuthorityType CAtype = CertificateAuthorityType.ROOT;

        // ** Execute core code samples for Root CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCAArn = CreateCertificateAuthority(configCA, CAtype, client);
        String csr = GetCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = IssueCertificate(rootCAArn, csr, client);
        String rootCertificate = GetCertificate(rootCertificateArn, rootCAArn, client);
        ImportCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

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

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

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

        return client;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CertificateAuthorityType CAtype, AWSACMPCA client) {
        // Create the request object.
```

```
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);

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

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

        return rootCAArn;
    }

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

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

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

```
            //Unexpected service exception.
        }

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

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

        return csr;
    }

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

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

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

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

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

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

        // Set the validity period for the certificate to be issued.
```

```
        Validity validity = new Validity();
        validity.withValue(3650L);
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

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

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

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

        return rootCertificateArn;
    }

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

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

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

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

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

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

        // Get the certificate and certificate chain and display the result.
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
    }

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

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

```
        ByteBuffer certByteBuffer = stringToByteBuffer(rootCertificate);
        importRequest.setCertificate(certByteBuffer);

        importRequest.setCertificateChain(null);

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Root CA certificate successfully imported.");
        System.out.println("Root CA activated successfully.");
    }

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

Output Anda harus serupa dengan berikut ini:

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

CreateCertificateAuthorityAuditReport

Contoh Java berikut menunjukkan cara menggunakan CreateCertificateAuthorityAuditReportoperasi.

Operasi membuat laporan audit yang berisi daftar setiap kali sertifikat diterbitkan atau dicabut. Laporan ini disimpan di bucket Amazon S3 yang Anda tentukan pada input. Anda dapat membuat laporan baru setiap 30 menit sekali.

```
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);

   }
}
```

Output Anda harus serupa dengan berikut ini:

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

CreatePermission

Contoh Java berikut menunjukkan cara menggunakan CreatePermissionoperasi.

Operasi ini memberikan izin akses dari CA privat ke layanan utama AWS yang ditunjuk. Layanan dapat diberikan izin untuk membuat dan mengambil sertifikat dari CA privat, serta mencantumkan izin aktif yang telah diberikan CA privat. Untuk memperbarui sertifikat secara otomatis melalui ACM, Anda harus menetapkan semua izin yang mungkin (`IssueCertificate`,`GetCertificate`, dan`ListPermissions`) dari CA ke prinsipal layanan ACM (). `acm.amazonaws.com` Anda dapat menemukan ARN CA dengan memanggil fungsi. ListCertificateAuthorities

Setelah izin dibuat, Anda dapat memeriksanya dengan ListPermissionsfungsi atau menghapusnya dengan DeletePermissionfungsi tersebut.

```
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

Contoh Java berikut menunjukkan cara menggunakan DeleteCertificateAuthorityoperasi.

Operasi ini menghapus otoritas sertifikat pribadi (CA) yang Anda buat menggunakan CreateCertificateAuthorityoperasi. Operasi `DeleteCertificateAuthority` mengharuskan Anda membiarkan ARN agar dihapus CA. Anda dapat menemukan ARN dengan memanggil operasi. ListCertificateAuthorities Anda dapat langsung menghapus CA privat jika statusnya `CREATING` atau `PENDING_CERTIFICATE`. Namun, jika Anda telah mengimpor sertifikat, Anda tidak dapat langsung menghapusnya. Anda harus terlebih dahulu menonaktifkan CA dengan memanggil UpdateCertificateAuthorityoperasi dan mengatur `Status` parameter ke`DISABLED`. Anda kemudian dapat menggunakan parameter `PermanentDeletionTimeInDays` dalam operasi `DeleteCertificateAuthority`untuk menentukan jumlah hari, dari 7 hingga 30 hari. Selama periode tersebut, CA privat dapat dikembalikan ke status `disabled`. Secara default, jika Anda tidak mengatur parameter `PermanentDeletionTimeInDays`, maka masa pemulihannya 30 hari. Setelah masa ini kedaluwarsa, CA privat dihapus secara permanen dan tidak dapat dipulihkan. Untuk informasi selengkapnya, lihat Memulihkan CA.

Untuk contoh Java yang menunjukkan cara menggunakan RestoreCertificateAuthorityoperasi, lihatRestoreCertificateAuthority.

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.DeleteCertificateAuthorityRequest;

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

public class DeleteCertificateAuthority {
```

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

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

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

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

      // Create a requrest object and set the ARN of the private CA to delete.
      DeleteCertificateAuthorityRequest req = new DeleteCertificateAuthorityRequest();

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

      // Set the recovery period.
      req.withPermanentDeletionTimeInDays(12);

      // Delete the CA.
      try {
         client.deleteCertificateAuthority(req);
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
```

```
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (RequestFailedException ex) {
         throw ex;
      }
   }
}
```

DeletePermission

Contoh Java berikut menunjukkan cara menggunakan DeletePermissionoperasi.

Operasi menghapus izin yang didelegasikan CA pribadi ke kepala AWS layanan menggunakan operasi. CreatePermissions Anda dapat menemukan ARN CA dengan memanggil fungsi. ListCertificateAuthorities Anda dapat memeriksa izin yang diberikan CA dengan memanggil fungsi. ListPermissions

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.DeletePermissionRequest;
import com.amazonaws.services.acmpca.model.DeletePermissionResult;

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

public class DeletePermission {

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

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
```

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

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

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

      // Create a request object.
      DeletePermissionRequest req =
          new DeletePermissionRequest();

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

      // Set the AWS service principal.
      req.setPrincipal("acm.amazonaws.com");

      // Create a result object.
      DeletePermissionResult result = null;
      try {
         result = client.deletePermission(req);
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (RequestFailedException ex) {
         throw ex;
```

```
      } catch (ResourceNotFoundException ex) {
         throw ex;
      }
   }
}
```

DeletePolicy

Contoh Java berikut menunjukkan cara menggunakan DeletePolicyoperasi.

Operasi ini menghapus kebijakan berbasis sumber daya yang melekat pada CA privat. Kebijakan berbasis sumber daya digunakan untuk mengaktifkan pembagian lintas akun CA. Anda dapat menemukan ARN CA pribadi dengan memanggil tindakan. ListCertificateAuthorities

Tindakan API terkait termasuk PutPolicydan 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

Contoh Java berikut menunjukkan cara menggunakan DescribeCertificateAuthorityoperasi.

Operasi ini mencantumkan informasi tentang Private Certificate Authority (CA) Anda. Anda harus menentukan ARN (Nama Sumber Daya Amazon) dari CA privat. Output berisi status CA Anda. Status tersebut dapat berupa salah satu dari hal berikut:

- `CREATING`— AWS Private CA adalah menciptakan otoritas sertifikat pribadi Anda.
- `PENDING_CERTIFICATE` – Sertifikat sedang ditunda. Anda harus menggunakan CA akar on premise atau bawahan untuk menandatangani CSR CA privat Anda, lalu mengimpornya ke PCA.
- `ACTIVE` – CA privat Anda sedang aktif.
- `DISABLED` – CA privat Anda telah dinonaktifkan.
- `EXPIRED` – Sertifikat CA privat Anda telah kedaluwarsa.
- `FAILED` – CA privat Anda tidak dapat dibuat.
- `DELETED` – CA privat Anda berada dalam masa pemulihan, yang setelahnya akan dihapus secara permanen.

```
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

Contoh Java berikut menunjukkan cara menggunakan DescribeCertificateAuthorityAuditReportoperasi.

Operasi mencantumkan informasi tentang laporan audit tertentu yang Anda buat dengan memanggil CreateCertificateAuthorityAuditReportoperasi. Informasi audit dibuat setiap kali kunci privat otoritas sertifikasi (CA) digunakan. Kunci privat digunakan ketika Anda menerbitkan sertifikat, menandatangani CRL, atau mencabut sertifikat.

```
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);
    }
}
```

Output Anda harus serupa dengan berikut ini:

```
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

Contoh Java berikut menunjukkan cara menggunakan GetCertificateoperasi.

Operasi ini mengambil sertifikat dari CA privat Anda. ARN sertifikat dikembalikan saat Anda memanggil operasi. IssueCertificate Anda harus menentukan ARN CA privat Anda dan ARN sertifikat yang dikeluarkan saat memanggil operasi `GetCertificate`. Anda dapat mengambil sertifikat jika dalam keadaan `ISSUED`. Anda dapat menghubungi CreateCertificateAuthorityAuditReportoperasi untuk membuat laporan yang berisi informasi tentang semua sertifikat yang dikeluarkan dan dicabut oleh CA pribadi Anda.

```
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);
    }
}
```

Output Anda harus berupa rantai sertifikat yang mirip dengan berikut ini untuk otoritas sertifikasi (CA) dan sertifikat yang Anda tentukan.

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

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

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

GetCertificateAuthorityCertificate

Contoh Java berikut menunjukkan cara menggunakan GetCertificateAuthorityCertificateoperasi.

Operasi ini mengambil sertifikat dan rantai sertifikat untuk Private Certificate Authority (CA) Anda. Sertifikat dan rantai adalah string berkode base64 dalam format PEM. Rantai tidak menyertakan sertifikat CA. Setiap sertifikat dalam rantai menandatangani yang sebelumnya.

```
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);
    }
}
```

Output Anda harus berupa sertifikat dan rantai yang mirip dengan berikut ini untuk otoritas sertifikasi (CA) yang Anda tentukan.

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

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

GetCertificateAuthorityCsr

Contoh Java berikut menunjukkan cara menggunakan GetCertificateAuthorityCsroperasi.

Operasi ini mengambil sertifikat permintaan penandatanganan (CSR) untuk Private Certificate Authority (CA) Anda. CSR dibuat saat Anda memanggil CreateCertificateAuthorityoperasi. Bawa CSR ke infrastruktur X.509 on premise Anda dan tandatangani menggunakan CA akar atau bawahan. Kemudian impor sertifikat yang ditandatangani kembali ke ACM PCA dengan memanggil operasi. ImportCertificateAuthorityCertificate CSR dikembalikan sebagai string berkode base64 dalam format PEM.

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
```

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

public class GetCertificateAuthorityCsr {

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

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

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

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

      // Create the request object and set the CA ARN.
      GetCertificateAuthorityCsrRequest req = new GetCertificateAuthorityCsrRequest();
```

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

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

      // Retrieve the CSR.
      GetCertificateAuthorityCsrResult result = null;
      try {
         result = client.getCertificateAuthorityCsr(req);
      } catch (RequestInProgressException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (RequestFailedException ex) {
         throw ex;
      }

      // Retrieve and display the CSR;
      String Csr = result.getCsr();
      System.out.println(Csr);
   }
}
```

Output Anda harus serupa dengan berikut ini untuk otoritas sertifikasi (CA) yang Anda tentukan. Permintaan penandatanganan sertifikat (CSR) adalah berkode base64 dalam format PEM. Simpan ke file on premise, kirim ke infrastruktur X.509 on premise Anda, dan tanda tangani dengan menggunakan CA akar atau bawahan.

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

GetPolicy

Contoh Java berikut menunjukkan cara menggunakan GetPolicyoperasi.

Operasi mengambil kebijakan berbasis sumber daya yang melekat pada CA privat. Kebijakan berbasis sumber daya digunakan untuk mengaktifkan berbagi lintas akun CA. Anda dapat menemukan ARN CA pribadi dengan memanggil tindakan. ListCertificateAuthorities

Tindakan API terkait termasuk PutPolicydan 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

Contoh Java berikut menunjukkan cara menggunakan ImportCertificateAuthorityCertificateoperasi.

Operasi ini mengimpor sertifikat CA pribadi Anda yang ditandatangani ke dalamAWS Private CA. Sebelum Anda dapat memanggil operasi ini, Anda harus membuat otoritas sertifikat pribadi dengan memanggil CreateCertificateAuthorityoperasi. Anda kemudian harus membuat permintaan penandatanganan sertifikat (CSR) dengan memanggil GetCertificateAuthorityCsroperasi. Ambil CSR ke CA on premise dan gunakan sertifikat akar atau sertifikat bawahan untuk menandatanganinya. Membuat rantai sertifikat dan menyalin sertifikat ditandatangani dan rantai sertifikat ke direktori kerja Anda.

```
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

Contoh Java berikut menunjukkan cara menggunakan IssueCertificateoperasi.

Operasi ini menggunakan Private Certificate Authority (CA) untuk menerbitkan sertifikat entitas akhir. Operasi ini mengembalikan Amazon Resource Name (ARN) sertifikat. Anda dapat mengambil sertifikat dengan memanggil GetCertificatedan menentukan ARN.

> **Note**
>
> IssueCertificateOperasi mengharuskan Anda untuk menentukan template sertifikat. Contoh ini menggunakan hal berikut: templat `EndEntityCertificate/V1`. Untuk informasi tentang semua templat yang tersedia, lihat Memahami templat sertifikat.

```
package com.amazonaws.samples;

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

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

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

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

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

public class IssueCertificate {
   public static ByteBuffer stringToByteBuffer(final String string) {
      if (Objects.isNull(string)) {
```

```
            return null;
        }
        byte[] bytes = string.getBytes(StandardCharsets.UTF_8);
        return ByteBuffer.wrap(bytes);
        }

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

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

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

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

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

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

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

```
        ByteBuffer csrByteBuffer = stringToByteBuffer(strCSR);
        req.setCsr(csrByteBuffer);

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

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

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

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

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

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

Output Anda harus serupa dengan berikut ini:

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

ListCertificateAuthorities

Sampel Java berikut menunjukkan cara menggunakanListCertificateAuthoritiesoperasi.

Operasi ini mencantumkan Private Certificate Authorate (CA) yang Anda buat menggunakanCreateCertificateAuthorityoperasi.

```
package com.amazonaws.samples;

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

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

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ListCertificateAuthoritiesRequest;
import com.amazonaws.services.acmpca.model.ListCertificateAuthoritiesResult;
import com.amazonaws.services.acmpca.model.InvalidNextTokenException;

public class ListCertificateAuthorities {

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

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

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

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

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

        // Create the request object.
        ListCertificateAuthoritiesRequest req = new ListCertificateAuthoritiesRequest();
        req.withMaxResults(10);

        // Retrieve a list of your CAs.
        ListCertificateAuthoritiesResult result= null;
        try {
            result = client.listCertificateAuthorities(req);
        } catch (InvalidNextTokenException ex) {
            throw ex;
        }

        // Display the CA list.
        System.out.println(result.getCertificateAuthorities());
    }
}
```

Jika Anda memiliki otoritas sertifikasi (CA) untuk dicantumkan, output Anda harus serupa dengan berikut ini:

```
[{
 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

Contoh Java berikut menunjukkan cara menggunakan ListPermissionsoperasi.

Operasi ini mencantumkan izin, jika ada, bahwa CA privat Anda telah ditetapkan. Izin`IssueCertificate`, termasuk`GetCertificate`,`ListPermissions`, dan, dapat ditetapkan ke kepala AWS layanan dengan CreatePermissionoperasi, dan dicabut dengan operasi. DeletePermissions

```
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);
   }
}
```

Jika CA privat yang ditunjuk telah menetapkan izin untuk layanan utama, output Anda harus mirip dengan berikut ini:

```
[{
       Arn: arn:aws:acm-
pca:region:account:permission/12345678-1234-1234-1234-123456789012,
       CreatedAt: WedFeb0317: 05: 39PST2019,
       Prinicpal: acm.amazonaws.com,
       Permissions: {
            ISSUE_CERTIFICATE,
            GET_CERTIFICATE,
            DELETE,CERTIFICATE
       },
       SourceAccount: account
}]
```

ListTags

Contoh Java berikut menunjukkan cara menggunakan ListTagsoperasi.

Operasi ini mencantumkan tag, jika ada, yang terkait dengan CA privat Anda. Tag adalah label yang dapat Anda gunakan untuk mengidentifikasi dan mengatur CA Anda. Setiap tanda terdiri dari kunci

dan nilai opsional. Panggil TagCertificateAuthorityoperasi untuk menambahkan satu atau beberapa tag ke CA Anda. Panggil UntagCertificateAuthorityoperasi untuk menghapus tag.

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.ListTagsRequest;
import com.amazonaws.services.acmpca.model.ListTagsResult;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArnException;

public class ListTags {

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

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

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

      // Create a client that you can use to make requests.
```

```
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
          .withEndpointConfiguration(endpoint)
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
          .build();

      // Create a request object and set the CA ARN.
      ListTagsRequest req = new ListTagsRequest();
      req.withCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // List the tags
      ListTagsResult result = null;
      try {
          result = client.listTags(req);
      } catch (InvalidArnException ex) {
          throw ex;
      } catch (ResourceNotFoundException ex) {
          throw ex;
      }

      // Retrieve and display the tags.
      System.out.println(result);
   }
}
```

Jika Anda memiliki tag untuk daftar, output Anda harus serupa dengan berikut ini:

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

PutPolicy

Contoh Java berikut menunjukkan cara menggunakan PutPolicyoperasi.

Operasi melampirkan kebijakan berbasis sumber daya ke CA privat, memungkinkan berbagi lintas-akun. Ketika disahkan oleh kebijakan, prinsip yang berada di akun AWS lain dapat menerbitkan dan memperbarui sertifikat entitas akhir privat menggunakan CA privat yang tidak dimilikinya. Anda dapat menemukan ARN CA pribadi dengan memanggil tindakan. ListCertificateAuthorities Untuk contoh kebijakan, lihat AWS Private CA panduan tentang Kebijakan Berbasis Sumber Daya.

Setelah kebijakan dilampirkan ke CA, Anda dapat memeriksanya dengan GetPolicytindakan atau menghapusnya dengan DeletePolicytindakan tersebut.

```
package com.amazonaws.samples;

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

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

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.PutPolicyRequest;
import com.amazonaws.services.acmpca.model.PutPolicyResult;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.LockoutPreventedException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class PutPolicy {

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

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

        // Define the endpoint for your sample.
```

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

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

        // Create the request object.
        PutPolicyRequest req = new PutPolicyRequest();

        // Set the resource ARN.
        req.withResourceArn("arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566");

        // Import and set the policy.
        // Note: This code assumes the file "ShareResourceWithAccountPolicy.json" is in
 a folder titled policy.
        String policy = new String(Files.readAllBytes(Paths.get("policy",
 "ShareResourceWithAccountPolicy.json")));
        req.withPolicy(policy);

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

```
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (AWSACMPCAException ex) {
            throw ex;
        }
    }
}
```

RestoreCertificateAuthority

Contoh Java berikut menunjukkan cara menggunakan RestoreCertificateAuthorityoperasi. CA privat dapat dipulihkan kapan saja selama masa pemulihan. Sekarang, masa ini dapat berlangsung 7 hingga 30 hari dari tanggal penghapusan dan dapat ditentukan ketika Anda menghapus CA. Untuk informasi lebih lanjut, lihat Memulihkan CA. Lihat juga contoh Java DeleteCertificateAuthority.

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.RestoreCertificateAuthorityRequest;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;

public class RestoreCertificateAuthority {

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

      // Retrieve your credentials from the C:\Users\name\.aws\credentials file
      // in Windows or the .aws/credentials file in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider("default").getCredentials();
```

```
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load your credentials from file.",
 e);
        }

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

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

        // Create the request object.
        RestoreCertificateAuthorityRequest req = new
 RestoreCertificateAuthorityRequest();

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

        // Restore the CA.
        try {
            client.restoreCertificateAuthority(req);
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        }
    }
}
```

RevokeCertificate

Contoh Java berikut menunjukkan cara menggunakan RevokeCertificateoperasi.

Operasi ini mencabut sertifikat yang Anda keluarkan dengan memanggil IssueCertificateoperasi. Jika Anda mengaktifkan daftar pencabutan sertifikat (CRL) saat membuat atau memperbarui CA pribadi Anda, informasi tentang sertifikat yang dicabut disertakan dalam CRL. AWS Private CAmenulis CRL ke bucket Amazon S3 yang Anda tentukan. Untuk informasi lebih lanjut, lihat CrlConfigurationstrukturnya.

```
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

Contoh Java berikut menunjukkan cara menggunakan TagCertificateAuthorityoperasi.

Operasi ini menambahkan satu atau lebih tag untuk CA privat Anda. Tag adalah label yang dapat Anda gunakan untuk mengidentifikasi dan mengatur sumber daya AWS Anda. Setiap tag terdiri dari kunci dan nilai opsional. Ketika Anda memanggil operasi ini, Anda menentukan CA privat oleh Amazon Resource Name (ARN). Anda menentukan tag dengan menggunakan pasangan nilai kunci. Untuk mengidentifikasi karakteristik tertentu dari CA itu, Anda dapat menerapkan tag untuk hanya satu CA privat. Atau, untuk memfilter hubungan umum di antara CAs tersebut, Anda dapat menerapkan tag yang sama untuk beberapa CA privat. Untuk menghapus satu atau lebih tag, gunakan UntagCertificateAuthorityoperasi. Panggil ListTagsoperasi untuk melihat tag apa yang terkait dengan CA Anda.

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.TagCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.Tag;

import java.util.ArrayList;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidTagException;
import com.amazonaws.services.acmpca.model.TooManyTagsException;

public class TagCertificateAuthorities {

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

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

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

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

      // Create a tag - method 1
      Tag tag1 = new Tag();
      tag1.withKey("Administrator");
      tag1.withValue("Bob");

      // Create a tag - method 2
      Tag tag2 = new Tag()
          .withKey("Purpose")
          .withValue("WebServices");

      // Add the tags to a collection.
      ArrayList<Tag> tags = new ArrayList<Tag>();
      tags.add(tag1);
      tags.add(tag2);

      // Create a request object and specify the certificate authority ARN.
      TagCertificateAuthorityRequest req = new TagCertificateAuthorityRequest();
      req.setCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");
      req.setTags(tags);
```

```
        // Add a tag
        try {
            client.tagCertificateAuthority(req);
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidTagException ex) {
            throw ex;
        } catch (TooManyTagsException ex) {
            throw ex;
        }
    }
}
```

UntagCertificateAuthority

Contoh Java berikut menunjukkan cara menggunakan UntagCertificateAuthorityoperasi.

Operasi ini menghapus satu atau lebih tag dari CA privat Anda. Satu tag terdiri dari pasangan nilai kunci. Jika Anda tidak menentukan bagian nilai tag saat memanggil operasi ini, tag akan dihapus, terlepas dari nilainya. Jika Anda menetapkan nilai, tag akan dihapus hanya jika dikaitkan dengan nilai yang ditentukan. Untuk menambahkan tag ke CA pribadi, gunakan TagCertificateAuthorityoperasi. Panggil ListTagsoperasi untuk melihat tag apa yang terkait dengan CA Anda.

```
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

Contoh Java berikut menunjukkan cara menggunakan UpdateCertificateAuthorityoperasi.

Operasi ini memperbarui status atau konfigurasi Private Certificate Authority (CA). CA privat Anda harus dalam keadaan `ACTIVE` atau `DISABLED` sebelum Anda dapat memperbaruinya. Anda dapat menonaktifkan CA privat yang ada dalam keadaan `ACTIVE` atau membuat CA yang dalam status aktif `DISABLED` lagi.

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.UpdateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CertificateAuthorityStatus;

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

```
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;

public class UpdateCertificateAuthority {

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

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

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

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

      // Create the request object.
      UpdateCertificateAuthorityRequest req = new UpdateCertificateAuthorityRequest();

      // Set the ARN of the private CA that you want to update.
      req.setCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Define the certificate revocation list configuration. If you do not want to
```

```
        // update the CRL configuration, leave the CrlConfiguration structure alone and
        // do not set it on your UpdateCertificateAuthorityRequest object.
        CrlConfiguration crlConfigure = new CrlConfiguration();
        crlConfigure.withEnabled(true);
        crlConfigure.withExpirationInDays(365);
        crlConfigure.withCustomCname("your-custom-name");
        crlConfigure.withS3BucketName("your-bucket-name");

        // Set the CRL configuration onto your UpdateCertificateAuthorityRequest object.
        // If you do not want to change your CRL configuration, do not use the
        // setCrlConfiguration method.
        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);
        req.setRevocationConfiguration(revokeConfig);

        // Set the status.
        req.withStatus(CertificateAuthorityStatus.<<ACTIVE>>);

        // Create the result object.
        try {
            client.updateCertificateAuthority(req);
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        }
    }
}
```

Buat CA dan sertifikat dengan nama subjek khusus

KlasterCustomAttributeobjek memungkinkan administrator untuk lulus pengenal objek kustom (OID) untuk CA pribadi dan sertifikat. ID kustom dapat digunakan untuk membuat hierarki subjek-nama khusus yang mencerminkan struktur dan kebutuhan organisasi Anda. Sertifikat yang disesuaikan

harus dibuat menggunakan salah satu`ApiPassthrough`templat. Untuk informasi lebih lanjut tentang templat, lihat Variasi templat. Untuk informasi selengkapnya tentang penggunaan attrributes khusus, lihatMenerbitkan sertifikat entitas akhir pribadidanProsedur untuk membuat CA (CLI) .

Anda tidak dapat menggunakan`StandardAttributes`dalam hubungannya dengan`CustomAttributes`. Namun, Anda dapat melewati ID standar sebagai bagian dari`CustomAttributes`. Nama subjek default tercantum dalam tabel berikut:

Nama Subjek	ID objek
Negara	2.5.4.6
CommonName	2.5.4.3
DistinguishedNameQualifier	2.5.4.46
GenerationQualifier	2.5.4.44
GivenName	2.5.4.42
Inisial	2.5.4.43
Lokalitas	2.5.4.7
Organisasi	2.5.4.10
OrganizationalUnit	2.5.4.11
Nama samaran	2.5.4.65
SerialNumber	2.5.4.5
Status	2.5.4.8
Nama keluarga	2.5.4.4
Judul	2.5.4.12

Topik

- Buat CA dengan CustomAttribute

- Menerbitkan sertifikat dengan CustomAttribute

Buat CA dengan CustomAttribute

```
package com.amazonaws.samples;

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


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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;


public class CreateCertificateAuthorityWithCustomAttributes {

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

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

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

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

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("2.5.4.6") // Country
                .withValue("US"),
            new CustomAttribute()
                .withObjectIdentifier("2.5.4.3") // CommonName
                .withValue("CommonName"),
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1") // CustomOID
                .withValue("ABCDEFG"),
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1") // CustomOID
                .withValue("BCDEFGH")
        );
```

```
        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);

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

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

        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);

        // Define a certificate authority type: ROOT or SUBORDINATE
        CertificateAuthorityType caType = CertificateAuthorityType.SUBORDINATE;

        // Create a tag - method 1
        Tag tag1 = new Tag();
        tag1.withKey("PrivateCA");
        tag1.withValue("Sample");

        // Create a tag - method 2
        Tag tag2 = new Tag()
            .withKey("Purpose")
            .withValue("WebServices");

        // Add the tags to a collection.
        ArrayList<Tag> tags = new ArrayList<Tag>();
        tags.add(tag1);
        tags.add(tag2);

        // Create the request object.
        CreateCertificateAuthorityRequest req = new
CreateCertificateAuthorityRequest();
        req.withCertificateAuthorityConfiguration(configCA);
```

```
        req.withRevocationConfiguration(revokeConfig);
        req.withIdempotencyToken("1234");
        req.withCertificateAuthorityType(caType);
        req.withTags(tags);


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

        // Retrieve the ARN of the private CA.
        String arn = result.getCertificateAuthorityArn();
        System.out.println(arn);
    }
}
```

Menerbitkan sertifikat dengan CustomAttribute

```
package com.amazonaws.samples;

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

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

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

```
import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

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

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

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

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

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

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

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

      // Set the CA ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
         "certificate-authority/12345678-1234-1234-1234-123456789012");

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

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

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

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

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

      // Define custom attributes
      List<CustomAttribute> customAttributes = Arrays.asList(
         new CustomAttribute()
               .withObjectIdentifier("2.5.4.6") // Country
               .withValue("US"),
```

```
            new CustomAttribute()
                  .withObjectIdentifier("2.5.4.3") // CommonName
                  .withValue("CommonName"),
            new CustomAttribute()
                  .withObjectIdentifier("1.3.6.1.4.1") // CustomOID
                  .withValue("ABCDEFG"),
            new CustomAttribute()
                  .withObjectIdentifier("1.3.6.1.4.1") // CustomOID
                  .withValue("BCDEFGH")
      );

      // Define certificate subject.
      ASN1Subject subject = new ASN1Subject();
      subject.setCustomAttributes(customAttributes);

      // Add subject to api-passthrough
      ApiPassthrough apiPassthrough = new ApiPassthrough();
      apiPassthrough.setSubject(subject);
      req.setApiPassthrough(apiPassthrough);

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

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

Membuat sertifikat dengan ekstensi khusus

KlasterCustomExtensionobjek memungkinkan administrator untuk mengatur ekstensi X.509 kustom dalam sertifikat pribadi. Sertifikat yang disesuaikan harus dibuat menggunakan salah satu ApiPassthrough Templat. Untuk informasi lebih lanjut tentang templat, lihat Variasi templat. Untuk informasi lebih lanjut tentang penggunaan ekstensi kustom, lihatMenerbitkan sertifikat entitas akhir pribadi.

Topik

- Aktifkan CA bawahan dengan NameConstraints luas
- Mengeluarkan sertifikat dengan ekstensi pernyataan QC

Aktifkan CA bawahan dengan NameConstraints luas

```
package com.amazonaws.samples;

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

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

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

import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;
import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
```

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

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

import lombok.SneakyThrows;

public class SubordinateCAActivationWithNameConstraints {
   public static void main(String[] args) throws Exception {
      // Place your own Root CA ARN here.
```

```
      String rootCAArn = "arn:aws:acm-pca:region:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012";

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

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

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

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

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

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

```
  }

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

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

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

      return client;
  }

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

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

      // Create a result object.
      GetCertificateAuthorityCertificateResult getCACertificateResult = null;
      try {
```

```
            getCACertificateResult =
client.getCertificateAuthorityCertificate(getCACertificateRequest);
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        }

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

        return rootCertificate;
    }

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

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

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

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

        return subordinateCAArn;
    }

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

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

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

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        // Retrieve and display the CSR;
```

```
        String csr = csrResult.getCsr();
        System.out.println("Subordinate CSR:");
        System.out.println(csr);

        return csr;
    }

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

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

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

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

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

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

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

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

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

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

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

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

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

        return subordinateCertificateArn;
    }

    @SneakyThrows
    private static String getNameConstraintExtensionValue() {
       // Generate Base64 encoded Nameconstraints extension value
       GeneralSubtree dnsPrivate = new GeneralSubtree(new
GeneralName(GeneralName.dNSName, ".private"));
       GeneralSubtree dnsLocal = new GeneralSubtree(new GeneralName(GeneralName.dNSName,
".local"));
       GeneralSubtree dnsCorp = new GeneralSubtree(new GeneralName(GeneralName.dNSName,
".corp"));
       GeneralSubtree dnsSecretCorp = new GeneralSubtree(new
GeneralName(GeneralName.dNSName, ".secret.corp"));
```

```
     GeneralSubtree dnsExample = new GeneralSubtree(new
GeneralName(GeneralName.dNSName, ".example.com"));
     GeneralSubtree[] permittedSubTree = new GeneralSubtree[] { dnsPrivate, dnsLocal,
dnsCorp };
     GeneralSubtree[] excludedSubTree = new GeneralSubtree[] { dnsSecretCorp,
dnsExample };
     NameConstraints nameConstraints = new NameConstraints(permittedSubTree,
excludedSubTree);

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

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

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

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

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

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

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

```
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

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

        return subordinateCertificate;
    }

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

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

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

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

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
```

```
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
        System.out.println("Subordinate CA certificate successfully imported.");
        System.out.println("Subordinate CA activated successfully.");
    }

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

Mengeluarkan sertifikat dengan ekstensi pernyataan QC

```
package com.amazonaws.samples;

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

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.Objects;

import com.amazonaws.services.acmpca.AWSACMPCA;
import com.amazonaws.services.acmpca.AWSACMPCAClientBuilder;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
```

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

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

import org.bouncycastle.asn1.ASN1EncodableVector;
import org.bouncycastle.asn1.ASN1ObjectIdentifier;
import org.bouncycastle.asn1.DERSequence;
import org.bouncycastle.asn1.DERUTF8String;
import org.bouncycastle.asn1.x509.qualified.ETSIQCObjectIdentifiers;
import org.bouncycastle.asn1.x509.qualified.QCStatement;

import lombok.SneakyThrows;

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

   @SneakyThrows
   private static String generateQCStatementBase64ExtValue() {
      DERSequence qcTypeSeq = new DERSequence(ETSIQCObjectIdentifiers.id_etsi_qct_web);
      QCStatement qcType = new QCStatement(ETSIQCObjectIdentifiers.id_etsi_qcs_QcType,
 qcTypeSeq);

      ASN1EncodableVector pspAIVector = new ASN1EncodableVector(2);
      pspAIVector.add(new ASN1ObjectIdentifier("0.4.0.19495.1.3"));
      pspAIVector.add(new DERUTF8String("PSP_AI"));
      DERSequence pspAISeq = new DERSequence(pspAIVector);

      ASN1EncodableVector pspASVector = new ASN1EncodableVector(2);
      pspASVector.add(new ASN1ObjectIdentifier("0.4.0.19495.1.1"));
```

```
      pspASVector.add(new DERUTF8String("PSP_AS"));
      DERSequence pspASSeq = new DERSequence(pspASVector);

      ASN1EncodableVector pspPIVector = new ASN1EncodableVector(2);
      pspPIVector.add(new ASN1ObjectIdentifier("0.4.0.19495.1.2"));
      pspPIVector.add(new DERUTF8String("PSP_PI"));
      DERSequence pspPISeq = new DERSequence(pspPIVector);

      ASN1EncodableVector pspICVector = new ASN1EncodableVector(2);
      pspICVector.add(new ASN1ObjectIdentifier("0.4.0.19495.1.4"));
      pspICVector.add(new DERUTF8String("PSP_IC"));
      DERSequence pspICSeq = new DERSequence(pspICVector);

      ASN1EncodableVector pspSeqVector = new ASN1EncodableVector(4);
      pspSeqVector.add(pspPISeq);
      pspSeqVector.add(pspICSeq);
      pspSeqVector.add(pspASSeq);
      pspSeqVector.add(pspAISeq);
      DERSequence pspSeq = new DERSequence(pspSeqVector);

      ASN1EncodableVector pspVector = new ASN1EncodableVector(3);
      pspVector.add(pspSeq);
      pspVector.add(new DERUTF8String("Your Financial Authority"));
      pspVector.add(new DERUTF8String("AB-CD"));
      DERSequence psp = new DERSequence(pspVector);
      QCStatement qcPSP = new QCStatement(new ASN1ObjectIdentifier("0.4.0.19495.2"),
psp);

      DERSequence qcSeq = new DERSequence(new QCStatement[] { qcType, qcPSP });

      byte[] qcExtValueInBytes = qcSeq.getEncoded();
      return Base64.getEncoder().encodeToString(qcExtValueInBytes);
   }

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

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

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

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

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

      // Set the CA ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
         "certificate-authority/12345678-1234-1234-1234-123456789012");

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

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

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

      // Set the validity period for the certificate to be issued.
      Validity validity = new Validity();
      validity.withValue(30L);
      validity.withType("DAYS");
      req.withValidity(validity);
```

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

      // Generate Base64 encoded extension value for QC Statement
      String base64EncodedExtValue = generateQCStatementBase64ExtValue();

      // Generate custom extension
      CustomExtension customExtension = new CustomExtension();
      customExtension.setObjectIdentifier("1.3.6.1.5.5.7.1.3"); // QC Statement
 Extension OID
      customExtension.setValue(base64EncodedExtValue);

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

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

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

Menggunakan AWS Private CA API untuk mengimplementasikan standar Matter (contoh Java)

Anda dapat menggunakan AWS Private Certificate Authority API untuk membuat sertifikat yang sesuai dengan standar konektivitas Matter. Matter menentukan konfigurasi sertifikat yang meningkatkan keamanan dan konsistensi perangkat Internet of Things (IoT) di berbagai platform teknik. Untuk informasi lebih lanjut tentang Matter, lihat buildwithmatter.com.

Contoh Java di bagian ini berinteraksi dengan layanan dengan mengirimkan permintaan HTTP. Layanan tersebut mengembalikan tanggapan HTTP. Untuk informasi selengkapnya, lihat Referensi AWS Private Certificate Authority API.

Selain HTTP API, Anda dapat menggunakan AWS SDK dan alat baris perintah untuk berinteraksiAWS Private CA. Hal ini direkomendasikan melalui API HTTP. Untuk informasi lebih lanjut, lihat Alat untuk Layanan Web Amazon. Topik berikut menunjukkan cara menggunakan AWS SDK for Javauntuk memprogram AWS Private CA API.

Itu GetCertificateAuthorityCsr, GetCertificate, dan DescribeCertificateAuthorityAuditReportoperasi mendukung pelayan. Anda dapat menggunakan penunggu untuk mengontrol perkembangan kode Anda berdasarkan kehadiran atau keadaan sumber daya tertentu. Untuk informasi selengkapnya, lihat topik berikut, serta Pelayan AWS SDK for Java di Blog AWS Pengembang.

Matter 1.2, dirilis pada Oktober 2023, mendukung pencabutan DAC menggunakan Daftar Pencabutan Sertifikat (CRL). Untuk membantu Anda menyesuaikan dengan standar Matter saat ini, saat Anda mengaktifkan pencabutan CRL untuk CA yang mengeluarkan sertifikat Matter, di `CrlConfiguration` objek, dalam `CrlDistributionPointExtensionConfiguration` struktur, disetel ke. `OmitExtension` `true`

Biasanya, CA menanamkan Titik Distribusi CRL (CDP) dalam sertifikat yang mereka terbitkan sehingga pihak yang mengandalkan yang melakukan validasi rantai sertifikat dapat mengambil CRL dan memeriksa status sertifikat. Dalam Matter, URI CDP tidak ditulis ke sertifikat. Sebagai gantinya, pengguna mengambil CDP dari Matter Distributed Compliance Ledger (DCL), penyimpanan data Matter tepercaya. Anda harus mengunggah URI CDP ke Matter DCL sehingga dapat ditemukan saat memvalidasi DAC. Untuk informasi selengkapnya tentang menentukan URI CDP, lihatMenentukan URI Titik Distribusi CRL (CDP) . Untuk informasi lebih lanjut tentang Materi, lihat dokumentasi Matter DCL.

Topik

- Aktifkan Otoritas Pengesahan Produk (PAA)
- Aktifkan Product Attestation Intermediate (PAI)
- Membuat Sertifikat Pengesahan Perangkat (DAC)
- Aktifkan Root CA untuk Node Operational Certificates (NOC).
- Aktifkan CA Bawahan untuk Sertifikat Operasional Node (NOC)
- Membuat Node Operational Certificate (NOC)

Aktifkan Otoritas Pengesahan Produk (PAA)

Contoh Java ini menunjukkan cara menggunakan Definisi RootCACertificate_APIPassthrough/V1 template untuk membuat dan menginstal sertifikat Matter Root CA (PAA) untuk pengesahan produk. Ekstensi AuthorityKeyIdentifier (AKI) adalah opsional untuk PaaS. Untuk menyetel AKI, Anda harus menghasilkan nilai AKI yang dikodekan Base64 dan meneruskannya melalui a. CustomExtension

Contoh ini memanggil tindakan AWS Private CA API berikut:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Jika Anda mengalami masalah, lihat Menggunakan standar Matter di bagian Pemecahan Masalah.

```
package com.amazonaws.samples.matter;

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


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

```
import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CrlDistributionPointExtensionConfiguration;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
```

```
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import org.bouncycastle.asn1.x509.SubjectPublicKeyInfo;
import org.bouncycastle.cert.jcajce.JcaX509ExtensionUtils;
import org.bouncycastle.openssl.PEMParser;
import org.bouncycastle.pkcs.PKCS10CertificationRequest;
import org.bouncycastle.util.io.pem.PemReader;

import lombok.SneakyThrows;

public class ProductAttestationAuthorityActivation {

    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "ap-
southeast-2"

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("2.5.4.3") // CommonName
                .withValue("Matter Test PAA"),
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1.37244.2.1") // Vendor ID
                .withValue("FFF1")
        );

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);

        // Define the CA configuration.
```

```
        CertificateAuthorityConfiguration configCA = new
CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.EC_prime256v1);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
        configCA.withSubject(subject);

        // Define a CRL distribution point extension configuration
        CrlDistributionPointExtensionConfiguration CDPConfigure = new
CrlDistributionPointExtensionConfiguration();
        CDPConfigure.withOmitExtension(true);

        // Define a certificate revocation list configuration.
        CrlConfiguration crlConfigure = new CrlConfiguration();
        crlConfigure.withEnabled(true);
        crlConfigure.withExpirationInDays(365);
        crlConfigure.withCustomCname(null);
        crlConfigure.withS3BucketName("your-bucket-name");
        crlConfigure.withS3ObjectAcl("BUCKET_OWNER_FULL_CONTROL");
        crlConfigure.withCrlDistributionPointExtensionConfiguration(CDPConfigure);

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.ROOT;

        // ** Execute core code samples for Root CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCAArn = CreateCertificateAuthority(configCA, crlConfigure, CAtype,
client);
        String csr = GetCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = IssueCertificate(rootCAArn, csr, client);
        String rootCertificate = GetCertificate(rootCertificateArn, rootCAArn, client);
        ImportCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

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

```
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                    e);
        }

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

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

        return client;
    }

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

        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);
        createCARequest.withRevocationConfiguration(revokeConfig);

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

```
        }

        // Retrieve the ARN of the private CA.
        String rootCAArn = createCAResult.getCertificateAuthorityArn();
        System.out.println("Product Attestation Authority (PAA) Arn: " + rootCAArn);

        return rootCAArn;
    }

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

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

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

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
```

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

        return csr;
    }

    @SneakyThrows
    private static String generateAuthorityKeyIdentifier(final String csrPEM) {
        PKCS10CertificationRequest csr = getPKCS10CertificationRequest(csrPEM);
        SubjectPublicKeyInfo spki = csr.getSubjectPublicKeyInfo();

        JcaX509ExtensionUtils extensionUtils = new JcaX509ExtensionUtils();
        byte[] akiBytes =
extensionUtils.createAuthorityKeyIdentifier(spki).getEncoded();

        return Base64.getEncoder().encodeToString(akiBytes);
    }

    @SneakyThrows
    private static PKCS10CertificationRequest getPKCS10CertificationRequest(final
String csrPEM) {
        ByteArrayInputStream bais = new ByteArrayInputStream(csrPEM.getBytes());
        PemReader pemReader = new PemReader(new InputStreamReader(bais));
        PEMParser parser = new PEMParser(pemReader);
        Object o = parser.readObject();
        if (o instanceof PKCS10CertificationRequest) {
            return (PKCS10CertificationRequest) o;
        }
        return null;
    }

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

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

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

        // Set the template ARN.
```

```
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/
RootCACertificate_APIPassthrough/V1");

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

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

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

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

        // Generate Base64 encoded extension value for AuthorityKeyIdentifier
        String base64EncodedExtValue = generateAuthorityKeyIdentifier(csr);

        // Generate custom extension
        CustomExtension customExtension = new CustomExtension();
        customExtension.setObjectIdentifier("2.5.29.35"); // AuthorityKeyIdentifier
 Extension OID
        customExtension.setValue(base64EncodedExtValue);

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

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

```
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Retrieve and display the certificate ARN.
        String rootCertificateArn = issueResult.getCertificateArn();
        System.out.println("Product Attestation Authority (PAA) Certificate Arn: " +
rootCertificateArn);

        return rootCertificateArn;
    }

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

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

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

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

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

        // Retrieve the certificate and certificate chain.
        GetCertificateResult certificateResult = null;
```

```
        try {
            certificateResult = client.getCertificate(certificateRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

        // Get the certificate and certificate chain and display the result.
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
    }

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

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

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

        importRequest.setCertificateChain(null);

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
```

```
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Product Attestation Authority (PAA) certificate
 successfully imported.");
        System.out.println("Product Attestation Authority (PAA) activated
 successfully.");
    }

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

Aktifkan Product Attestation Intermediate (PAI)

Contoh Java ini menunjukkan cara menggunakan BlankSubordinatePathLenCacertificate_0_apiPassthrough/definisi v1 template untuk membuat dan menginstal sertifikat Matter Subordinate CA (PAI) untuk pengesahan produk. Anda harus menghasilkan KeyUsage nilai yang dikodekan Base64 dan meneruskannya melalui a. CustomExtension

Contoh ini memanggil tindakan AWS Private CA API berikut:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

- GetCertificateAuthorityCertificate

Jika Anda mengalami masalah, lihat Menggunakan standar Matter di bagian Pemecahan Masalah.

```
package com.amazonaws.samples.matter;

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


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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CrlDistributionPointExtensionConfiguration;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateResult;
```

```
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import lombok.SneakyThrows;

public class ProductAttestationIntermediateActivation {

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

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

        // Define custom attributes
```

```
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("2.5.4.3") // CommonName
                .withValue("Matter Test PAI"),
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1.37244.2.1") // Vendor ID
                .withValue("FFF1"),
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1.37244.2.2") // Product ID
                .withValue("8000")
        );

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.EC_prime256v1);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
        configCA.withSubject(subject);

        // Define a CRL distribution point extension configuration
        CrlDistributionPointExtensionConfiguration CDPConfigure = new
CrlDistributionPointExtensionConfiguration();
        CDPConfigure.withOmitExtension(true);

        // Define a certificate revocation list configuration.
        CrlConfiguration crlConfigure = new CrlConfiguration();
        crlConfigure.withEnabled(true);
        crlConfigure.withExpirationInDays(365);
        crlConfigure.withCustomCname(null);
        crlConfigure.withS3BucketName("your-bucket-name");
        crlConfigure.withS3ObjectAcl("BUCKET_OWNER_FULL_CONTROL");
        crlConfigure.withCrlDistributionPointConfiguration(CDPConfigure);

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.SUBORDINATE;

        // ** Execute core code samples for Subordinate CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCertificate = GetCertificateAuthorityCertificate(paaArn, client);
```

```
        String subordinateCAArn = CreateCertificateAuthority(configCA, crlConfigure,
CAtype, client);
        String csr = GetCertificateAuthorityCsr(subordinateCAArn, client);
        String subordinateCertificateArn = IssueCertificate(paaArn, csr, client);
        String subordinateCertificate = GetCertificate(subordinateCertificateArn,
paaArn, client);
        ImportCertificateAuthorityCertificate(subordinateCertificate, rootCertificate,
subordinateCAArn, client);

    }

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

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

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

        return client;
    }

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

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

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

        // Retrieve and display the certificate information.
        String rootCertificate = getCACertificateResult.getCertificate();
        System.out.println("Product Attestation Authority (PAA) Certificate /
Certificate Chain:");
        System.out.println(rootCertificate);

        return rootCertificate;
    }

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

        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);
        createCARequest.withRevocationConfiguration(revokeConfig);

        // Create the private CA.
        CreateCertificateAuthorityResult createCAResult = null;
```

```
        try {
            createCAResult = client.createCertificateAuthority(createCARequest);
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        } catch (LimitExceededException ex) {
            throw ex;
        }

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

        return subordinateCAArn;
    }

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

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

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

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
```

```
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

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

        return csr;
    }

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

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

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

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/
BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1");

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

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(730L); // Approximately two years
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

        // Set the idempotency token.
```

```
        issueRequest.setIdempotencyToken("1234");

        ApiPassthrough apiPassthrough = new ApiPassthrough();

        // Generate Base64 encoded extension value for ExtendedKeyUsage
        String base64EncodedKUValue = generateKeyUsageValue();

        // Generate custom extension
        CustomExtension customKeyUsageExtension = new CustomExtension();
        customKeyUsageExtension.setObjectIdentifier("2.5.29.15");
        customKeyUsageExtension.setValue(base64EncodedKUValue);
        customKeyUsageExtension.setCritical(true);

        // Set KeyUsage extension to api passthrough
        Extensions extensions = new Extensions();
        extensions.setCustomExtensions(Arrays.asList(customKeyUsageExtension));
        apiPassthrough.setExtensions(extensions);
        issueRequest.setApiPassthrough(apiPassthrough);

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

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

        return subordinateCertificateArn;
    }
```

```
   @SneakyThrows
   private static String generateKeyUsageValue() {
      KeyUsage keyUsage = new KeyUsage(X509KeyUsage.keyCertSign |
X509KeyUsage.cRLSign);
      byte[] kuBytes = keyUsage.getEncoded();
      return Base64.getEncoder().encodeToString(kuBytes);
   }

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

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

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

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

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

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

```
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

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

        return subordinateCertificate;
    }

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

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

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

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

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
```

```
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
        System.out.println("Product Attestation Intermediate (PAI) certificate
 successfully imported.");
        System.out.println("Product Attestation Intermediate (PAI) activated
 successfully.");
    }

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

Membuat Sertifikat Pengesahan Perangkat (DAC)

Contoh Java ini menunjukkan cara menggunakan template BlankEndEntityCertificate_ CriticalBasicConstraints _apiPassthrough/v1 untuk membuat Sertifikat Pengesahan Perangkat Matter. Anda harus menghasilkan KeyUsage nilai yang dikodekan Base64 dan meneruskannya melalui a. CustomExtension

Contoh ini memanggil tindakan AWS Private CA API berikut:

- IssueCertificate

Jika Anda mengalami masalah, lihat Menggunakan standar Matter di bagian Pemecahan Masalah.

```
package com.amazonaws.samples.matter;

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

```
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

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

import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

import lombok.SneakyThrows;

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

   @SneakyThrows
```

```
   private static String generateKeyUsageValue() {
      KeyUsage keyUsage = new KeyUsage(X509KeyUsage.digitalSignature);
      byte[] kuBytes = keyUsage.getEncoded();
      return Base64.getEncoder().encodeToString(kuBytes);
   }

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

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

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

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

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

      // Set the CA ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:region:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012");

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

```
      ByteBuffer csrByteBuffer = stringToByteBuffer(strCSR);
      req.setCsr(csrByteBuffer);

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

      // Set the signing algorithm.
      req.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

      // Set the validity period for the certificate to be issued.
      Validity validity = new Validity();
      validity.withValue(10L);
      validity.withType("DAYS");
      req.withValidity(validity);

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

      // Define custom attributes
      List<CustomAttribute> customAttributes = Arrays.asList(
         new CustomAttribute()
            .withObjectIdentifier("2.5.4.3")
            .withValue("Matter Test DAC 0001"),
         new CustomAttribute()
            .withObjectIdentifier("1.3.6.1.4.1.37244.2.1")
            .withValue("FFF1"),
         new CustomAttribute()
            .withObjectIdentifier("1.3.6.1.4.1.37244.2.2")
            .withValue("8000")
      );

      // Define a cert subject.
      ASN1Subject subject = new ASN1Subject();
      subject.setCustomAttributes(customAttributes);

      ApiPassthrough apiPassthrough = new ApiPassthrough();
      apiPassthrough.setSubject(subject);

      // Generate Base64 encoded extension value for ExtendedKeyUsage
      String base64EncodedKUValue = generateKeyUsageValue();

      // Generate custom extension
      CustomExtension customKeyUsageExtension = new CustomExtension();
```

```
        customKeyUsageExtension.setObjectIdentifier("2.5.29.15"); // KeyUsage Extension
 OID
        customKeyUsageExtension.setValue(base64EncodedKUValue);
        customKeyUsageExtension.setCritical(true);

        Extensions extensions = new Extensions();
        extensions.setCustomExtensions(Arrays.asList(customKeyUsageExtension));
        apiPassthrough.setExtensions(extensions);
        req.setApiPassthrough(apiPassthrough);

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

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

Aktifkan Root CA untuk Node Operational Certificates (NOC).

Contoh Java ini menunjukkan cara menggunakan Definisi RootCACertificate_APIPassthrough/V1 template untuk membuat dan menginstal sertifikat Matter Root CA untuk mengeluarkan NOC. Ekstensi AuthorityKeyIdentifier (AKI) adalah opsional untuk sertifikat NOC Root CA. Untuk menyetel AKI, Anda harus menghasilkan nilai AKI yang dienkode Base64 dan meneruskannya melalui a. CustomExtension

Contoh ini memanggil tindakan AWS Private CA API berikut:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

Jika Anda mengalami masalah, lihat Menggunakan standar Matter di bagian Pemecahan Masalah.

```
package com.amazonaws.samples.matter;

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


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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import org.bouncycastle.asn1.x509.SubjectPublicKeyInfo;
import org.bouncycastle.cert.jcajce.JcaX509ExtensionUtils;
import org.bouncycastle.openssl.PEMParser;
import org.bouncycastle.pkcs.PKCS10CertificationRequest;
import org.bouncycastle.util.io.pem.PemReader;
```

```
import lombok.SneakyThrows;

public class RootCAActivation {
    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = "region";  // Substitute your region here, e.g. "ap-
southeast-2"

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1.37244.1.4")
                .withValue("CACACACA00000001")
        );

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
 CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.EC_prime256v1);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
        configCA.withSubject(subject);

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.ROOT;

        // ** Execute core code samples for Root CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCAArn = CreateCertificateAuthority(configCA, CAtype, client);
        String csr = GetCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = IssueCertificate(rootCAArn, csr, client);
        String rootCertificate = GetCertificate(rootCertificateArn, rootCAArn, client);
        ImportCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

    private static AWSACMPCA ClientBuilder(String endpointRegion) {
        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
```

```
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load the credentials from the credential profiles file. " +
                    "Please make sure that your credentials file is at the correct " +
                    "location (C:\\Users\\joneps\\.aws\\credentials), and is in valid
format.",
                    e);
        }

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

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

        return client;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CertificateAuthorityType CAtype, AWSACMPCA client) {
        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);

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

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

        return rootCAArn;
    }

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

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

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

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
```

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

        return csr;
    }

    @SneakyThrows
    private static String generateAuthorityKeyIdentifier(final String csrPEM) {
        PKCS10CertificationRequest csr = getPKCS10CertificationRequest(csrPEM);
        SubjectPublicKeyInfo spki = csr.getSubjectPublicKeyInfo();

        JcaX509ExtensionUtils extensionUtils = new JcaX509ExtensionUtils();
        byte[] akiBytes =
 extensionUtils.createAuthorityKeyIdentifier(spki).getEncoded();

        return Base64.getEncoder().encodeToString(akiBytes);
    }

    @SneakyThrows
    private static PKCS10CertificationRequest getPKCS10CertificationRequest(final
 String csrPEM) {
        ByteArrayInputStream bais = new ByteArrayInputStream(csrPEM.getBytes());
        PemReader pemReader = new PemReader(new InputStreamReader(bais));
        PEMParser parser = new PEMParser(pemReader);
        Object o = parser.readObject();
        if (o instanceof PKCS10CertificationRequest) {
            return (PKCS10CertificationRequest) o;
        }
        return null;
    }

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

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

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

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

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

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

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

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

        // Generate Base64 encoded extension value for AuthorityKeyIdentifier
        String base64EncodedExtValue = generateAuthorityKeyIdentifier(csr);

        // Generate custom extension
        CustomExtension customExtension = new CustomExtension();
        customExtension.setObjectIdentifier("2.5.29.35"); // AuthorityKeyIdentifier
Extension OID
        customExtension.setValue(base64EncodedExtValue);

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

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

```
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

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

        return rootCertificateArn;
    }

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

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

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

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

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

        // Retrieve the certificate and certificate chain.
        GetCertificateResult certificateResult = null;
        try {
            certificateResult = client.getCertificate(certificateRequest);
        } catch (RequestInProgressException ex) {
```

```
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

        // Get the certificate and certificate chain and display the result.
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
    }

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

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

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

        importRequest.setCertificateChain(null);

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
```

```
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Root CA certificate successfully imported.");
        System.out.println("Root CA activated successfully.");
    }

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

Aktifkan CA Bawahan untuk Sertifikat Operasional Node (NOC)

Contoh Java ini menunjukkan cara menggunakan BlankSubordinatePathLenCacertificate_0_apiPassthrough/definisi v1 template untuk menerbitkan dan menginstal sertifikat Matter Subordinate CA untuk mengeluarkan NOC. Anda harus menghasilkan KeyUsage nilai yang dikodekan Base64 dan meneruskannya melalui a. CustomExtension

Contoh ini memanggil tindakan AWS Private CA API berikut:

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

Jika masalah terjadi, lihat Menggunakan standar Matter di bagian Pemecahan Masalah.

```
package com.amazonaws.samples.matter;

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


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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCertificateResult;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
```

```
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import lombok.SneakyThrows;

public class IntermediateCAActivation {

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

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

        // Define custom attributes
        List<CustomAttribute> customAttributes = Arrays.asList(
            new CustomAttribute()
                .withObjectIdentifier("1.3.6.1.4.1.37244.1.3")
                .withValue("CACACACA00000003")
        );

        // Define a CA subject.
```

```
        ASN1Subject subject = new ASN1Subject();
        subject.setCustomAttributes(customAttributes);

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
CertificateAuthorityConfiguration();
        configCA.withKeyAlgorithm(KeyAlgorithm.EC_prime256v1);
        configCA.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
        configCA.withSubject(subject);

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.SUBORDINATE;

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

    }

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

        String endpointProtocol = "https://acm-pca." + endpointRegion +
".amazonaws.com/";
        EndpointConfiguration endpoint =
```

```
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
endpointRegion);

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

        return client;
    }

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

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

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

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

        return rootCertificate;
    }

    private static String CreateCertificateAuthority(CertificateAuthorityConfiguration
configCA, CertificateAuthorityType CAtype, AWSACMPCA client) {
```

```
        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
        createCARequest.withIdempotencyToken("123987");
        createCARequest.withCertificateAuthorityType(CAtype);

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

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

        return subordinateCAArn;
    }

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

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

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

```
        } catch(AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Get the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

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

        return csr;
    }

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

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

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

        // Set the template ARN.
        issueRequest.withTemplateArn("arn:aws:acm-pca:::template/
BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1");

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

        // Set the signing algorithm.
        issueRequest.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);
```

```
        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(730L); // Approximately two years
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

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

        ApiPassthrough apiPassthrough = new ApiPassthrough();

        // Generate base64 encoded extension value for ExtendedKeyUsage
        String base64EncodedKUValue = generateKeyUsageValue();

        // Generate custom extension
        CustomExtension customKeyUsageExtension = new CustomExtension();
        customKeyUsageExtension.setObjectIdentifier("2.5.29.15");
        customKeyUsageExtension.setValue(base64EncodedKUValue);
        customKeyUsageExtension.setCritical(true);

        // Set KeyUsage extension to api passthrough
        Extensions extensions = new Extensions();
        extensions.setCustomExtensions(Arrays.asList(customKeyUsageExtension));
        apiPassthrough.setExtensions(extensions);
        issueRequest.setApiPassthrough(apiPassthrough);

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

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

        return subordinateCertificateArn;
    }

    @SneakyThrows
    private static String generateKeyUsageValue() {
       KeyUsage keyUsage = new KeyUsage(X509KeyUsage.keyCertSign |
X509KeyUsage.cRLSign);
       byte[] kuBytes = keyUsage.getEncoded();
       return Base64.getEncoder().encodeToString(kuBytes);
    }

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

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

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

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

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

        // Get the certificate and certificate chain.
```

```
        GetCertificateResult certificateResult = null;
        try {
            certificateResult = client.getCertificate(certificateRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        }

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

        return subordinateCertificate;
    }

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

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

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

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

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
```

```
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }
        System.out.println("Subordinate CA certificate successfully imported.");
        System.out.println("Subordinate CA activated successfully.");
    }

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

Membuat Node Operational Certificate (NOC)

Contoh Java ini menunjukkan cara menggunakan template BlankEndEntityCertificate_ CriticalBasicConstraints _apiPassthrough/v1 untuk membuat Sertifikat Operasional Node Matter. Anda harus menghasilkan KeyUsage nilai yang dikodekan Base64 dan meneruskannya melalui a. CustomExtension

Contoh ini memanggil tindakan AWS Private CA API berikut:

- IssueCertificate

Jika Anda mengalami masalah, lihat Menggunakan standar Matter di bagian Pemecahan Masalah.

```
package com.amazonaws.samples.matter;
```

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

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;
import java.util.Objects;

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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CustomAttribute;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

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

import org.bouncycastle.asn1.x509.ExtendedKeyUsage;
import org.bouncycastle.asn1.x509.KeyPurposeId;
import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

import lombok.SneakyThrows;

public class IssueNodeOperatingCertificate {
   public static ByteBuffer stringToByteBuffer(final String string) {
```

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

   @SneakyThrows
   private static String generateExtendedKeyUsageValue() {
      KeyPurposeId[] keyPurposeIds = new KeyPurposeId[]
 { KeyPurposeId.id_kp_clientAuth, KeyPurposeId.id_kp_serverAuth };
      ExtendedKeyUsage eku = new ExtendedKeyUsage(keyPurposeIds);
      byte[] ekuBytes = eku.getEncoded();
      return Base64.getEncoder().encodeToString(ekuBytes);
   }

   @SneakyThrows
   private static String generateKeyUsageValue() {
      KeyUsage keyUsage = new KeyUsage(X509KeyUsage.digitalSignature);
      byte[] kuBytes = keyUsage.getEncoded();
      return Base64.getEncoder().encodeToString(kuBytes);
   }

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

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

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

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
```

```
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

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

      // Set the CA ARN.
      req.withCertificateAuthorityArn("arn:aws:acm-pca:region:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012");

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

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

      // Set the signing algorithm.
      req.withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA);

      // Set the validity period for the certificate to be issued.
      Validity validity = new Validity();
      validity.withValue(10L);
      validity.withType("DAYS");
      req.withValidity(validity);

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

      // Define custom attributes
      List<CustomAttribute> customAttributes = Arrays.asList(
         new CustomAttribute()
            .withObjectIdentifier("1.3.6.1.4.1.37244.1.1")
            .withValue("DEDEDEDE00010001"),
         new CustomAttribute()
            .withObjectIdentifier("1.3.6.1.4.1.37244.1.5")
            .withValue("FAB000000000001D")
```

```
      );

      // Define a cert subject.
      ASN1Subject subject = new ASN1Subject();
      subject.setCustomAttributes(customAttributes);

      ApiPassthrough apiPassthrough = new ApiPassthrough();
      apiPassthrough.setSubject(subject);

      // Generate Base64 encoded extension value for ExtendedKeyUsage
      String base64EncodedKUValue = generateKeyUsageValue();

      // Generate custom extension
      CustomExtension customKeyUsageExtension = new CustomExtension();
      customKeyUsageExtension.setObjectIdentifier("2.5.29.15");
      customKeyUsageExtension.setValue(base64EncodedKUValue);
      customKeyUsageExtension.setCritical(true);

      // Generate Base64 encoded extension value for ExtendedKeyUsage
      String base64EncodedEKUValue = generateExtendedKeyUsageValue();

      CustomExtension customExtendedKeyUsageExtension = new CustomExtension();
      customExtendedKeyUsageExtension.setObjectIdentifier("2.5.29.37"); //
ExtendedKeyUsage Extension OID
      customExtendedKeyUsageExtension.setValue(base64EncodedEKUValue);
      customExtendedKeyUsageExtension.setCritical(true);

      // Set KeyUsage and ExtendedKeyUsage extension to api-passthrough
      Extensions extensions = new Extensions();
      extensions.setCustomExtensions(Arrays.asList(customKeyUsageExtension,
customExtendedKeyUsageExtension));
      apiPassthrough.setExtensions(extensions);
      req.setApiPassthrough(apiPassthrough);

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

```
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidArgsException ex) {
         throw ex;
      } catch (MalformedCSRException ex) {
         throw ex;
      }

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

Menggunakan AWS Private CA API untuk mengimplementasikan standar lisensi mengemudi seluler (mDL) (contoh Java)

Anda dapat menggunakan AWS Private Certificate Authority API untuk membuat sertifikat yang sesuai dengan standar ISO/IEC untuk SIM seluler (mDL). Standar ini menetapkan spesifikasi antarmuka untuk penerapan SIM yang terkait dengan perangkat seluler, termasuk konfigurasi sertifikat.

Contoh Java di bagian ini berinteraksi dengan layanan dengan mengirimkan permintaan HTTP. Layanan tersebut mengembalikan tanggapan HTTP. Untuk informasi lebih lanjut, lihat Referensi API AWS Private Certificate Authority.

Selain HTTP API, Anda juga dapat menggunakan AWS SDK dan AWS CLI alat untuk mengelolaAWS Private CA. Sebaiknya gunakan SDK atau AWS CLI melalui HTTP API. Untuk informasi lebih lanjut, lihat Alat untuk Amazon Web Services. Topik berikut menunjukkan cara menggunakan AWS SDK for Javauntuk memprogram AWS Private CA API.

Itu `GetCertificateAuthorityCsr`, `GetCertificate`, dan `DescribeCertificateAuthorityAuditReport`operasi mendukung pelayan. Anda dapat menggunakan penunggu untuk mengontrol perkembangan kode Anda berdasarkan kehadiran atau keadaan sumber daya tertentu. Untuk informasi selengkapnya, lihat topik berikut, dan Pelayan di AWS SDK for Java di Blog Pengembang. AWS

Topik

- Aktifkan sertifikat otoritas otoritas penerbitan (IACA)
- Buat sertifikat penandatangan dokumen

Aktifkan sertifikat otoritas otoritas penerbitan (IACA)

Contoh Java ini menunjukkan cara menggunakan BlankRootPathLenCacertificate_ 0_apiPassthrough/definisi v1 template untuk membuat dan menginstal sertifikat ISO/IEC mDL standard -compliant issuing authority certificate authority (IACA). Anda harus menghasilkan nilai yang dikodekan base64 untuk`KeyUsage`,, dan `IssuerAlternativeNameCRLDistributionPoint`, dan meneruskannya. `CustomExtensions`

> **Note**
>
> Sertifikat tautan IACA menetapkan jalur kepercayaan dari sertifikat root IACA lama ke sertifikat root IACA yang baru. Otoritas penerbit dapat menghasilkan dan mendistribusikan sertifikat tautan IACA selama proses re-key IACA. Anda tidak dapat mengeluarkan sertifikat tautan IACA dengan menggunakan sertifikat root IACA dengan `pathLen=0` set.

Contoh ini memanggil tindakan AWS Private CA API berikut:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

```
package com.amazonaws.samples.mdl;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
```

```
import java.util.Base64;
import java.util.Objects;

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;

import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

import org.bouncycastle.asn1.x509.GeneralNames;
import org.bouncycastle.asn1.x509.GeneralName;
import org.bouncycastle.asn1.x509.CRLDistPoint;
import org.bouncycastle.asn1.x509.DistributionPoint;
import org.bouncycastle.asn1.x509.DistributionPointName;
import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

import lombok.SneakyThrows;

public class IssuingAuthorityCertificateAuthorityActivation {
```

```
    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = null;  // Substitute your region here, e.g. "ap-
southeast-2"
        if (endpointRegion == null) throw new Exception("Region cannot be null");

        // Define a CA subject.
        ASN1Subject subject = new ASN1Subject()
                .withCountry("US") // mDL spec requires ISO 3166-1-alpha-2 country code
 e.g. "US"
                .withCommonName("mDL Test IACA");

        // Define the CA configuration.
        CertificateAuthorityConfiguration configCA = new
 CertificateAuthorityConfiguration()
                .withKeyAlgorithm(KeyAlgorithm.EC_prime256v1)
                .withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA)
                .withSubject(subject);

        // Define a certificate authority type
        CertificateAuthorityType CAType = CertificateAuthorityType.ROOT;

        // Execute core code samples for Root CA activation in sequence
        AWSACMPCA client = buildClient(endpointRegion);
        String rootCAArn = createCertificateAuthority(configCA, CAType, client);
        String csr = getCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = issueCertificate(rootCAArn, csr, client);
        String rootCertificate = getCertificate(rootCertificateArn, rootCAArn, client);
        importCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

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

        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
```

```
                .withRegion(endpointRegion)
                .withCredentials(new AWSStaticCredentialsProvider(credentials))
                .build();

        return client;
    }

    private static String createCertificateAuthority(CertificateAuthorityConfiguration
configCA, CertificateAuthorityType CAtype, AWSACMPCA client) {
        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest()
                .withCertificateAuthorityConfiguration(configCA)
                .withIdempotencyToken("123987")
                .withCertificateAuthorityType(CAtype);

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

        // Get the ARN of the private CA.
        String rootCAArn = createCAResult.getCertificateAuthorityArn();
        System.out.println("Issuing Authority Certificate Authority (IACA) Arn: " +
rootCAArn);

        return rootCAArn;
    }

    private static String getCertificateAuthorityCsr(String rootCAArn, AWSACMPCA
client) {

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

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

        // Get the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

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

        return csr;
    }

    @SneakyThrows
    private static String issueCertificate(String rootCAArn, String csr, AWSACMPCA
 client) {
        IssueCertificateRequest issueRequest = new IssueCertificateRequest()
                .withCertificateAuthorityArn(rootCAArn)
                .withTemplateArn("arn:aws:acm-pca:::template/
BlankRootCACertificate_PathLen0_APIPassthrough/V1")
                .withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA)
```

```
                .withIdempotencyToken("1234");

        // Set the CSR.
        ByteBuffer csrByteBuffer = stringToByteBuffer(csr);
        issueRequest.setCsr(csrByteBuffer);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity()
                .withValue(3650L)
                .withType("DAYS");
        issueRequest.setValidity(validity);

        // Generate base64 encoded extension value for KeyUsage
        KeyUsage keyUsage = new KeyUsage(X509KeyUsage.keyCertSign +
 X509KeyUsage.cRLSign);
        byte[] kuBytes = keyUsage.getEncoded();
        String base64EncodedKUValue = Base64.getEncoder().encodeToString(kuBytes);

        CustomExtension keyUsageCustomExtension = new CustomExtension()
                .withObjectIdentifier("2.5.29.15") // KeyUsage Extension OID
                .withValue(base64EncodedKUValue)
                .withCritical(true);

        // Generate base64 encoded extension value for IssuerAlternativeName
        GeneralNames issuerAlternativeName = new GeneralNames(new
 GeneralName(GeneralName.uniformResourceIdentifier, "https://issuer-alternative-
name.com"));
        String base64EncodedIANValue =
 Base64.getEncoder().encodeToString(issuerAlternativeName.getEncoded());

        CustomExtension ianCustomExtension = new CustomExtension()
                .withValue(base64EncodedIANValue)
                .withObjectIdentifier("2.5.29.18"); // IssuerAlternativeName Extension
 OID

        // Generate base64 encoded extension value for CRLDistributionPoint
        CRLDistPoint crlDistPoint = new CRLDistPoint(new DistributionPoint[]{new
 DistributionPoint(new DistributionPointName(
                new GeneralNames(new GeneralName(GeneralName.uniformResourceIdentifier,
 "dummycrl.crl"))), null, null)});
        String base64EncodedCDPValue =
 Base64.getEncoder().encodeToString(crlDistPoint.getEncoded());

        CustomExtension cdpCustomExtension = new CustomExtension()
```

```
                .withValue(base64EncodedCDPValue)
                .withObjectIdentifier("2.5.29.31"); // CRLDistributionPoint Extension
OID

        // Add custom extension to api-passthrough
        Extensions extensions = new Extensions()
                .withCustomExtensions(Arrays.asList(keyUsageCustomExtension,
ianCustomExtension, cdpCustomExtension));
        ApiPassthrough apiPassthrough = new ApiPassthrough()
                .withExtensions(extensions);
        issueRequest.setApiPassthrough(apiPassthrough);

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

        // Get and display the certificate ARN.
        String rootCertificateArn = issueResult.getCertificateArn();
        System.out.println("mDL IACA Certificate Arn: " + rootCertificateArn);

        return rootCertificateArn;
    }

    private static String getCertificate(String rootCertificateArn, String rootCAArn,
AWSACMPCA client) {

        // Create a request object.
        GetCertificateRequest certificateRequest = new GetCertificateRequest()
                .withCertificateArn(rootCertificateArn)
                .withCertificateAuthorityArn(rootCAArn);
```

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

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

        // Get the certificate and certificate chain and display the result.
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
    }

    private static void importCertificateAuthorityCertificate(String rootCertificate,
String rootCAArn, AWSACMPCA client) {

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

```
                        .withCertificateChain(null)
                        .withCertificateAuthorityArn(rootCAArn);

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Root CA certificate successfully imported.");
        System.out.println("Root CA activated successfully.");
    }

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

Buat sertifikat penandatangan dokumen

Contoh Java ini menunjukkan cara menggunakan template [BlankEndEntityCertificate_apiPassthrough/v1 untuk membuat sertifikat penandatangan]

dokumen standar ISO/IEC mDL. Anda harus menghasilkan nilai yang dikodekan base64 untuk`KeyUsage`,`IssuerAlternativeName`, `CRLDistributionPoint` dan meneruskannya. `CustomExtensions`

Contoh ini memanggil tindakan AWS Private CA API berikut:

- IssueCertificate

```
package com.amazonaws.samples.mdl;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.Base64;
import java.util.Objects;

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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.ApiPassthrough;
import com.amazonaws.services.acmpca.model.ExtendedKeyUsage;
import com.amazonaws.services.acmpca.model.CustomExtension;
import com.amazonaws.services.acmpca.model.Extensions;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

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

import org.bouncycastle.asn1.x509.GeneralNames;
```

```
import org.bouncycastle.asn1.x509.GeneralName;
import org.bouncycastle.asn1.x509.CRLDistPoint;
import org.bouncycastle.asn1.x509.DistributionPoint;
import org.bouncycastle.asn1.x509.DistributionPointName;
import org.bouncycastle.asn1.x509.KeyUsage;
import org.bouncycastle.jce.X509KeyUsage;

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

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

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

        // Create a client that you can use to make requests.
        String endpointRegion = null;  // Substitute your region here, e.g. "ap-
southeast-2"
        if (endpointRegion == null) throw new Exception("Region cannot be null");

        AWSACMPCA client = AWSACMPCAClientBuilder.standard()
                .withRegion(endpointRegion)
                .withCredentials(new AWSStaticCredentialsProvider(credentials))
                .build();

        // Create a certificate request:
        String caArn = null;
        if (caArn == null) throw new Exception("Certificate authority ARN cannot be
 null");

        IssueCertificateRequest req = new IssueCertificateRequest()
```

```
                .withCertificateAuthorityArn(caArn)
                .withTemplateArn("arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APIPassthrough/V1")
                .withSigningAlgorithm(SigningAlgorithm.SHA256WITHECDSA)
                .withIdempotencyToken("1234");

        // Specify the certificate signing request (CSR) for the certificate to be
 signed and issued.
        // Format: "-----BEGIN CERTIFICATE REQUEST-----\n" +
        //         "base64-encoded certificate\n" +
        //         "-----END CERTIFICATE REQUEST-----\n";
        String strCSR = null;
        if (strCSR == null) throw new Exception("CSR string cannot be null");

        ByteBuffer csrByteBuffer = stringToByteBuffer(strCSR);
        req.setCsr(csrByteBuffer);

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity()
                .withValue(365L)
                .withType("DAYS");
        req.setValidity(validity);

        // Define a cert subject.
        ASN1Subject subject = new ASN1Subject()
                .withCountry("US") // mDL spec requires ISO 3166-1-alpha-2 country code
 e.g. "US"
                .withCommonName("mDL Test DS");

        ApiPassthrough apiPassthrough = new ApiPassthrough()
                .withSubject(subject);

        // Generate base64 encoded extension value for KeyUsage
        KeyUsage keyUsage = new KeyUsage(X509KeyUsage.digitalSignature);
        byte[] kuBytes = keyUsage.getEncoded();
        String base64EncodedKUValue = Base64.getEncoder().encodeToString(kuBytes);

        CustomExtension customKeyUsageExtension = new CustomExtension()
                .withObjectIdentifier("2.5.29.15") // KeyUsage Extension OID
                .withValue(base64EncodedKUValue)
                .withCritical(true);

        // Generate base64 encoded extension value for IssuerAlternativeName
```

```
        GeneralNames issuerAlternativeName = new GeneralNames(new
 GeneralName(GeneralName.uniformResourceIdentifier, "https://issuer-alternative-
name.com"));
        String base64EncodedIANValue =
 Base64.getEncoder().encodeToString(issuerAlternativeName.getEncoded());

        CustomExtension ianCustomExtension = new CustomExtension()
                .withValue(base64EncodedIANValue)
                .withObjectIdentifier("2.5.29.18"); // IssuerAlternativeName Extension
 OID

        // Generate base64 encoded extension value for CRLDistributionPoint
        CRLDistPoint crlDistPoint = new CRLDistPoint(new DistributionPoint[]{new
 DistributionPoint(new DistributionPointName(
                new GeneralNames(new GeneralName(GeneralName.uniformResourceIdentifier,
 "dummycrl.crl"))), null, null)});
        String base64EncodedCDPValue =
 Base64.getEncoder().encodeToString(crlDistPoint.getEncoded());

        CustomExtension cdpCustomExtension = new CustomExtension()
                .withValue(base64EncodedCDPValue)
                .withObjectIdentifier("2.5.29.31"); // CRLDistributionPoint Extension
 OID

        // Generate EKU
        ExtendedKeyUsage eku = new ExtendedKeyUsage()
                .withExtendedKeyUsageObjectIdentifier("1.0.18013.5.1.2"); // EKU value
 reserved for mDL DS

        // Set KeyUsage, ExtendedKeyUsage, IssuerAlternativeName, CRL Distribution
 Point extensions to api-passthrough
        Extensions extensions = new Extensions()
                .withCustomExtensions(Arrays.asList(customKeyUsageExtension,
 ianCustomExtension, cdpCustomExtension))
                .withExtendedKeyUsage(Arrays.asList(eku));
        apiPassthrough.setExtensions(extensions);
        req.setApiPassthrough(apiPassthrough);

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

```
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (MalformedCSRException ex) {
            throw ex;
        }

        // Get and display the certificate ARN.
        String arn = result.getCertificateArn();
        System.out.println("mDL DS Certificate Arn: " + arn);
    }
}
```

Sertifikat CA pribadi yang ditandatangani secara eksternal

Jika akar kepercayaan hierarki CA pribadi Anda harus berupa CA di luarAWS Private CA, Anda dapat membuat dan menandatangani sendiri CA root Anda sendiri. Atau, Anda dapat memperoleh sertifikat CA pribadi yang ditandatangani oleh CA privat eksternal yang dioperasikan oleh organisasi Anda. Apa pun sumbernya, Anda dapat menggunakan CA yang diperoleh secara eksternal ini untuk menandatangani sertifikat CA bawahan pribadi yang mengelola. AWS Private CA

> **Note**
>
> Prosedur untuk membuat atau memperoleh penyedia layanan kepercayaan eksternal berada di luar cakupan panduan ini.

Menggunakan CA induk eksternal dengan AWS Private CA memungkinkan Anda untuk menerapkan batasan nama CA seperti yang didefinisikan di bagian Kendala Nama RFC 5280. Batasan nama menyediakan cara bagi administrator CA untuk membatasi nama subjek dalam sertifikat.

Jika Anda berencana untuk menandatangani sertifikat CA bawahan pribadi dengan CA eksternal, ada tiga tugas yang harus diselesaikan sebelum Anda memiliki CA yang berfungsi: AWS Private CA

1. Buat permintaan penandatanganan sertifikat (CSR).
2. Kirimkan CSR ke otoritas penandatanganan eksternal Anda dan kembalikan dengan sertifikat dan rantai sertifikat yang ditandatangani.
3. Instal sertifikat yang ditandatangani diAWS Private CA.

Prosedur berikut menjelaskan cara menyelesaikan tugas-tugas ini menggunakan salah satu AWS Management Console atauAWS CLI.

Untuk mendapatkan dan menginstal sertifikat CA (konsol) yang ditandatangani secara eksternal

1. (Opsional) Jika Anda belum berada di halaman detail CA, buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/home. Pada halaman Otoritas sertifikat pribadi, pilih CA bawahan dengan status Sertifikat tertunda, Aktif, Dinonaktifkan, atau Kedaluwarsa.
2. Pilih Tindakan, Instal Sertifikat CA untuk membuka halaman Instal sertifikat CA bawahan.
3. Pada halaman Instal sertifikat CA bawahan, di bawah Pilih jenis CA, pilih CA pribadi eksternal.

4. Di bawah CSR untuk CA ini, konsol menampilkan teks ASCII yang dikodekan Base64 dari CSR. Anda dapat menyalin teks menggunakan tombol Salin atau Anda dapat memilih Ekspor CSR ke file dan menyimpannya secara lokal.

   > **Note**
   >
   > Format yang tepat dari teks CSR harus dipertahankan saat mengatasi dan menempel.

5. Jika Anda tidak dapat segera melakukan langkah-langkah offline untuk mendapatkan sertifikat yang ditandatangani dari otoritas penandatanganan eksternal Anda, Anda dapat menutup halaman dan kembali ke sana setelah Anda memiliki sertifikat yang ditandatangani dan rantai sertifikat.

   Jika tidak, jika Anda siap, lakukan salah satu hal berikut:

   - Tempelkan teks ASCII yang dikodekan Base64 dari badan sertifikat Anda dan rantai sertifikat Anda ke dalam kotak teks masing-masing.
   - Pilih Unggah untuk memuat badan sertifikat dan rantai sertifikat dari file lokal ke dalam kotak teks masing-masing.

6. Pilih Konfirmasi dan instal.

Untuk mendapatkan dan menginstal sertifikat CA (CLI) yang ditandatangani secara eksternal

1. Gunakan get-certificate-authority-csrperintah untuk mengambil permintaan penandatanganan sertifikat (CSR) untuk CA pribadi Anda. Jika Anda ingin mengirim CSR ke tampilan Anda, gunakan opsi `--output text` untuk menghilangkan karakter CR/LF dari akhir setiap baris. Untuk mengirim CSR ke file, gunakan opsi mengalihkan kembali (>) diikuti dengan nama file.

   ```
   $ 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
   ```

   Setelah menyimpan CSR sebagai file lokal, Anda dapat memeriksanya dengan menggunakan perintah OpenSSL berikut:

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

Perintah ini menghasilkan output yang terlihat seperti berikut ini. Perhatikan bahwa ekstensi CA adalah TRUE, yang menunjukkan bahwa CSR adalah untuk sertifikat CA.

```
Certificate Request:
Data:
Version: 0 (0x0)
Subject: O=ExampleCompany, OU=Corporate Office, CN=Example CA 1
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
        Modulus:
            00:d4:23:51:b3:dd:01:09:01:0b:4c:59:e4:ea:81:
            1d:7f:48:36:ef:2a:e9:45:82:ec:95:1d:c6:d7:c9:
            7f:19:06:73:c5:cd:63:43:14:eb:c8:03:82:f8:7b:
            c7:89:e6:8d:03:eb:b6:76:58:70:f2:cb:c3:4c:67:
            ea:50:fd:b9:17:84:b8:60:2c:64:9d:2e:d5:7d:da:
            46:56:38:34:a9:0d:57:77:85:f1:6f:b8:ce:73:eb:
            f7:62:a7:8e:e6:35:f5:df:0c:f7:3b:f5:7f:bd:f4:
            38:0b:95:50:2c:be:7d:bf:d9:ad:91:c3:81:29:23:
            b2:5e:a6:83:79:53:f3:06:12:20:7e:a8:fa:18:d6:
            a8:f3:a3:89:a5:a3:6a:76:da:d0:97:e5:13:bc:84:
            a6:5c:d6:54:1a:f0:80:16:dd:4e:79:7b:ff:6d:39:
            b5:67:56:cb:02:6b:14:c3:17:06:0e:7d:fb:d2:7e:
            1c:b8:7d:1d:83:13:59:b2:76:75:5e:d1:e3:23:6d:
            8a:5e:f5:85:ca:d7:e9:a3:f1:9b:42:9f:ed:8a:3c:
            14:4d:1f:fc:95:2b:51:6c:de:8f:ee:02:8c:0c:b6:
            3e:2d:68:e5:f8:86:3f:4f:52:ec:a6:f0:01:c4:7d:
            68:f3:09:ae:b9:97:d6:fc:e4:de:58:58:37:09:9a:
            f6:27
        Exponent: 65537 (0x10001)
Attributes:
Requested Extensions:
    X509v3 Basic Constraints:
        CA:TRUE
Signature Algorithm: sha256WithRSAEncryption
 c5:64:0e:6c:cf:11:03:0b:b7:b8:9e:48:e1:04:45:a0:7f:cc:
 a7:fd:e9:4d:c9:00:26:c5:6e:d0:7e:69:7a:fb:17:1f:f3:5d:
 ac:f3:65:0a:96:5a:47:3c:c1:ee:45:84:46:e3:e6:05:73:0c:
 ce:c9:a0:5e:af:55:bb:89:46:21:92:7b:10:96:92:1b:e6:75:
 de:02:13:2d:98:72:47:bd:b1:13:1a:3d:bb:71:ae:62:86:1a:
 ee:ae:4e:f4:29:2e:d6:fc:70:06:ac:ca:cf:bb:ee:63:68:14:
 8e:b2:8f:e3:8d:e8:8f:e0:33:74:d6:cf:e2:e9:41:ad:b6:47:
```

```
f8:2e:7d:0a:82:af:c6:d8:53:c2:88:a0:32:05:09:e0:04:8f:
79:1c:ac:0d:d4:77:8e:a6:b2:5f:07:f8:1b:e3:98:d4:12:3d:
28:32:82:b5:50:92:a4:b2:4c:28:fc:d2:73:75:75:ff:10:33:
2c:c0:67:4b:de:fd:e6:69:1c:a8:bb:e8:31:93:07:35:69:b7:
d6:53:37:53:d5:07:dd:54:35:74:50:50:f9:99:7d:38:b7:b6:
7f:bd:6c:b8:e4:2a:38:e5:04:00:a8:a3:d9:e5:06:38:e0:38:
4c:ca:a9:3c:37:6d:ba:58:38:11:9c:30:08:93:a5:62:00:18:
d1:83:66:40
```

2. Kirimkan CSR ke otoritas penandatanganan eksternal Anda dan dapatkan file yang berisi sertifikat dan rantai sertifikat berkode PEM Base64.
3. Gunakan import-certificate-authority-certificateperintah untuk mengimpor file sertifikat CA pribadi dan file rantai ke dalamAWS Private CA.

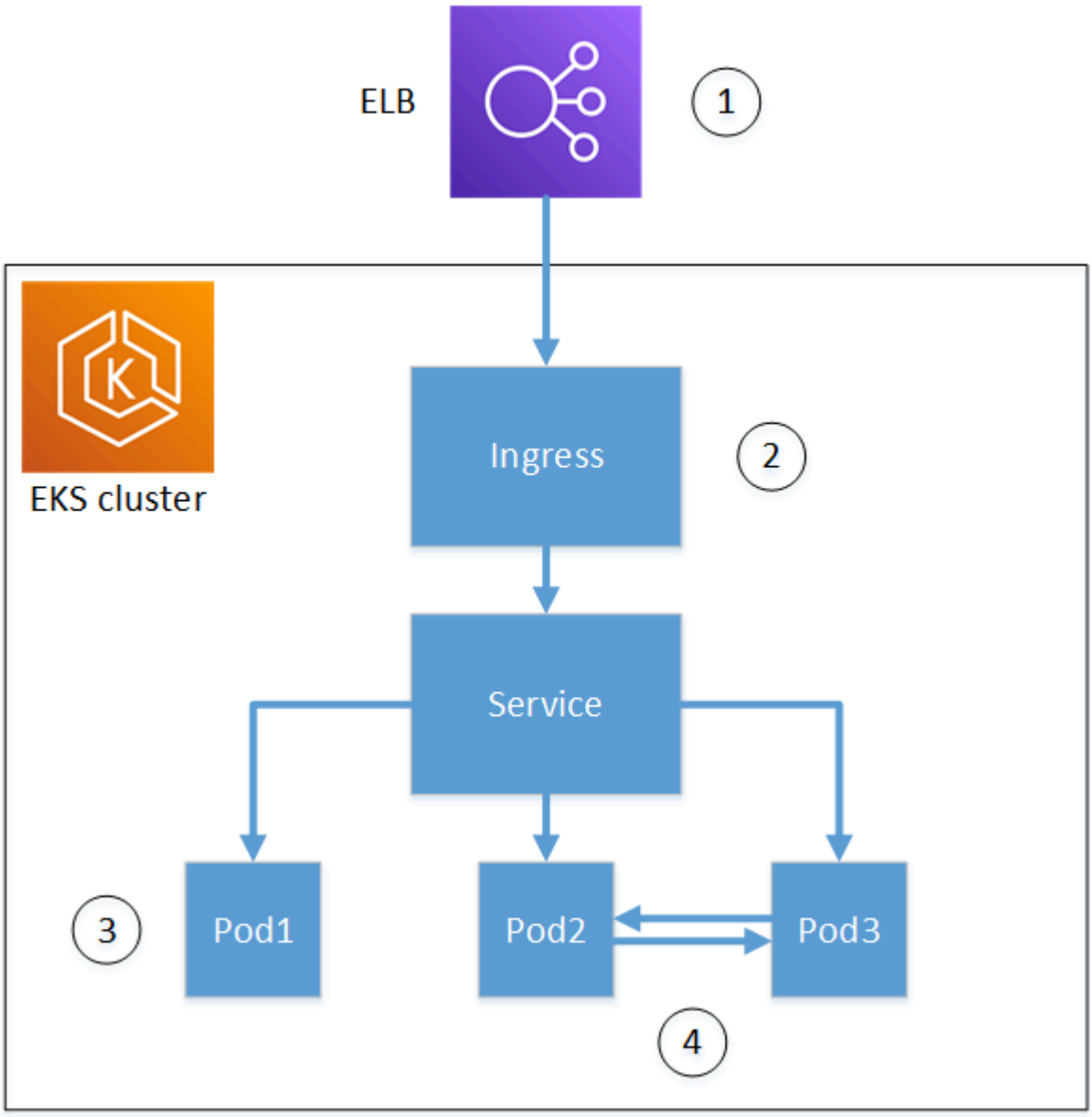
```
$ aws acm-pca import-certificate-authority-certificate \
--certificate-authority-arn arn:aws:acm-pca:region:account:\
certificate-authority/12345678-1234-1234-1234-123456789012 \
--certificate file://C:\example_ca_cert.pem \
--certificate-chain file://C:\example_ca_cert_chain.pem
```

Mengamankan Kubernetes dengan AWS Private CA

Kontainer dan aplikasi Kubernetes menggunakan sertifikat digital untuk menyediakan autentikasi dan enkripsi yang aman melalui TLS. Solusi yang diadopsi secara luas untuk manajemen siklus hidup sertifikat TLS di Kubernetes adalah cert-manager, add-on untuk Kubernetes yang meminta sertifikat, mendistribusikannya ke container Kubernetes, dan mengotomatiskan pembaruan sertifikat.

AWS Private CAmenyediakan plug-in open-source ke cert-manager, aws-privateca-issuer, untuk pengguna cert-manager yang ingin menyiapkan CA tanpa menyimpan kunci pribadi di cluster. Pengguna dengan persyaratan peraturan untuk mengontrol akses ke dan mengaudit operasi CA mereka dapat menggunakan solusi ini untuk meningkatkan auditabilitas dan mendukung kepatuhan. Anda dapat menggunakan plugin Penerbit Private CA AWS dengan Amazon Elastic Kubernetes Service (Amazon EKS), Kubernetes yang dikelola sendiri di AWS, atau di Kubernetes on premise.

Diagram di bawah ini menunjukkan beberapa opsi yang tersedia untuk menggunakan TLS di cluster Amazon EKS. Klaster contoh ini, yang berisi berbagai sumber daya, terletak di belakang penyeimbang beban. Angka-angka tersebut mengidentifikasi titik akhir yang mungkin untuk komunikasi yang diamankan dengan TLS, termasuk penyeimbang beban eksternal, pengendali ingress, pod individu dalam layanan, dan sepasang pod yang berkomunikasi secara aman satu sama lain.



1. Penghentian di penyeimbang beban.

   Elastic Load Balancing (ELB) adalah layanan AWS Certificate Manager terintegrasi, yang berarti Anda dapat menyediakan ACM dengan CA pribadi, menandatangani sertifikat dengannya, dan menginstalnya menggunakan konsol ELB. Solusi ini menyediakan komunikasi terenkripsi antara klien jarak jauh dan penyeimbang beban. Data diteruskan tanpa dienkripsi ke klaster EKS. Atau, Anda dapat memberikan sertifikat privat ke penyeimbang beban bukan AWS untuk mengakhiri TLS.

2. Penghentian pada pengontrol ingress Kubernetes.

Pengendali ingress berada di dalam klaster EKS sebagai penyeimbang beban asli dan router. Jika Anda telah menginstal cert-manager dan aws-privateca-issuer, dan menyediakan klaster dengan CA pribadi, Kubernetes dapat menginstal sertifikat TLS yang ditandatangani pada pengontrol, yang memungkinkannya berfungsi sebagai titik akhir klaster untuk komunikasi eksternal. Komunikasi antara penyeimbang beban dan pengendali ingress dienkripsi, dan setelah ingress, data melewati tanpa terenkripsi ke sumber daya klaster.

3. Penghentian di pod.

   Setiap pod adalah grup dari satu atau lebih kontainer yang berbagi penyimpanan dan sumber daya jaringan. Jika Anda telah menginstal cert-manager dan aws-privateca-issuer, dan menyediakan klaster dengan CA pribadi, Kubernetes dapat menginstal sertifikat TLS yang ditandatangani pada pod sesuai kebutuhan. Koneksi TLS yang berakhir pada pod tidak tersedia secara default untuk pod lain dalam klaster.

4. Mengamankan komunikasi antar pod.

   Anda juga dapat menyediakan beberapa pod dengan sertifikat yang memungkinkan mereka untuk berkomunikasi satu sama lain. Skenario berikut ini mungkin terjadi:

   - Penyediaan dengan sertifikat yang ditandatangani sendiri dan dibuat oleh Kubernetes. Ini mengamankan komunikasi antar pod, tetapi sertifikat yang ditandatangani sendiri tidak memenuhi persyaratan HIPAA atau FIPS.
   - Penyediaan dengan sertifikat yang ditandatangani oleh CA privat. Seperti pada angka 2 dan 3 di atas, ini memerlukan pemasangan cert-manager dan aws-privateca-issuer, dan menyediakan cluster dengan CA pribadi. Kubernetes kemudian dapat menginstal sertifikat TLS yang ditandatangani pada pod sesuai kebutuhan.

Penggunaan cross-account dari cert-manager

Administrator dengan akses lintas akun ke CA dapat menggunakan cert-manager untuk menyediakan klaster Kubernetes. Untuk informasi selengkapnya, lihat Praktik terbaik keamanan untuk akses lintas akun ke CA pribadi.

> **Note**
>
> Hanya templat AWS Private CA sertifikat tertentu yang dapat digunakan dalam skenario lintas akun. Lihat the section called “Templat sertifikat yang didukung ” daftar templat yang tersedia.

Templat sertifikat yang didukung

Tabel berikut mencantumkan AWS Private CA template yang dapat digunakan dengan cert-manager untuk menyediakan klaster Kubernetes.

Template yang didukung untuk Kubernetes	Support untuk penggunaan lintas akun
BlankEndEntityCertificate_CSRPassthrough/ definisi v1	
CodeSigningCertificate/V1 definisi	
EndEntityCertificate/V1 definisi	✓
EndEntityClientAuthCertificate/V1 definisi	✓
EndEntityServerAuthCertificate/V1 definisi	✓
Definisi SigningCertificate OCSP/V1	

Contoh solusi

Solusi integrasi berikut menunjukkan cara mengonfigurasi akses ke klaster AWS Private CA Amazon EKS.

- Cluster Kubernetes berkemampuan TLS dengan dan Amazon EKS AWS Private CA
- Menyiapkan enkripsi end-to-end TLS di Amazon EKS dengan AWS Load Balancer Controller baru

AWS Private CAKonektor untuk Active Directory

Apa itu AWS Private CA Konektor untuk Direktori Aktif

AWS Private CAdapat menerbitkan dan mengelola sertifikat yang diperlukan olehAWS Managed Microsoft AD. Menggunakan AWS Private CA Konektor untuk Direktori Aktif (Konektor untuk AD), Anda dapat mengganti perusahaan lokal atau CA pihak ketiga lainnya dengan CA pribadi terkelola yang Anda miliki, memberikan pendaftaran sertifikat kepada pengguna, grup, dan mesin yang dikelola oleh AD Anda.

Anda dapat menggunakan Connector for AD AWS Managed Microsoft AD untuk menghilangkan infrastruktur lokal dengan memigrasikan infrastruktur AD dan kunci publik ke cloud. Untuk pelanggan yang ingin menggunakan AWS Private CA AD lokal mereka, fitur ini juga terintegrasi dengan AWS Managed Microsoft AD Connector.

Topik

- Apakah Anda Konektor Pertama Kali untuk Pengguna AD?
- Mengakses Konektor untuk AD
- Harga untuk Konektor untuk AD

Apakah Anda Konektor Pertama Kali untuk Pengguna AD?

Jika Anda adalah pengguna pertama kali Connector for AD, kami sarankan Anda mulai dengan membaca bagian berikut:

- Apa itu AWS Private CA?
- Apa itu AWS Directory Service?

Mengakses Konektor untuk AD

Anda dapat mengakses Connector for AD melalui konsol,AWS CLI, dan API. Anda bisa mendapatkan akses ke konektor di konsol dari AWS Private CA konsol, dari AWS Directory Service konsol Anda, atau dengan mencari Konektor untuk AD di bilah AWS Management Console pencarian.

Harga untuk Konektor untuk AD

Konektor untuk AD ditawarkan sebagai fitur tanpa AWS Private CA biaya tambahan. Anda hanya membayar otoritas sertifikat swasta dan sertifikat yang Anda terbitkan melalui mereka.

Untuk informasi AWS Private CA harga terbaru, lihat AWS Private Certificate AuthorityHarga. Anda juga dapat menggunakan kalkulator AWS harga untuk memperkirakan biaya.

Memulai AWS Private CA Konektor untuk Direktori Aktif

Dengan AWS Private CA Connector for Active Directory, Anda dapat mengeluarkan sertifikat dari CA pribadi ke objek Active Directory untuk otentikasi dan enkripsi. Saat Anda membuat konektor, AWS Private Certificate Authority buat titik akhir untuk Anda di VPC agar objek direktori Anda meminta sertifikat.

Untuk mengeluarkan sertifikat, Anda membuat konektor dan templat yang kompatibel dengan iklan untuk konektor. Saat membuat templat, Anda dapat mengatur izin pendaftaran untuk grup iklan Anda.

Topik

Konektor untuk Prasyarat AD

Berikut ini diperlukan untuk Konektor untuk AD.

Untuk membuat Konektor untuk AD, Anda perlu mengatur AWS Private Certificate Authority (CA) dan direktori. Anda kemudian perlu berbagi CA pribadi dan direktori dengan Connector for AD service. Anda juga perlu mengatur grup keamanan dan kebijakan IAM dengan benar untuk membuat konektor.

AWS Private CA

Siapkan AWS Private CA untuk menerbitkan sertifikat ke objek direktori Anda. Untuk informasi selengkapnya, lihat Administrasi CA swasta.

AWS Private CAHarus dalam `Active` keadaan untuk membuat Konektor untuk AD. Nama subjek CA pribadi harus menyertakan nama umum. Pembuatan konektor akan gagal jika Anda mencoba membuat konektor menggunakan CA pribadi tanpa nama umum.

Direktori Aktif

Selain ituAWS Private CA, Anda memerlukan Direktori Aktif di cloud pribadi virtual (VPC). Konektor untuk AD mendukung jenis direktori berikut yang ditawarkan olehAWS Directory Service:

- AWSDirektori Aktif Microsoft Terkelola: Dengan AWS Directory Service Anda dapat menjalankan Microsoft Active Directory (AD) sebagai layanan terkelola. AWS Directory Service for Microsoft Active Directoryjuga disebut sebagaiAWS Managed Microsoft AD, didukung oleh Windows Server 2019. DenganAWS Managed Microsoft AD, Anda dapat menjalankan beban kerja sadar direktori di, AWS Cloud termasuk Microsoft Sharepoint dan aplikasi berbasis Net dan SQL Server kustom.
- Konektor Direktori Aktif: AD Connector adalah gateway direktori yang dapat mengarahkan permintaan direktori ke Microsoft Active Directory lokal Anda, tanpa menyimpan informasi apa pun di cloud. AD Connector mendukung koneksi ke domain yang dihosting di Amazon EC2

> **Note**
>
> Mendaftarkan pengontrol domain tidak didukung saat menggunakan Konektor untuk AD dengan. AWS Managed Microsoft AD

Akun Layanan

Saat menggunakan Directory Service AD Connector, Anda perlu mendelegasikan izin tambahan ke akun layanan. Setel daftar kontrol akses (ACL) pada akun layanan untuk memungkinkan kemampuan:

- Menambahkan dan menghapus Service Principal Name (SPN) ke dirinya sendiri
- Buat dan perbarui otoritas sertifikasi dalam wadah berikut:

```
#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
```

- Buat dan perbarui objek NT AuthCertificates Certification Authority (CA). Catatan: jika objek NT AuthCertificates CA ada maka Anda harus mendelegasikan izin untuk itu. Jika objek tidak ada maka Anda harus mendelegasikan kemampuan untuk membuat objek anak pada wadah Layanan Kunci Publik.

```
#objects
CN=NTAuthCertificates,CN=Public Key Services,CN=Services,CN=Configuration
```

> **Note**
>
> Jika Anda menggunakan AWS Managed Microsoft AD maka izin tambahan akan didelegasikan secara otomatis ketika Anda mengotorisasi Connector for AD service dengan direktori Anda. Anda dapat melewati langkah prasyarat ini.

Anda dapat menggunakan PowerShell skrip ini untuk mendelegasikan izin tambahan. Ini akan membuat objek otoritas AuthCertifiates sertifikasi NT. Ganti “myconnectoraccount” dengan nama akun layanan.

```
$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"
```

Kebijakan IAM

Untuk membuat konektor untuk AD, Anda memerlukan kebijakan IAM yang memungkinkan Anda membuat sumber daya konektor, membagikan CA pribadi Anda dengan layanan Connector for AD, dan mengotorisasi layanan Connector for AD dengan direktori Anda.

Ini adalah contoh kebijakan yang dikelola pengguna:

```
{
    "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/*"
        }
    ]
}
```

Konektor untuk AD memerlukan AWS RAM izin tambahan, untuk penggunaan konsol dan baris perintah.

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

Bagikan AWS Private CA dengan Konektor untuk AD

Anda harus berbagi AWS Private CA dengan layanan konektor dengan menggunakan berbagi prinsip AWS Resource Access Manager layanan.

Saat Anda membuat konektor di AWS konsol, pembagian sumber daya secara otomatis dibuat untuk Anda.

Saat Anda membuat berbagi sumber daya menggunakanAWS CLI, Anda akan menggunakan AWS RAM create-resource-share perintah.

Perintah berikut membuat pembagian sumber daya:

```
$  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
```

Prinsipal layanan yang memanggil CreateConnector memiliki izin penerbitan sertifikat pada PCA. Untuk mencegah prinsip layanan yang menggunakan Connector for AD memiliki akses umum ke AWS Private CA sumber daya Anda, batasi izin mereka menggunakan. `CalledVia`

Otorisasi Konektor untuk AD dengan direktori Anda

Anda mengotorisasi Konektor untuk layanan AD dengan direktori Anda sehingga konektor dapat berkomunikasi dengan direktori Anda. Untuk mengotorisasi Konektor untuk layanan AD, Anda membuat pendaftaran direktori. Untuk informasi selengkapnya tentang membuat pendaftaran direktori, lihat Mengelola pendaftaran direktori

Grup keamanan

Komunikasi antara VPC Anda dan Konektor untuk konektor AD sudah selesaiAWS PrivateLink, yang memerlukan grup keamanan dengan aturan masuk yang membuka port 443 TCP dan UDP pada VPC Anda. Anda akan diminta untuk grup keamanan ini ketika Anda membuat konektor. Anda dapat menentukan sumber sebagai kustom dan memilih blok CIDR VPC Anda. Anda dapat memilih untuk membatasi ini lebih lanjut (yaitu IP, CIDR, dan ID grup keamanan).

Buat konektor

Untuk instruksi, lihat prosedurnya Membuat konektor

Konfigurasikan kebijakan AD

Konektor untuk AD tidak dapat melihat atau mengelola konfigurasi objek kebijakan grup (GPO) pelanggan. GPO mengontrol perutean permintaan AD ke pelanggan atau ke otentikasi AWS Private CA atau server penjual sertifikat lainnya. Konfigurasi GPO yang tidak valid dapat mengakibatkan permintaan Anda dirutekan secara tidak benar. Terserah pelanggan untuk mengkonfigurasi dan menguji Konektor untuk konfigurasi AD.

Kebijakan Grup dikaitkan dengan Konektor, dan Anda dapat memilih untuk membuat beberapa Konektor untuk satu AD. Terserah Anda untuk mengelola kontrol akses ke setiap konektor jika konfigurasi kebijakan grupnya berbeda.

Keamanan panggilan pesawat data tergantung pada Kerberos dan konfigurasi VPC Anda. Siapa pun yang memiliki akses ke VPC dapat melakukan panggilan pesawat data selama mereka diautentikasi ke AD yang sesuai. Ini ada di luar batas AWSAuth dan mengelola otorisasi dan otentikasi terserah Anda, pelanggan.

Di Active Directory, ikuti langkah-langkah di bawah ini untuk membuat GPO yang menunjuk ke URI yang dihasilkan saat Anda membuat konektor. Langkah ini diperlukan untuk menggunakan Connector for AD dari konsol atau baris perintah.

Konfigurasikan GPO.

1. Buka Server Manager di DC
2. Buka Alat dan pilih Manajemen Kebijakan Grup di sudut kanan atas konsol.
3. Pergi ke Hutan> Domain. Pilih nama domain Anda dan klik kanan pada domain Anda. Pilih Buat GPO di domain ini, dan tautkan di sini... dan masukkan `PCA GPO` untuk namanya.
4. GPO yang baru dibuat sekarang akan terdaftar di bawah nama domain Anda.
5. Pilih PCA GPO dan pilih Edit. Jika kotak dialog terbuka dengan pesan peringatan Ini adalah tautan dan perubahan itu akan disebarkan secara global, akui pesan untuk melanjutkan. Editor Manajemen Kebijakan Grup harus terbuka.
6. Di Editor Manajemen Kebijakan Grup, buka Konfigurasi Komputer > Kebijakan > Pengaturan Windows> Pengaturan Keamanan > Kebijakan Kunci Publik (pilih folder).
7. Pergi ke jenis objek dan pilih Certificate Services Client - Certificate Enrollment Policy
8. Dalam opsi, ubah Model Konfigurasi ke Diaktifkan.
9. Konfirmasikan bahwa Kebijakan Pendaftaran Direktori Aktif dicentang dan Diaktifkan. Pilih Tambahkan.
10. Jendela Server Kebijakan Pendaftaran Sertifikat harus terbuka.
11. Masukkan titik akhir server kebijakan pendaftaran sertifikat yang dihasilkan saat Anda membuat konektor di bidang URI kebijakan server Enter enrollment.
12. Biarkan Jenis Otentikasi sebagai Windows terintegrasi.
13. Pilih Validasi. Setelah validasi berhasil, pilih Tambah. Kotak dialog ditutup.

14. Kembali ke Certificate Services Client - Certificate Enrollment Policy dan centang kotak di samping konektor yang baru dibuat untuk memastikan bahwa konektor adalah kebijakan pendaftaran default
15. Pilih Kebijakan Pendaftaran Direktori Aktif dan pilih Hapus.
16. Di kotak dialag konfirmasi, pilih Ya untuk menghapus otentikasi berbasis LDAP.
17. Pilih Terapkan dan OK pada jendela Certificate Services Client > Certificate Enrollment Policy dan tutup.
18. Buka folder Kebijakan Kunci Publik dan pilih Certificate Services Client - Auto-Enrollment.
19. Ubah opsi Model Konfigurasi ke Diaktifkan.
20. Konfirmasikan bahwa Perpanjang sertifikat kedaluwarsa dan Sertifikat Pembaruan keduanya diperiksa. Biarkan pengaturan lain apa adanya.
21. Pilih Terapkan, lalu OK, dan tutup kotak dialog.

Konfigurasikan Kebijakan Kunci Publik untuk konfigurasi pengguna selanjutnya. Buka Konfigurasi Pengguna> Kebijakan > Pengaturan Windows> Pengaturan Keamanan > Kebijakan Kunci Publik. Ikuti prosedur yang diuraikan dari langkah 6 hingga langkah 21 untuk mengonfigurasi Kebijakan Kunci Publik untuk konfigurasi pengguna.

Setelah Anda selesai mengonfigurasi GPO dan Kebijakan Kunci Publik, objek dalam domain akan meminta sertifikat dari AWS Private CA Connector for AD dan mendapatkan sertifikat yang dikeluarkan oleh. AWS Private CA

Buat template

Untuk instruksi, lihat prosedurnya Membuat template konektor

Kelola izin grup AD

Untuk instruksi, lihat prosedurnya Mengelola grup AD dan izin untuk templat

AWS Private CAKonektor untuk Prosedur Direktori Aktif

Prosedur di bagian ini menjelaskan cara membuat konektor, mengkonfigurasi template, dan mengintegrasikan dengan AWS Private CA dan Active Directory. Anda dapat melakukan operasi ini dari AWS Private CA Connector for AD console, dengan menggunakan bagian Connector for ADAWS CLI, atau dengan menggunakan AWS Private CA Connector for AD API.

> **Note**
>
> Meskipun AWS Private CA Connector for AD terintegrasi erat denganAWS Private CA, kedua layanan memiliki API terpisah. Untuk informasi selengkapnya, lihat Referensi AWS Private Certificate Authority API dan Referensi API AWS Private CA Connector for Active Directory.

Prosedur

- Membuat konektor
- Membuat template konektor
- Konektor daftar untuk Active Directory
- Daftar templat konektor
- Lihat detail konektor
- Lihat detail templat konektor
- Mengelola pendaftaran direktori
- Mengelola grup AD dan izin untuk templat
- Mengkonfigurasi nama utama layanan
- Tagging Connector untuk sumber daya AD

Membuat konektor

Gunakan prosedur berikut untuk membuat konektor menggunakan konsol, baris perintah, atau API untuk AWS Private CA Connector for Active Directory.

Membuat konektor (konsol)

Lengkapi prosedur berikut untuk membuat dan mengkonfigurasi konektor menggunakan AWS konsol.

Tugas

- Buka konsol
- Buka Buat konektor
- Pilih atau buat direktori

- Pilih CA pribadi
- Konfigurasikan penandaan
- Tinjau dan buat

Buka konsol

Masuk ke AWS akun Anda dan buka konsol AWS Private CA Connector for Active Directory di**https://console.aws.amazon.com/pca-connector-ad/home**.

Buka Buat konektor

Pada halaman landing layanan pertama kali atau halaman Konektor untuk Direktori Aktif, pilih Buat konektor.

Pilih atau buat direktori

Pada halaman Create Private CA Connector for Active Directory, berikan informasi di bagian Active Directory.

- Di bawah Pilih jenis Active Directory Anda, pilih salah satu dari dua jenis yang tersedia:
  - AWS Directory Service for Microsoft Active Directory- Menentukan Active Directory dikelola olehAWS Directory Service.
  - Active Directory lokal dengan AWS AD Connector — Menggunakan AD Connector untuk mengakses Active Directory yang Anda host di lokasi.
- Di bawah Pilih direktori Anda, pilih direktori Anda dari daftar.

  Atau, Anda dapat memilih Buat direktori, yang membuka AWS Directory Service konsol di jendela baru. Setelah Anda selesai membuat direktori baru, kembali ke konsol AWS Private CA Connector for Active Directory dan segarkan daftar direktori. Direktori baru Anda harus tersedia untuk dipilih.

  > **Note**
  >
  > Saat membuat direktori, perhatikan bahwa Connector for AD hanya mendukung jenis direktori berikut yang ditawarkan di AWS Directory Service konsol:
  >
  > - AWSMicrosoft AD yang dikelola
  > - Konektor AD

- Di bawah Pilih grup keamanan untuk titik akhir VPC, pilih grup keamanan dari daftar.

Atau, Anda dapat memilih Buat grup keamanan, yang membuka konsol Amazon EC2 ke halaman Buat grup keamanan di jendela baru. Setelah Anda selesai membuat grup keamanan, kembali ke konsol AWS Private CA Connector for Active Directory dan segarkan daftar grup keamanan. Grup keamanan baru Anda harus tersedia untuk dipilih.

Pilih CA pribadi

Di bagian Otoritas sertifikat pribadi, pilih CA pribadi dari daftar.

Atau, Anda dapat memilih Buat CA Pribadi, yang membuka AWS Private CA konsol ke halaman otoritas sertifikat pribadi di jendela baru. Setelah Anda selesai membuat CA, kembali ke konsol AWS Private CA Connector for Active Directory dan segarkan daftar CA. CA baru Anda harus tersedia untuk dipilih.

Konfigurasikan penandaan

Di tag — panel opsional, Anda dapat menerapkan dan menghapus metadata pada sumber daya AD Anda. Tag adalah pasangan string nilai kunci di mana kunci harus unik untuk sumber daya dan nilainya opsional. Panel menampilkan tag yang ada untuk sumber daya dalam tabel. Tindakan berikut didukung.

- Pilih Kelola tag untuk membuka halaman Kelola tag.
- Pilih Tambahkan tag baru untuk membuat tag. Isi bidang Key dan, secara opsional, bidang Value. Pilih Simpan perubahan untuk menerapkan tag.
- Pilih tombol Hapus di samping tag untuk menandainya untuk dihapus, dan pilih Simpan perubahan untuk mengonfirmasi.

Tinjau dan buat

Setelah memberikan informasi yang diperlukan dan meninjau pilihan Anda, pilih Buat konektor. Ini membuka halaman detail Konektor untuk Direktori Aktif di mana dapat melihat kemajuan konektor Anda saat dibuat.

Setelah proses pembuatan konektor selesai, tetapkan nama utama layanan.

Buat konektor untuk Active Directory (AWS CLI)

Untuk membuat konektor untuk Active Directory dengan CLI, gunakan perintah create-connector di bagian Connector for Active Directory dari AWS Private CA file. AWS CLI

Buat konektor untuk Active Directory (API)

Untuk membuat konektor Active Directory dengan API, gunakan CreateConnectortindakan di AWS Private CA Connector for Active Directory API.

Membuat template konektor

Membuat template konektor (konsol)

Selesaikan prosedur berikut untuk membuat dan mengkonfigurasi templat konektor menggunakanAWSkonsol.

Tugas

- Buka konsol
- Pilih konektor
- Temukan bagian template
- Metode pembuatan template
- Pengaturan template
- Konfigurasikan pengaturan sertifikat
- Konfigurasikan pengaturan penanganan permintaan dan pendaftaran
- Konfigurasikan ekstensi penggunaan kunci
- Tetapkan kebijakan aplikasi
- Konfigurasikan kebijakan aplikasi khusus
- Konfigurasikan pengaturan kriptografi
- Konfigurasikan grup dan izin
- Konfigurasikan templat pengganti
- Konfigurasikan penandaan
- Tinjau dan buat

Buka konsol

Masuk ke AndaAWSakun dan bukaAWS Private CAKonektor untuk konsol Active Directory di**https://console.aws.amazon.com/pca-connector-ad/home**.

Pilih konektor

Pilih konektor dariKonektor untuk Active Directorydaftar dan kemudian pilihLihat detail.

Temukan bagian template

Pada halaman detail untuk konektor, temukanTemplatebagian dan kemudian pilihBuat template.

Metode pembuatan template

PadaBuat templatehalaman, diMetode pembuatan templatebagian, pilih salah satu opsi metode.

- Mulai dari template yang telah ditentukan(default) - Pilih dari daftar template yang telah ditentukan untuk aplikasi AD:
  - Penandatanganan Kode
  - Komputer
  - Otentikasi Pengontrol Domain
  - Agen Pemulihan EFS
  - Agen Pendaftaran
  - Agen Pendaftaran (Komputer)
  - IPsec
  - Otentikasi Kerberos
  - Server RAS dan IAS
  - Logon Kartu Pintar
  - Penandatanganan Daftar Kepercayaan
  - Tanda Tangan Pengguna
  - Otentikasi Workstation
- Mulai dari template yang sudah ada yang Anda buat— Pilih dari daftar template khusus yang Anda buat sebelumnya.
- Mulai dari template kosong— Pilih opsi ini untuk mulai membuat template yang sama sekali baru.

Pengaturan template

DiPengaturan templatebagian, memberikan informasi berikut:

- Nama template— Nama template

- Versi skema template— Versi skema template. Versi skema mempengaruhi ketersediaan opsi template sebagai berikut:

  Skema versi 2

  - Mendukung kompatibilitas klien Windows XP /Windows Server 2003 dan yang lebih tinggi.
  - Hanya mendukung Penyedia Layanan Kriptografi Warisan.

  Skema versi 3

  - Mendukung kompatibilitas klien Windows Vista/Windows Server 2008 dan yang lebih tinggi.
  - Mendukung memungkinkan pemohon untuk memperbarui dengan kunci yang ada.
  - Hanya mendukung Penyedia Penyimpanan Kunci.

  Skema versi 4

  - Mendukung kompatibilitas klien Windows 8/Windows Server 2012 dan yang lebih tinggi.
  - Mendukung memungkinkan pemohon untuk memperbarui dengan kunci yang ada.
  - Mendukung Penyedia Layanan Kriptografi Warisan dan Penyedia Penyimpanan Kunci.
- Kompatibilitas klien— Tingkat sistem operasi minimum yang kompatibel dengan template. Pilih salah satu opsi yang tercantum:
  - Windows XP /Windows Server 2003
  - Windows Visa/Windows Server 2008
  - Windows 7/Windows Server 2008 R2
  - Windows 8 dan lebih tinggi/Windows Server 2012
  - Windows 8 dan lebih tinggi/Windows Server 2012 R2
  - Windows 8 dan lebih tinggi/Windows Server 2016 dan lebih tinggi

Konfigurasikan pengaturan sertifikat

DiPengaturan sertifikatbagian, tentukan pengaturan berikut untuk sertifikat berdasarkan template ini.

- Jenis sertifikat— Tentukan apakah akan membuatPenggunaatauKomputersertifikat.
- Pendaftaran otomatis— Pilih apakah akan mengaktifkan pendaftaran otomatis untuk sertifikat berdasarkan template ini.
- Masa berlaku— Tentukan periode validitas sertifikat sebagai nilai bilangan bulat jam, hari, minggu, bulan, atau tahun. Nilai minimum adalah 2 jam.

- Periode perpanjangan— Tentukan periode perpanjangan sertifikat sebagai nilai bilangan bulat jam, hari, minggu, bulan, atau tahun. Masa perpanjangan tidak boleh lebih dari 75% dari masa berlaku.
- Nama subjek— Pilih satu atau lebih opsi untuk dimasukkan dalam nama subjek berdasarkan informasi yang terkandung dalam Active Directory.

> **Note**
>
> Setidaknya satu nama subjek atau opsi nama alternatif subjek harus ditentukan.

  - Nama umum
  - DNS sebagai nama umum
  - Jalur direktori
  - Email
- Nama alternatif subjek— Pilih satu atau lebih opsi untuk dimasukkan dalam nama alternatif subjek berdasarkan informasi yang terkandung dalam Active Directory.

> **Note**
>
> Setidaknya satu nama subjek atau opsi nama alternatif subjek harus ditentukan.

  - Direktori GUID
  - Nama DNS
  - Domain DNS
  - Email
  - Nama utama layanan (SPN)
  - Nama Utama Pengguna (UPN)

Konfigurasikan pengaturan penanganan permintaan dan pendaftaran

DiOpsi penanganan dan pendaftaran permintaan sertifikatbagian, tentukan tujuan sertifikat berdasarkan template, memilih salah satu opsi berikut.

- Tanda tangan

- Enkripsi
- Tanda tangan dan enkripsi
- Tanda tangan dan logon kartu pintar

Selanjutnya, pilih mana dari fitur berikut untuk diaktifkan. Opsi bervariasi tergantung pada tujuan sertifikat.

- Hapus sertifikat yang tidak valid (jangan arsipkan)
- Sertakan algoritma simetris
- Kunci pribadi yang dapat diekspor

Terakhir, pilih opsi pendaftaran sertifikat. Opsi bervariasi tergantung pada tujuan sertifikat.

- Tidak diperlukan masukan pengguna
- Prompt pengguna selama pendaftaran
- Meminta pengguna selama pendaftaran dan memerlukan masukan pengguna

Konfigurasikan ekstensi penggunaan kunci

DiPengaturan ekstensi penggunaan kuncibagian, pilih opsi untuk penggunaan tanda tangan dan penggunaan kunci enkripsi.

Penggunaan kunci tanda tangan

- Tanda tangan digital
- Tanda tangan adalah bukti asal (nonrepudiation)

Penggunaan kunci enkripsi

- Izinkan pertukaran kunci tanpa enkripsi kunci (perjanjian kunci)
- Izinkan pertukaran kunci hanya dengan enkripsi kunci (encipherment kunci)
- Izinkan enkripsi data pengguna (encipherment data)

Anda juga dapat memilih untukJadikan ekstensi penggunaan kunci pentinguntuk kedua jenis kunci.

Tetapkan kebijakan aplikasi

DiKebijakan aplikasibagian, pilih semua kebijakan aplikasi yang berlaku. Kebijakan yang tersedia tercantum di beberapa halaman. Beberapa kebijakan mungkin telah dipilih sebelumnya karena pengaturan sebelumnya.

Konfigurasikan kebijakan aplikasi khusus

DiKebijakan aplikasi khususbagian, Anda dapat menambahkan OID kustom ke template, dan menentukan apakah ekstensi kebijakan aplikasi sangat penting.

Konfigurasikan pengaturan kriptografi

DiPengaturan kriptografibagian, pilih kategori pengaturan kriptografi berikut untuk sertifikat berdasarkan template ini.

1. Konten di bagian atas bagian ditentukan olehMetode pembuatan template danPengaturan template yang Anda pilih sebelumnya.

   - Jika Anda menerima defaultTemplate versi 2diPengaturan template , maka pesan status berikut ditampilkan di sini:
     - Kategori penyedia kriptografi
     - Penyedia layanan kriptografi lama

     Dalam hal ini tidak ada pengaturan untuk dikonfigurasi dan Anda dapat melanjutkan ke langkah berikutnya.
   - Jika Anda menentukanTemplate versi 3diPengaturan template , maka pesan status berikut ditampilkan di sini:
     - Kategori penyedia kriptografi
     - Penyedia penyimpanan kunci

     Anda juga harus memilihAlgoritma kuncidari opsi yang tercantumECDH_P256,ECDH_P384,ECDH_P521, danRSA.
   - Jika Anda menentukanTemplate versi 4diPengaturan template , maka Anda harus memilih antaraPenyedia penyimpanan kuncidanPenyedia layanan kriptografi lama. Jika Anda memilihPenyimpanan kunci menyediakan, makaAlgoritma kuncijuga harus dipilih dari opsi yang tercantumECDH_P256,ECDH_P384,ECDH_P521, danRSA.
2. Ukuran kunci minimum (bit)— Tentukan ukuran kunci minimum. Pengaturan ini akan memengaruhi penyedia kriptografi mana yang tersedia.

3. Pilih penyedia kriptografi mana yang dapat digunakan untuk permintaan— Pilih salah satu dari dua opsi yang tersedia:

   - Permintaan dapat menggunakan penyedia apa pun yang tersedia di komputer subjek
   - Permintaan harus menggunakan salah satu penyedia yang dipilih berikut

     Memilih opsi ini membukaPenyedia kriptografidaftar. Anda dapat memilih dan memprioritaskan penyedia menggunakan tombol diMemesankolom. Penyedia berikut didukung:

     - Penyedia Kriptografi Pangkalan Microsoft v1.0
     - Microsoft Base DSS dan Penyedia Kriptografi Diffie-Hellman
     - Penyedia Crypto Kartu Cerdas Pangkalan Microsoft
     - Penyedia Kriptografi Microsoft DH SChannel
     - Penyedia Kriptografi yang Ditingkatkan Microsoft v1.0
     - Microsoft Enhanced DSS dan Penyedia Kriptografi Diffie-Hellman
     - Microsoft Enhanced RSA dan Penyedia Kriptografi AES
     - Penyedia Kriptografi Microsoft RSA SChannel

Konfigurasikan grup dan izin

DiGrup dan izinbagian, Anda dapat melihat template grup yang ada dan izin untuk pendaftaran, atau Anda dapat memilihTambahkan grup dan izin barutombol untuk menambahkan yang baru. Tombol membuka formulir yang membutuhkan informasi berikut:

- Nama tampilan
- Pengidentifikasi keamanan(SID)
- Mendaftar, dengan opsi IZINKAN | TOLAK | TIDAK DIATUR
- Mendaftar otomatis, dengan opsi IZINKAN | TOLAK | TIDAK DIATUR

Konfigurasikan templat pengganti

DiMenggantikan templatebagian, Anda dapat memberi tahu Active Directory bahwa template saat ini menggantikan satu atau beberapa templat yang dibuat di AD. Terapkan template pengganti dengan memilihTambahkan template dari Active Directory untuk menggantikandan menentukan nama umum dari template pengganti.

Konfigurasikan penandaan

DiTag - opsionalpanel, Anda dapat menerapkan dan menghapus metadata pada sumber daya AD Anda. Tag adalah pasangan string nilai kunci di mana kunci harus unik untuk sumber daya dan nilainya opsional. Panel menampilkan tag yang ada untuk sumber daya dalam tabel. Tindakan berikut didukung.

- PilihMengelola taguntuk membukaMengelola taghalaman.
- Pilih Tambahkan tag baru untuk membuat tag. IsiKuncibidang dan, secara opsional,Nilailapangan. PilihSimpan perubahanuntuk menerapkan tag.
- PilihHapustombol di sebelah tag untuk menandainya untuk dihapus, dan pilihSimpan perubahanuntuk mengkonfirmasi.

Tinjau dan buat

Setelah memberikan informasi yang diperlukan dan meninjau pilihan Anda, pilihBuat template. Ini terbukaRincian template, di mana Anda dapat meninjau pengaturan template baru, mengedit atau menghapus template, mengelola grup dan izin, mengelola template yang digantikan, mengelola tag, dan mengatur pendaftaran ulang otomatis untuk pemegang sertifikat.

Membuat template konektor (CLI)

Gunakanbuat-templateperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Membuat template konektor (API)

Gunakan CreateTemplateaksi diAWS Private CAKonektor untuk Active Directory API.

Konektor daftar untuk Active Directory

Anda dapat menggunakanAWS Private CAKonektor untuk konsol Active Directory atauAWS CLIuntuk membuat daftar konektor yang Anda miliki.

Untuk membuat daftar konektor Anda menggunakan konsol

1. Masuk ke AndaAWSakun dan bukaAWS Private CAKonektor untuk konsol Active Directory di**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Tinjau informasi diKonektor untuk Active Directorydaftar. Anda dapat menavigasi melalui beberapa halaman konektor menggunakan nomor halaman di kanan atas. Setiap konektor menempati baris yang menampilkan kolom informasi berikut secara default.

- ID Konektor— ID unik konektor.
- Nama direktori— Sumber daya Direktori Aktif yang terkait dengan konektor.
- Status konektor- Status konektor. Nilai yang mungkin adalah:Menciptakan|Aktif|Menghapus|Gagal.
- Status nama utama layanan— Status nama utama layanan (SPN) yang terkait dengan konektor. Nilai yang mungkin adalah:Menciptakan|Aktif|Menghapus|Gagal.
- Status pendaftaran direktori— Status pendaftaran direktur asosiasi. Nilai yang mungkin adalah:Menciptakan|Aktif|Menghapus|Gagal.
- Dibuat di— Cap waktu pada pembuatan konektor.

Dengan memilih ikon roda gigi di sudut kanan atas konsol, Anda dapat menyesuaikan jumlah konektor yang ditampilkan pada halaman menggunakanUkuran halamanpreferensi.

Untuk membuat daftar konektor Anda menggunakanAWS CLI

Gunakandaftar-konektorperintah untuk membuat daftar konektor Anda.

Untuk membuat daftar konektor Anda menggunakan API

GunakanListConnectorsaksi diAWS Private CAKonektor untuk Active Directory API.

Daftar templat konektor

Anda dapat menggunakanAWS Private CAKonektor untuk konsol Active Directory atauAWS CLIuntuk daftar template untuk konektor yang Anda miliki. Template konektor didasarkan padaAWS Private CA BlankEndEntityCertificate_apiPassthrough/v1templat.

Untuk membuat daftar template Anda menggunakan konsol

1. Masuk ke AndaAWSakun dan bukaAWS Private CAKonektor untuk konsol Active Directory di**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Pilih konektor dariKonektor untuk Active Directorydaftar dan kemudian pilihLihat detail.
3. Pada halaman detail konektor, tinjau informasi diTemplatebagian. Anda dapat menavigasi melalui beberapa halaman template menggunakan nomor halaman di kanan atas. Setiap template menempati baris yang menampilkan kolom informasi berikut.

- Nama template— Nama template yang dapat dibaca manusia.

- Status templat— Status template. Nilai yang mungkin adalah:Aktif|Menghapus.
- ID Templat— Pengidentifikasi unik dari template.

Untuk membuat daftar template Anda menggunakanAWS CLI

Gunakandaftar-templateperintah untuk daftar template untuk konektor yang ditentukan.

Untuk membuat daftar template Anda menggunakan API

Gunakan ListTemplatesaksi diAWS Private CAKonektor untuk Active Directory API untuk mencantumkan templat untuk konektor yang ditentukan.

Lihat detail konektor

Gunakan prosedur berikut untuk melihat detail konfigurasi konektor di konsol, baris perintah, atau APIAWS Private CAKonektor untuk Active Directory.

Lihat konektor (konsol)

Untuk melihat detail untuk konektor (konsol)

1. Masuk ke AndaAWSakun dan bukaAWS Private CAKonektor untuk konsol Active Directory di**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Pilih konektor dariKonektor untuk Active Directorydaftar dan kemudian pilihLihat detail.
3. Pada halaman detail konektor, tinjau informasi di panel Detail konektor, yang mencakup hal-hal berikut:
   - ID Konektor
   - Status konektor
   - Detail status tambahan
   - Konektor ARN
   - Titik akhir server kebijakan pendaftaran sertifikat
   - Nama direktori
   - ID Direktori
   - AWS Private CAsubjek
   - AWS Private CAstatus
   - Titik akhir VPC dan grup keamanan

4. DiTemplatepanel, Anda dapat membuat atau mengelola template yang terkait dengan konektor.
5. DariNama utama layanan (SPN)panel, Anda dapat melihat nama prinsip layanan yang terkait dengan konektor.
6. DariPendaftaran Direktoripanel, Anda dapat melihat atau mengubah pendaftaran direktori yang terkait dengan konektor.
7. DariTag -pilihanpanel, Anda dapat membuat atau mengelola tag yang terkait dengan konektor.

Lihat konektor (CLI)

Gunakandapatkan-konektorperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Lihat konektor (API)

GunakanGetConnectoraksi diAWS Private CAKonektor untuk Active Directory API.

Lihat detail templat konektor

Gunakan prosedur berikut untuk melihat detail konfigurasi templat konektor menggunakan konsol, baris perintah, atau APIAWS Private CAKonektor untuk Active Directory

Lihat template (konsol)

Untuk melihat detail untuk template konektor (konsol)

1. Masuk ke AndaAWSakun dan bukaAWS Private CAKonektor untuk konsol Active Directory di**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Pilih konektor dariKonektor untuk Active Directorydaftar dan kemudian pilihLihat detail.
3. Pada halaman detail konektor, tinjau informasi diTemplatebagian, dan pilih template yang ingin Anda periksa. Kemudian pilihLihat detail.
4. Pada halaman detail,Rincian templatepanel menampilkan informasi berikut tentang template:
   - Nama template
   - ID Templat
   - Status templat
   - Versi skema template

- Versi template
- Templat ARN
- Jenis sertifikat
- Pendaftaran otomatis diaktifkan
- Masa berlaku
- Periode perpanjangan
- Persyaratan nama subjek
- Persyaratan nama alternatif subjek
- Permintaan sertifikat dan pengaturan pendaftaran
- Kategori penyedia kriptografi
- Algoritma kunci
- Ukuran kunci minimum (bit)
- Algoritma hash
- Penyedia kriptografi
- Pengaturan ekstensi penggunaan kunci

Dari panel ini, Anda juga dapat melakukan tindakan berikut menggunakanSunting,Hapus, danTindakantombol.

- Sunting
- Hapus
- Mengelola grup dan izin— Untuk informasi lebih lanjut, lihatKonfigurasikan grup dan izin.
- Kelola templat yang digantikan— Untuk informasi lebih lanjut, lihatTinjau dan buat.
- Kelola tag— Untuk informasi lebih lanjut, lihatTagging Connector untuk sumber daya AD.
- Daftarkan kembali semua pemegang sertifikat— Pengaturan ini memungkinkan versi utama template ditingkatkan secara otomatis. Semua anggota grup Active Directory yang diizinkan untuk mendaftar dengan template akan menerima sertifikat baru yang dikeluarkan menggunakan template tersebut. Untuk informasi lebih lanjut, lihat UpdateTemplateAPI.

5. Panel bawah menampilkan deretan tab yang memungkinkan perubahan pada konfigurasi template.

   - Grup dan izin— Lihat dan kelola izin untuk grup Active Directory untuk mendaftarkan sertifikat menggunakan template ini. Untuk informasi lebih lanjut, lihatKonfigurasikan grup dan izin.

- Kebijakan aplikasi— Lihat dan kelola kebijakan aplikasi template. Untuk informasi lebih lanjut, lihatTetapkan kebijakan aplikasi.
- Template yang digantikan— Lihat dan kelola template yang digantikan. Untuk informasi lebih lanjut, lihatTinjau dan buat.
- Tagpilihan— Lihat dan kelola penandaan pada template ini. Untuk informasi selengkapnya, lihat Tagging Connector untuk sumber daya AD.

Untuk melihat detail untuk template konektor (AWS CLI)

Lihat template (CLI)

Gunakandapatkan-templateperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Lihat template (API)

Untuk melihat detail untuk template konektor (API)

Gunakan GetTemplateaksi diAWS Private CAKonektor untuk Active Directory API.

Mengelola pendaftaran direktori

Untuk mengelola pendaftaran direktori (konsol)

Pendaftaran direktori untuk konektor dapat dikelola dari tingkat atasAWS Private CAKonektor untuk konsol Active Directory. Topik ini berjalan melalui opsi manajemen yang tersedia.

1. Masuk ke AndaAWSakun dan bukaAWS Private CAKonektor untuk konsol Active Directory di**https://console.aws.amazon.com/pca-connector-ad/home**.
2. Di area navigasi kiri, pilihPendaftaran direktori.
3. ThePendaftaran direktorihalaman menampilkan tabel direktori terdaftar dengan bidang-bidang berikut:

   - ID Direktori— ID unik dari direktori
   - Nama direktori— Nama situs domain direktori
   - Jenis direktori
   - Terdaftar— Status pendaftaran. Nilai yang didukung adalah CREATING | ACTIVE | DELETING | FAILED.

- Status direktori— Status direktori

Gunakan bisa menggunakanDaftar direktoriuntuk membuat pendaftaran baru.

4. Anda dapat memilih salah satu pendaftaran yang terdaftar untuk mengelolanya. Hal ini memungkinkanLihat detail pendaftarandanDirektori deregistertombol. TheLihat detail pendaftarantombol membuka halaman detail untuk pendaftaran.
5. TheDetail pendaftaran direktoripanel menampilkan informasi berikut:

   - Nama situs domain direktori
   - ID Direktori— ID unik dari direktori. Memilih tautan akan membawa Anda keAWS Directory Servicekonsol.
   - Jenis direktori
   - Status— Status direktori
   - Pendaftaran direktori ARN— Nama sumber daya Amazon dari pendaftaran direktori
   - Informasi status tambahan
6. DiKonektor dan nama utama layanan (SPN)panel, Anda dapat mengelola SPN untuk konektor. Untuk informasi lebih lanjut, lihatLihat detail konektor.
7. DiTag - opsionalpanel, Anda dapat menerapkan dan menghapus metadata pada sumber daya AD Anda. Tag adalah pasangan string nilai kunci di mana kunci harus unik untuk sumber daya dan nilainya opsional. Panel menampilkan tag yang ada untuk sumber daya dalam tabel. Tindakan berikut didukung.

   - PilihKelola taguntuk membukaKelola taghalaman.
   - Pilih Tambahkan tag baru untuk membuat tag. IsiKuncibidang dan, secara opsional,Nilailapangan. PilihSimpan perubahanuntuk menerapkan tag.
   - PilihHapustombol di sebelah tag untuk menandainya untuk dihapus, dan pilihSimpan perubahanuntuk mengkonfirmasi.

Untuk mengelola pendaftaran direktori (CLI)

Buat: Gunakancreate-directory-registrationperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Ambil:get-directory-registrationperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Daftar:list-directory-registrationsperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Hapus:delete-directory-registrationperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Untuk mengelola pendaftaran direktori (API)

Buat:CreateDirectoryRegistrationaksi diAWS Private CAKonektor untuk Active Directory API.

Ambil:GetDirectoryRegistrationaksi diAWS Private CAKonektor untuk Active Directory API.

Daftar:ListDirectoryRegistrationsaksi diAWS Private CAKonektor untuk Active Directory API.

Hapus:DeleteDirectoryRegistrationaksi diAWS Private CAKonektor untuk Active Directory API.

Mengelola grup AD dan izin untuk templat

Untuk mengelola grup template dan izin (konsol)

Grup dan izin untuk template yang ada dapat dikelola dari halaman detail template. Untuk informasi selengkapnya, lihat Lihat detail templat konektor.

Tetapkan izin pada grup mana yang dapat atau tidak dapat mendaftarkan sertifikat untuk templat tertentu. Anda memberikan pengenal keamanan (SID) grup. Kemudian Anda mengatur izin pendaftaran dan pendaftaran otomatis untuk grup. Untuk pendaftaran otomatis, pendaftaran dan pendaftaran otomatis harus disetel ke “Izinkan.”

Cari pengenal keamanan grup di Active Directory

Anda dapat menggunakan skrip di bawah ini untuk mencari pengidentifikasi keamanan grup di Active Directory.

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

Untuk mengelola grup template dan izin (CLI)

Buat: perintah create-template-group-access-control-entry di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

Perbarui: perintah update-template-group-access-control-entry di bagian AWS Private CA Connector for Active Directory dari. AWS CLI

Ambil: perintah get-template-group-access-control-entry di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

List: perintah list-template-group-access-control-entries di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

Hapus: perintah delete-template-group-access-control-entries di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

Untuk mengelola grup template dan izin (API)

Create: CreateTemplateGroupAccessControlEntryaction di AWS Private CA Connector for Active Directory API.

Update: UpdateTemplateGroupAccessControlEntrytindakan di AWS Private CA Connector for Active Directory API.

Ambil: GetTemplateGroupAccessControlEntrytindakan di AWS Private CA Connector for Active Directory API.

Daftar: ListTemplateGroupAccessControlEntriestindakan di AWS Private CA Connector for Active Directory API.

Hapus: DeleteTemplateGroupAccessControlEntrytindakan di AWS Private CA Connector for Active Directory API.

Mengkonfigurasi nama utama layanan

Untuk mengelola nama utama layanan (konsol)

Nama utama layanan (SPN) dari konektor AD yang ada dapat dikelola dari halaman detail konektor. Untuk informasi selengkapnya, lihat Mengelola pendaftaran direktoriLihat detail konektor

Untuk mengelola nama utama layanan (CLI)

Buat: create-service-principal-nameperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Ambil: get-service-principal-nameperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Daftar: list-service-principal-namesperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Hapus: delete-service-principal-nameperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Untuk mengelola nama utama layanan (API)

Buat: CreateServicePrincipalNameaksi diAWS Private CAKonektor untuk Active Directory API.

Ambil: GetServicePrincipalNameaksi diAWS Private CAKonektor untuk Active Directory API.

Daftar: ListServicePrincipalNamesaksi diAWS Private CAKonektor untuk Active Directory API.

Hapus: DeleteServicePrincipalNameaksi diAWS Private CAKonektor untuk Active Directory API.

Tagging Connector untuk sumber daya AD

Anda dapat menerapkan tag ke konektor, templat, dan pendaftaran direktori Anda. Penandaan menambahkan metadata ke sumber daya yang dapat membantu organisasi dan manajemen.

Untuk mengelola penandaan sumber daya (konsol)

Penandaan sumber daya yang ada dikelola pada halaman detail untuk sumber daya. Untuk informasi selengkapnya, lihat prosedur berikut:

- Lihat detail templat konektor
- Mengelola pendaftaran direktori

Untuk mengelola penandaan sumber daya (CLI)

Tag:tag-sumber dayaperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Daftar tag: list-tags-for-resourceperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Membatalkan tag:untag-sumber dayaperintah diAWS Private CAKonektor untuk bagian Direktori Aktif dariAWS CLI.

Untuk mengelola penandaan sumber daya (API)

Tag: TagResourceaksi diAWS Private CAKonektor untuk Active Directory API.

Daftar tag: ListTagsForResourceaksi diAWS Private CAKonektor untuk Active Directory API.

Membatalkan tag: UntagResourceaksi diAWS Private CAKonektor untuk Active Directory API.

Penting - Dapat diterima untuk menggunakan tag untuk melabeli objek yang berisi data rahasia. Namun, tag itu sendiri tidak boleh berisi informasi identitas pribadi (PII), informasi sensitif, atau rahasia.

AWS Private CA Konektor untuk Protokol Pendaftaran Sertifikat Sederhana (SCEP) (Pratinjau)

Konektor untuk SCEP dalam rilis pratinjau untuk AWS Private CA dan dapat berubah sewaktu-waktu.

Apa itu Konektor untuk SCEP?

Konektor untuk Simple Certificate Enrollment Protocol (SCEP) terhubung AWS Private Certificate Authority ke perangkat seluler dan peralatan jaringan berkemampuan SCEP Anda. Dengan Connector for SCEP, Anda dapat menggunakan AWS Private CA untuk menerbitkan sertifikat dan mendaftarkan perangkat SCEP Anda. Konektor untuk SCEP tersedia untuk digunakan dengan sistem manajemen perangkat seluler (MDM) populer dan dirancang untuk bekerja dengan klien atau titik akhir yang mendukung SCEP.

Topik

Fitur Konektor untuk SCEP

Support for SCEP protocol - SCEP adalah protokol yang diadopsi secara luas untuk mendapatkan sertifikat identitas digital dari otoritas sertifikat (CA) dan mendistribusikannya ke perangkat seluler dan perlengkapan jaringan. Anda dapat menggunakan Connector for SCEP untuk membantu Anda mendaftarkan endpoint Anda menggunakan SCEP.

Pendaftaran perangkat seluler - Anda dapat menggunakan Konektor untuk SCEP dengan sistem MDM populer termasuk Microsoft Intune dan Jamf Pro.

Menerbitkan sertifikat dalam skala besar - Setelah Anda mengonfigurasi perangkat berkemampuan SCEP untuk meminta sertifikat melalui titik akhir SCEP konektor, klien Anda dapat meminta sertifikat secara otomatis. AWS Private CA

Cara memulai dengan Konektor untuk SCEP

Untuk memulai, luncurkan panduan panduan dari Konektor untuk konsol manajemen SCEP yang membantu Anda membuat konektor dan menunjuk CA pribadi untuk digunakan dengan konektor. Setelah menyelesaikan langkah-langkah ini, Connector for SCEP menyediakan endpoint dan parameter konfigurasi lainnya yang dapat Anda masukkan ke dalam sistem MDM atau peralatan jaringan Anda. Setelah mengkonfigurasi sistem MDM atau peralatan jaringan Anda, klien Anda akan secara otomatis meminta sertifikat dari. AWS Private CA Untuk mempelajari lebih lanjut tentang cara memulai dengan Connector for SCEP, lihatMemulai dengan AWS Private Certificate Authority Konektor untuk SCEP.

Layanan terkait

Konektor untuk SCEP terkait dengan AWS layanan berikut.

- AWS Private Certificate Authority- AWS Private CA memberi Anda layanan CA pribadi yang sangat tersedia tanpa investasi di muka dan biaya pemeliharaan berkelanjutan untuk mengoperasikan CA pribadi Anda sendiri.
- AWS Private CA Konektor untuk Direktori Aktif - Konektor untuk AD menautkan Active Directory (AD) Anda ke AWS Private CA. Konektor menengahi pertukaran sertifikat dari AWS Private CA pengguna dan mesin yang dikelola oleh AD Anda.

Mengakses Konektor untuk SCEP

Anda dapat membuat, mengakses, dan mengelola Konektor untuk konektor SCEP menggunakan salah satu antarmuka berikut:

- AWS Management Console- Menyediakan antarmuka web yang dapat Anda gunakan untuk mengakses Konektor untuk SCEP. Lihat Konektor untuk konsol manajemen SCEP.
- AWS Command Line Interface- Menyediakan perintah untuk serangkaian AWS layanan yang luas, termasuk Konektor untuk SCEP. AWS CLI Ini didukung di Windows, macOS, dan Linux. Untuk informasi selengkapnya, lihat AWS Command Line Interface.

- AWS SDK - Menyediakan API khusus bahasa dan mengurus banyak detail koneksi, seperti menghitung tanda tangan, menangani percobaan ulang permintaan, dan penanganan kesalahan. Untuk informasi selengkapnya, lihat AWS Command Line Interface.
- Connector for SCEP API - Menyediakan tindakan API tingkat rendah yang Anda panggil menggunakan permintaan HTTPS. Menggunakan Connector for SCEP API adalah cara paling langsung untuk mengakses layanan. Namun, Connector for SCEP API mengharuskan aplikasi Anda menangani detail tingkat rendah seperti membuat hash untuk menandatangani permintaan, dan penanganan kesalahan. Untuk informasi selengkapnya, lihat Konektor untuk referensi API SCEP.

Harga untuk Konektor untuk SCEP

Konektor untuk SCEP ditawarkan sebagai fitur tanpa AWS Private CA biaya tambahan. Anda hanya membayar untuk AWS Private Certificate Authority operasi dan sertifikat yang digunakan untuk membuat dan memperbarui konektor.

Untuk informasi AWS Private CA harga terbaru, lihat AWS Private Certificate Authority Harga. Anda juga dapat menggunakan kalkulator AWS harga untuk memperkirakan biaya.

Konektor untuk konsep SCEP

> Konektor untuk SCEP dalam rilis pratinjau untuk AWS Private CA dan dapat berubah sewaktu-waktu.

Konektor untuk SCEP adalah fitur add-on untuk. AWS Private Certificate Authority

Berikut ini adalah konsep kunci untuk Konektor untuk SCEP:

Permintaan Penandatanganan Sertifikat (CSR)

: Informasi yang diperlukan diberikan kepada CA untuk memiliki sertifikat digital yang dikeluarkan. Informasi ini berisi kunci publik serta identitas.

Tantang kata sandi

: Protokol SCEP menggunakan kata sandi tantangan untuk mengautentikasi permintaan sebelum mengeluarkan sertifikat dari CA. Konektor untuk SCEP menangani kata sandi tantangan SCEP

berdasarkan jenis konektor. Untuk informasi selengkapnya, lihat Bagaimana Konektor untuk SCEP bekerja.

Pencabutan sertifikat

Pencabutan sertifikat adalah proses pencabutan sertifikat yang dikeluarkan sebelum tanggal kedaluwarsa. Anda dapat mencabut sertifikat CA pribadi yang terkait dengan konektor dengan memanggil RevokeCertificateAPI, AWS SDK, AWS Command Line Interface atau. AWS CloudFormation

Konektor untuk SCEP

Konektor untuk tautan SCEP AWS Private CA ke perangkat berkemampuan SCEP Anda.

Manajemen Perangkat Seluler

Manajemen Perangkat Seluler (MDM) memungkinkan administrator TI untuk mengontrol, mengamankan, dan menegakkan kebijakan pada ponsel cerdas, tablet, dan titik akhir atau perangkat lainnya. Banyak sistem MDM menyediakan integrasi bawaan untuk pendaftaran sertifikat berbasis SCEP.

SCEP

SCEP adalah protokol standar (RFC 8894) untuk mendistribusikan sertifikat secara otomatis. Protokol menyediakan titik akhir bagi perangkat untuk meminta sertifikat dari CA. SCEP menggunakan kata sandi tantangan untuk mengotorisasi penerbitan sertifikat ke perangkat. SCEP umumnya diterapkan untuk sistem manajemen perangkat seluler (MDM) dan peralatan jaringan. Solusi MDM memungkinkan administrator TI untuk mengontrol, mengamankan, dan menegakkan kebijakan pada ponsel cerdas, tablet, dan entitas lain seperti workstation Apple. Sebagian besar solusi MDM mendukung SCEP, seperti Microsoft Intune, Apple MDM, dan Jamf Pro. Sebagian besar peralatan jaringan, seperti router, penyeimbang beban, hub Wi-Fi, perangkat VPN, dan firewall, menggunakan SCEP untuk pendaftaran sertifikat otomatis.

Profil SCEP

Profil SCEP berisi parameter konfigurasi yang digunakan untuk menentukan profil sertifikat. Ini termasuk periode validitas sertifikat, ukuran kunci, nama konfigurasi SCEP, kata sandi tantangan, jumlah percobaan ulang yang gagal dan interval percobaan ulang, dan informasi lain yang relevan dengan penerbitan sertifikat. Sistem MDM dan platform manajemen sertifikat biasanya mengirim profil SCEP ke klien yang akan meminta sertifikat untuk otentikasi.

Bagaimana Konektor untuk SCEP bekerja

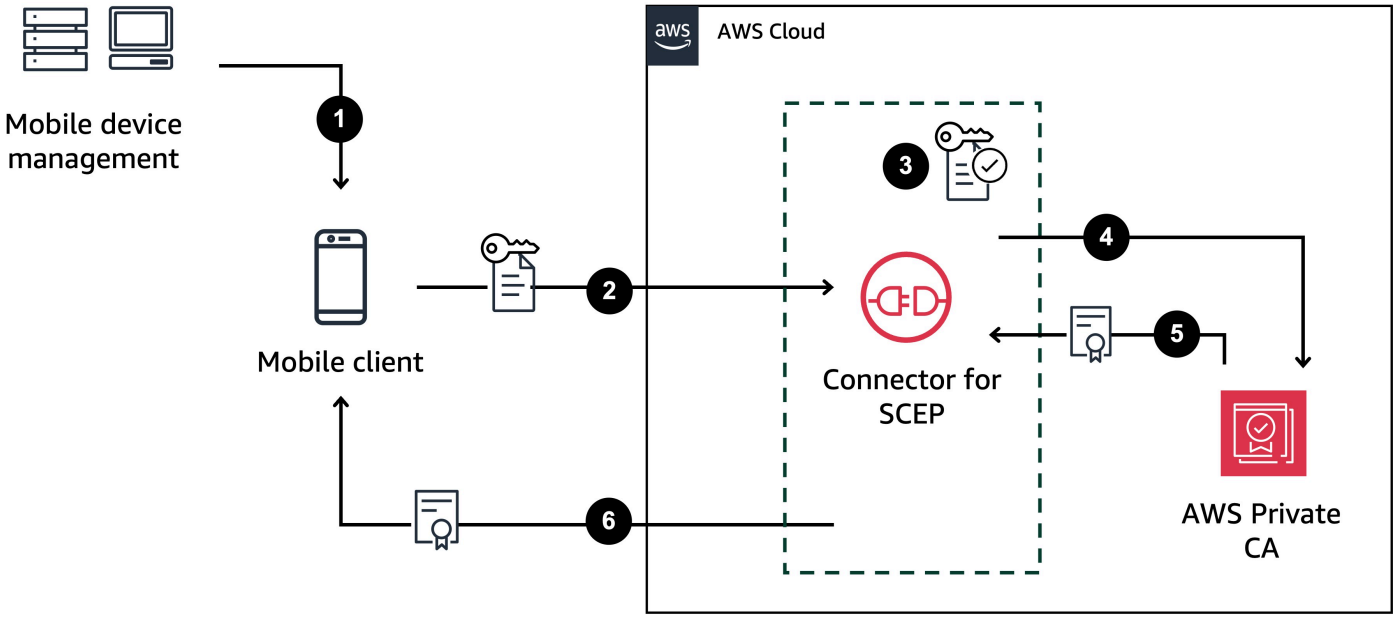
Konektor untuk SCEP dalam rilis pratinjau untuk AWS Private CA dan dapat berubah sewaktu-waktu.

Simple Certificate Enrollment Protocol (SCEP) adalah protokol standar yang digunakan untuk pendaftaran dan pembaruan sertifikat. Konektor untuk SCEP adalah server SCEP berbasis RFC 8894 yang secara otomatis mengeluarkan sertifikat dari klien SCEP AWS Private Certificate Authority Anda. Saat Anda membuat konektor, Connector for SCEP menyediakan titik akhir HTTPS bagi klien SCEP untuk meminta sertifikat. Klien mengautentikasi menggunakan kata sandi tantangan yang disertakan sebagai bagian dari permintaan penandatanganan sertifikat (CSR) mereka ke layanan. Anda dapat menggunakan Connector for SCEP dengan solusi manajemen perangkat seluler (MDM) populer termasuk Microsoft Intune dan Jamf Pro untuk mendaftarkan perangkat seluler. Ia bekerja dengan klien atau titik akhir yang mendukung SCEP.

Konektor untuk SCEP menawarkan dua jenis konektor—tujuan umum dan Konektor untuk SCEP untuk Microsoft Intune. Bagian berikut menjelaskan cara kerjanya.

Tujuan umum

Konektor tujuan umum dirancang untuk bekerja dengan titik akhir yang mendukung SCEP — kecuali Microsoft Intune — di mana kami memiliki konektor khusus. Dengan konektor tujuan umum, Anda mengelola kata sandi tantangan SCEP. Diagram berikut menggunakan sistem manajemen perangkat seluler (MDM) sebagai contoh, tetapi fungsionalitas yang sama berlaku jika Anda menggunakan sistem atau perangkat berkemampuan SCEP analog.

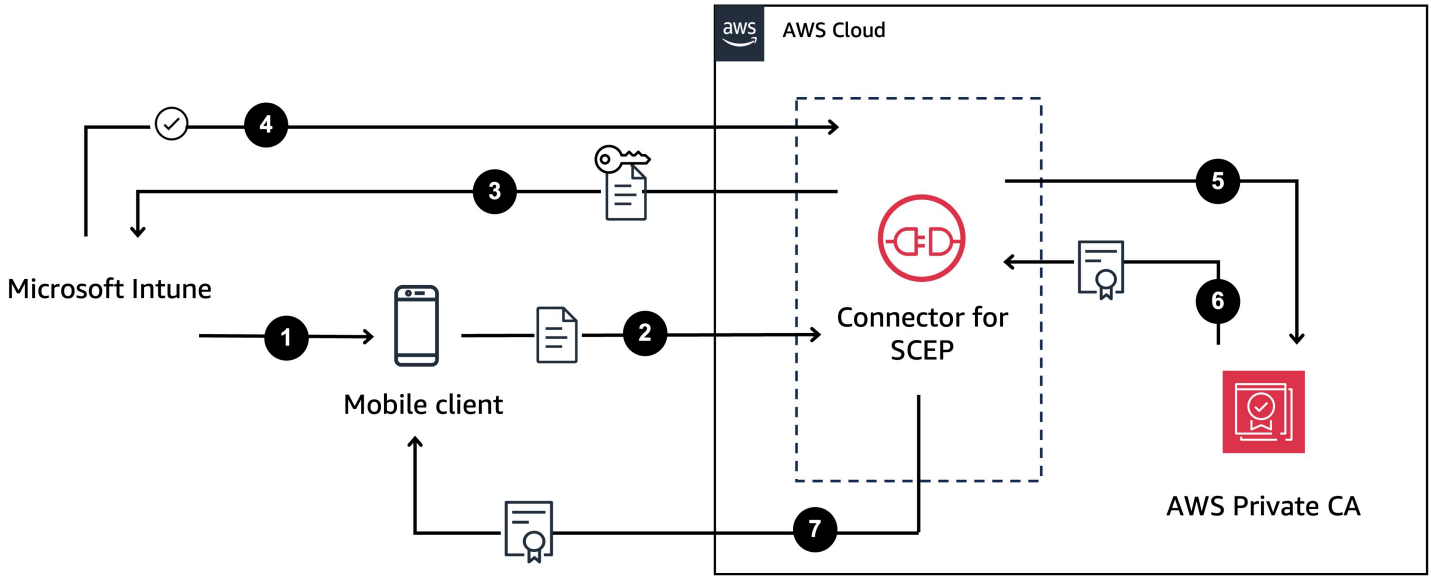


1. Sistem MDM (atau perangkat atau sistem analog) mengirimkan profil SCEP ke klien seluler. Profil SCEP berisi parameter konfigurasi yang digunakan untuk menentukan profil sertifikat, termasuk periode validitas sertifikat, ukuran kunci, nama konfigurasi SCEP, kata sandi tantangan, jumlah upaya yang gagal dan interval coba lagi, dan informasi lain yang relevan dengan penerbitan sertifikat.
2. Klien seluler meminta sertifikat dan juga mengirimkan permintaan penandatanganan sertifikat (CSR) yang menyertakan kata sandi tantangan.
3. Konektor untuk SCEP memvalidasi kata sandi tantangan. Jika valid, maka layanan meminta sertifikat dari AWS Private CA atas nama klien seluler.
4. AWS Private CA mengeluarkan sertifikat dan mengirimkannya ke Konektor untuk SCEP.
5. Konektor untuk SCEP mengirimkan sertifikat yang dikeluarkan ke klien seluler.

AWS Private Certificate Authority Konektor untuk SCEP untuk Microsoft Intune

AWS Private CA Konektor untuk SCEP untuk Microsoft Intune dirancang untuk digunakan dengan Microsoft Intune. Dengan jenis konektor Connector for SCEP for Microsoft Intune, Anda akan menggunakan Microsoft Intune untuk mengelola kata sandi tantangan SCEP Anda. Untuk informasi selengkapnya tentang penggunaan Connector for SCEP dengan Microsoft Intune, lihat. [Menggunakan Konektor untuk SCEP untuk Microsoft Intune]

Saat Anda menggunakan Connector for SCEP dengan Microsoft Intune, fungsionalitas tertentu diaktifkan dengan mengakses Microsoft Intune melalui Microsoft API. Penggunaan Konektor untuk SCEP dan AWS layanan yang menyertainya tidak menghilangkan kebutuhan Anda untuk memiliki lisensi yang valid untuk penggunaan layanan Microsoft Intune. Anda juga harus meninjau Kebijakan Perlindungan Aplikasi Microsoft Intune®.



1. Microsoft Intune mengirimkan profil SCEP ke klien seluler. Profil berisi kata sandi tantangan terenkripsi yang ditempatkan klien seluler ke dalam CSR.
2. Klien seluler meminta sertifikat dan mengirimkan CSR ke Connector untuk SCEP.
3. Konektor untuk SCEP mengirimkan CSR ke Microsoft Intune untuk otorisasi.
4. Microsoft Intune mendekripsi kata sandi tantangan di CSR. Jika valid, Microsoft Intune mengirimkan persetujuan ke Connector untuk SCEP untuk menerbitkan sertifikat ke klien seluler.
5. Konektor untuk SCEP meminta sertifikat dari AWS Private CA atas nama klien seluler.
6. AWS Private CA mengeluarkan sertifikat dan mengirimkannya ke Konektor untuk SCEP.
7. Konektor untuk SCEP mengirimkan sertifikat yang dikeluarkan ke klien seluler.

Pertimbangan dan batasan saat bekerja dengan Konektor untuk SCEP

Pertimbangan

Mode operasi CA

Anda hanya dapat menggunakan Konektor untuk SCEP dengan CA pribadi yang menggunakan mode operasi tujuan umum. Konektor untuk SCEP default untuk menerbitkan sertifikat dengan masa berlaku satu tahun. CA pribadi yang menggunakan mode sertifikat berumur pendek tidak mendukung penerbitan sertifikat dengan masa berlaku lebih dari tujuh hari. Untuk informasi tentang mode operasi, lihatMode otoritas sertifikat.

Tantang kata sandi

- Bagikan kata sandi tantangan Anda dengan sangat hati-hati dan bagikan hanya dengan individu dan klien yang sangat tepercaya. Kata sandi tantangan tunggal dapat digunakan untuk mengeluarkan sertifikat apa pun, dengan subjek dan SAN apa pun, yang menimbulkan risiko keamanan.
- Jika menggunakan konektor tujuan umum, kami sarankan Anda sering memutar kata sandi tantangan secara manual.

Kesesuaian dengan RFC 8894

Konektor untuk SCEP menyimpang dari protokol RFC 8894 dengan menyediakan titik akhir HTTPS alih-alih titik akhir HTTP.

CSR

- Jika permintaan penandatanganan sertifikat (CSR) yang dikirim ke Connector for SCEP tidak menyertakan ekstensi Extended Key Usage (EKU), kami akan menetapkan nilai EKU. `clientAuthentication` Untuk informasi, lihat 4.2.1.12. Penggunaan Kunci yang Diperpanjang di RFC 5280.
- Kami mendukung `ValidityPeriod` dan atribut `ValidityPeriodUnits` khusus di CSR. Jika CSR Anda tidak menyertakan a`ValidityPeriod`, kami menerbitkan sertifikat yang memiliki masa berlaku satu tahun. Perlu diingat bahwa Anda mungkin tidak dapat mengatur atribut ini dalam sistem MDM Anda. Tetapi jika Anda dapat mengaturnya, kami mendukung mereka. Untuk informasi tentang atribut ini, lihat SzenRollment_NAME_VALUE_PAIR.

Berbagi titik akhir

Mendistribusikan titik akhir konektor hanya kepada pihak tepercaya. Perlakukan titik akhir sebagai rahasia karena siapa pun yang dapat menemukan nama domain dan jalur unik Anda yang memenuhi syarat dapat mengambil sertifikat CA Anda.

Batasan

Batasan berikut berlaku untuk Konektor untuk SCEP.

Kata sandi tantangan dinamis

Anda hanya dapat membuat kata sandi tantangan statis dengan konektor tujuan umum. Untuk menggunakan kata sandi dinamis dengan konektor tujuan umum, Anda harus membangun mekanisme rotasi Anda sendiri yang menggunakan kata sandi statis konektor. Konektor untuk SCEP untuk jenis konektor Microsoft Intune menawarkan dukungan untuk kata sandi dinamis, yang Anda kelola menggunakan Microsoft Intune.

HTTP

Konektor untuk SCEP hanya mendukung HTTPS, dan membuat pengalihan untuk panggilan HTTP. Jika sistem Anda bergantung pada HTTP, pastikan bahwa itu dapat mengakomodasi pengalihan HTTP yang disediakan oleh Connector for SCEP.

CA pribadi bersama

Anda hanya dapat menggunakan Konektor untuk AD dengan CA pribadi di mana Anda adalah pemiliknya.

Menyiapkan Konektor untuk SCEP

> Konektor untuk SCEP dalam rilis pratinjau untuk AWS Private CA dan dapat berubah sewaktu-waktu.

Prosedur di bagian ini membantu Anda memulai dengan Connector for SCEP. Ini mengasumsikan bahwa Anda telah membuat AWS akun. Setelah Anda menyelesaikan langkah-langkah di halaman ini, Anda dapat melanjutkan dengan membuat konektor untuk SCEP.

Topik

- Langkah 1: Buat AWS Identity and Access Management kebijakan
- Langkah 2: Buat CA pribadi
- Langkah 3: Buat berbagi sumber daya menggunakan AWS Resource Access Manager

Langkah 1: Buat AWS Identity and Access Management kebijakan

Untuk membuat konektor untuk SCEP, Anda perlu membuat kebijakan IAM yang memberi Konektor untuk SCEP kemampuan untuk membuat dan mengelola sumber daya yang dibutuhkan oleh konektor, dan menerbitkan sertifikat atas nama Anda. Untuk informasi lebih lanjut tentang IAM lihat Apa itu IAM? di Panduan Pengguna IAM.

Contoh berikut adalah kebijakan yang dikelola pelanggan yang dapat Anda gunakan untuk Connector for SCEP.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "pca-connector-scep:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListCertificateAuthorities",
                "acm-pca:ListTags",
                "acm-pca:PutPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "acm-pca:IssueCertificate",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APICSRPassthrough/V*"
                },
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "pca-connector-scep.amazonaws.com"
                }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:CreateResourceShare",
                "ram:GetResourcePolicies",
                "ram:GetResourceShareAssociations",
                "ram:GetResourceShares",
                "ram:ListPrincipals",
                "ram:ListResources",
                "ram:ListResourceSharePermissions",
                "ram:ListResourceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

Langkah 2: Buat CA pribadi

Untuk menggunakan Konektor untuk SCEP, Anda perlu mengaitkan CA pribadi dari AWS Private Certificate Authority ke konektor. Kami menyarankan Anda menggunakan CA pribadi yang hanya untuk konektor, karena kerentanan keamanan yang melekat yang ada dalam protokol SCEP.

CA pribadi harus memenuhi persyaratan berikut:

- Itu harus dalam keadaan aktif dan menggunakan mode operasi tujuan umum.
- Anda harus memiliki CA pribadi. Anda tidak dapat menggunakan CA pribadi yang dibagikan dengan Anda melalui berbagi lintas akun.

Perhatikan pertimbangan berikut saat mengonfigurasi CA pribadi Anda untuk digunakan dengan Connector for SCEP:

- Kendala nama DNS — Pertimbangkan untuk menggunakan batasan nama DNS sebagai cara untuk mengontrol domain mana yang diizinkan atau dilarang dalam sertifikat yang dikeluarkan untuk perangkat SCEP Anda. Untuk informasi selengkapnya, lihat Cara menerapkan batasan nama DNS di. AWS Private Certificate Authority
- Pencabutan - Aktifkan OCSP atau CRL pada CA pribadi Anda untuk memungkinkan pencabutan. Untuk informasi selengkapnya, lihat Menyiapkan metode pencabutan sertifikat.

- PII — Kami menyarankan agar Anda tidak menambahkan informasi identitas pribadi (PII) atau informasi rahasia atau sensitif lainnya dalam sertifikat CA Anda. Jika terjadi eksploitasi keamanan, ini membantu membatasi paparan informasi sensitif.
- Simpan sertifikat root di toko kepercayaan — Simpan sertifikat CA root Anda di toko kepercayaan perangkat Anda, sehingga Anda dapat memverifikasi sertifikat dan nilai pengembalian GetCertificateAuthorityCertificate. Untuk informasi tentang toko kepercayaan yang terkait dengannya AWS Private CA, lihatCA akar .

Untuk informasi tentang cara membuat CA pribadi, lihatMembuat CA pribadi.

Langkah 3: Buat berbagi sumber daya menggunakan AWS Resource Access Manager

Jika Anda menggunakan Connector for SCEP secara terprogram menggunakan AWS Command Line Interface, AWS SDK, atau Connector for SCEP API, Anda perlu membagikan CA pribadi Anda dengan Connector for SCEP dengan menggunakan berbagi prinsip layanan. AWS Resource Access Manager Ini memberikan Konektor untuk SCEP akses bersama ke CA pribadi Anda. Saat Anda membuat konektor di AWS konsol, kami secara otomatis membuat pembagian sumber daya untuk Anda. Untuk informasi tentang berbagi sumber daya, lihat Membuat pembagian sumber daya di Panduan AWS RAM Pengguna.

Untuk membuat berbagi sumber daya menggunakan AWS CLI, Anda dapat menggunakan AWS RAM create-resource-share perintah. Perintah berikut menciptakan berbagi sumber daya. *Tentukan ARN CA pribadi yang ingin Anda bagikan sebagai nilai resource-arns.*

```
$  aws ram create-resource-share \
--region us-east-1 \
--name MyPcaConnectorScepResourceShare \
--permission-arns arn:aws:ram::aws:permission/
AWSRAMBlankEndEntityCertificateAPICSRPassthroughIssuanceCertificateAuthority \
--resource-arns arn:aws:acm-pca:Region:account:certificate-authority/CA_ID \
--principals pca-connector-scep.amazonaws.com \
--sources account
```

Prinsipal layanan yang menelepon `CreateConnector` memiliki izin penerbitan sertifikat pada CA pribadi. Untuk mencegah prinsip layanan yang menggunakan Connector for SCEP memiliki akses umum ke AWS Private CA sumber daya Anda, batasi izin mereka menggunakan. `CalledVia`

Memulai dengan AWS Private Certificate Authority Konektor untuk SCEP

Konektor untuk SCEP dalam rilis pratinjau untuk AWS Private CA dan dapat berubah.

Dengan AWS Private Certificate Authority Connector for SCEP, Anda dapat mengeluarkan sertifikat dari CA pribadi Anda ke perangkat berkemampuan SCEP dan sistem manajemen perangkat seluler (MDM). Saat Anda membuat konektor, AWS Private Certificate Authority buat URL SCEP publik agar Anda dapat meminta sertifikat, dan juga memberi Anda informasi yang dapat Anda gunakan untuk diintegrasikan ke dalam sistem MDM Anda.

Untuk menerbitkan sertifikat, Anda harus membuat CA AWS Private Certificate Authority pribadi, membuat konektor, dan kemudian mengonfigurasi sistem dan perangkat MDM berkemampuan SCEP Anda untuk meminta sertifikat dari konektor.

Topik

Sebelum Anda mulai

Untuk mengikuti tutorial memandu Anda melalui proses membuat konektor untuk SCEP.

Untuk mengikuti tutorial ini, Anda memerlukan CA pribadi dan perangkat berkemampuan SCEP. Anda juga harus terlebih dahulu memenuhi prasyarat yang tercantum di bagian ini. Menyiapkan Konektor untuk SCEP

Prosedur berikut memandu Anda cara membuat konektor menggunakan AWS konsol.

Tugas

Langkah 1: Buat konektor

Anda akan membuat konektor untuk penggunaan tujuan umum atau Konektor untuk SCEP untuk Microsoft Intune. Konektor tujuan umum dirancang untuk digunakan dengan titik akhir berkemampuan SCEP, dan Anda mengelola kata sandi tantangan SCEP. Konektor untuk SCEP untuk Microsoft Intune adalah untuk digunakan dengan Microsoft Intune, dan Anda mengelola password tantangan menggunakan Microsoft Intune.

General-purpose

Untuk membuat konektor untuk penggunaan tujuan umum

Masuk ke AWS akun Anda dan buka Konektor untuk konsol SCEP di**https://console.aws.amazon.com/pca-connector-scep/home**.

1. Pilih Buat konektor.
2. Di halaman Buat konektor, secara opsional berikan konektor nama ramah di bidang tag Nama. Nama akan ditampilkan dalam daftar konektor Anda. Jika mau, Anda dapat menambahkan lebih banyak tag ke konektor dengan memilih Tambahkan lebih banyak tag. Tag adalah label yang Anda tetapkan ke AWS sumber daya. Setiap tanda terdiri dari kunci dan nilai opsional. Anda dapat menggunakan tag untuk mencari dan memfilter sumber daya Anda atau melacak AWS biaya Anda.
3. Di bawah Jenis konektor, pilih Tujuan umum.
4. Di bawah Private CA, pilih CA pribadi untuk digunakan dengan konektor ini. Atau, buat yang baru dengan memilih Buat CA pribadi. Karena kerentanan yang melekat pada protokol SCEP, sebaiknya gunakan CA pribadi yang didedikasikan untuk konektor ini. Jika Anda membuat CA baru, setelah Anda selesai membuatnya AWS Private CA, kembali ke Konektor untuk konsol SCEP dan segarkan daftar CA pribadi. CA pribadi baru Anda harus tersedia untuk dipilih.
5. Di bawah Kata sandi tantangan pilih Secara otomatis menghasilkan kata sandi tantangan. Kami akan menghasilkan kata sandi tantangan statis untuk Anda saat kami membuat konektor ini.
6. Pilih Buat konektor.

Microsoft Intune

Untuk membuat Konektor untuk SCEP untuk Microsoft Intune

Masuk ke AWS akun Anda dan buka Konektor untuk konsol SCEP di**https://console.aws.amazon.com/pca-connector-scep/home**.

1. Pilih Buat konektor.
2. Pada halaman Buat konektor, secara opsional berikan konektor nama ramah di bidang tag Nama. Nama akan ditampilkan dalam daftar konektor Anda. Jika mau, Anda dapat menambahkan lebih banyak tag ke konektor dengan memilih Tambahkan lebih banyak tag. Tag adalah label yang Anda tetapkan ke AWS sumber daya. Setiap tanda terdiri dari kunci dan nilai opsional. Anda dapat menggunakan tag untuk mencari dan memfilter sumber daya Anda atau melacak AWS biaya Anda.
3. Di bawah Jenis konektor, pilih Microsoft Intune.
   a. Untuk ID Aplikasi (klien), masukkan ID aplikasi (klien) dari pendaftaran aplikasi Microsoft Entra ID Anda. Untuk informasi tentang menggunakan Microsoft Intune dengan Konektor untuk SCEP, lihat. Menggunakan Konektor untuk SCEP dengan sistem MDM
   b. Untuk ID Direktori (penyewa) atau domain utama, masukkan ID direktori (penyewa) atau domain utama dari pendaftaran aplikasi Microsoft Entra ID Anda.
4. Di bawah Private CA, pilih CA pribadi untuk digunakan dengan konektor ini. Atau, buat yang baru dengan memilih Buat CA pribadi. Karena kerentanan yang melekat pada protokol SCEP, sebaiknya gunakan CA pribadi yang didedikasikan untuk konektor ini. Jika Anda membuat CA baru, setelah Anda selesai membuatnya AWS Private CA, kembali ke Konektor untuk konsol SCEP dan segarkan daftar CA pribadi. CA pribadi baru Anda harus tersedia untuk dipilih.
5. Pilih Buat konektor.

Langkah 2: Salin detail konektor ke sistem MDM Anda

Setelah membuat konektor, Anda harus menyalin detail berikut dari konektor ke sistem MDM Anda. Untuk melihat detail konektor menggunakan konsol, pilih konektor dari daftar di halaman Konektor untuk konsol SCEP.

- URL SCEP Publik - Ini adalah titik akhir konektor tempat klien SCEP Anda akan meminta sertifikat. Berhati-hatilah untuk hanya menyediakan titik akhir ini ke entitas tepercaya.

- (Tujuan umum) Kata sandi tantangan - Di bawah kata sandi Tantang, pilih kata sandi yang Anda buat secara otomatis dalam prosedur sebelumnya dan kemudian pilih Lihat kata sandi untuk melihat kata sandi. Untuk membuat kata sandi tambahan, pilih Buat kata sandi. Berhati-hatilah untuk mendistribusikan kata sandi dengan hati-hati dan hanya kepada individu dan klien yang sangat tepercaya. Kata sandi tantangan tunggal dapat digunakan untuk mengeluarkan sertifikat apa pun, dengan subjek dan SAN apa pun, dan karenanya harus ditangani dengan hati-hati.
- (Microsoft Intune) Buka nilai ID - Jika Anda berintegrasi dengan Microsoft Intune, Anda harus menyalin penerbit Open ID, subjek Open ID, dan Open ID ke kredensi OpenID Connect (OIDC) pendaftaran aplikasi Microsoft Entra Anda. Untuk informasi selengkapnya, lihat Menggunakan Konektor untuk SCEP dengan sistem MDM.

Menggunakan Konektor untuk SCEP dengan sistem MDM

Konektor untuk SCEP dalam rilis pratinjau untuk AWS Private Certificate Authority dan dapat berubah sewaktu-waktu.

Bagian berikut menjelaskan cara menggunakan Konektor untuk SCEP dengan sistem manajemen perangkat seluler (MDM) tertentu.

Topik

- Menggunakan Konektor untuk SCEP dengan Jamf Pro
- Menggunakan Konektor untuk SCEP untuk Microsoft Intune

Menggunakan Konektor untuk SCEP dengan Jamf Pro

Anda dapat menggunakan AWS Private CA sebagai otoritas sertifikat eksternal (CA) dengan solusi manajemen perangkat seluler Jamf Pro (MDM). Panduan ini memberikan petunjuk tentang cara mengkonfigurasi Jamf Pro sebagai Proxy SCEP setelah membuat AWS Private Certificate Authority Konektor untuk SCEP.

Persyaratan Jamf Pro

Implementasi Jamf Pro Anda harus memenuhi persyaratan berikut.

- Anda harus menggunakan Jamf Pro 10.0.0 atau yang lebih baru.

- Anda harus mengaktifkan pengaturan Aktifkan otentikasi berbasis sertifikat di Jamf Pro. Anda dapat menemukan detail tentang pengaturan ini di halaman Pengaturan Keamanan Jamf Pro di dokumentasi Jamf Pro.

Prasyarat

Untuk menggunakan Konektor untuk SCEP dengan Jamf Pro, Anda harus terlebih dahulu membuat CA pribadi dan konektor tujuan umum untuk SCEP. Untuk petunjuk, lihat Menyiapkan Konektor untuk SCEP.

Konfigurasikan AWS Private CA sebagai CA eksternal di Jamf Pro

Setelah Anda membuat konektor untuk SCEP, Anda harus mengatur AWS Private CA sebagai CA eksternal di Jamf Pro. Anda dapat mengatur AWS Private CA sebagai CA eksternal global. Atau Anda dapat menggunakan profil konfigurasi Jamf Pro untuk mengeluarkan sertifikat yang berbeda dari AWS Private CA untuk kasus penggunaan yang berbeda, seperti menerbitkan sertifikat ke subset perangkat di organisasi Anda. Panduan penerapan profil konfigurasi Jamf Pro berada di luar cakupan dokumen ini.

Untuk mengkonfigurasi AWS Private CA sebagai CA eksternal di Jamf Pro

1. Di konsol Jamf Pro, buka halaman pengaturan sertifikat PKI dengan masuk ke Pengaturan > Global > sertifikat PKI.
2. Pilih tab Templat Sertifikat Manajemen.
3. Pilih CA Eksternal.
4. Pilih Edit.
5. (Opsional) Pilih Aktifkan Jamf Pro sebagai SCEP Proxy untuk profil konfigurasi. Anda dapat menggunakan profil konfigurasi Jamf Pro untuk mengeluarkan sertifikat berbeda yang disesuaikan dengan kasus penggunaan tertentu. Untuk panduan tentang cara menggunakan profil konfigurasi di Jamf Pro, lihat Mengaktifkan Jamf Pro sebagai Proxy SCEP untuk Profil Konfigurasi dalam dokumentasi Jamf Pro.
6. Pilih Gunakan CA eksternal berkemampuan SCEP untuk pendaftaran komputer dan perangkat seluler.
7. (Opsional) Pilih Gunakan Jamf Pro sebagai SCEP Proxy untuk pendaftaran komputer dan perangkat seluler. Jika Anda mengalami kegagalan instalasi profil, lihatMemecahkan masalah kegagalan instalasi profil.

8. Salin dan tempel Konektor untuk URL SCEP publik SCEP dari detail konektor ke bidang URL di Jamf Pro. Untuk melihat detail konektor, pilih konektor dari daftar Konektor untuk SCEP. Atau, Anda bisa mendapatkan URL dengan memanggil GetConnectordan menyalin `Endpoint` nilai dari respons.
9. (Opsional) Masukkan nama instance di bidang Nama. Misalnya, Anda bisa menamainya AWS Private CA.
10. Pilih Statis untuk jenis tantangan.
11. Salin kata sandi tantangan konektor Anda dan tempelkan ke bidang Tantangan. Untuk melihat kata sandi tantangan konektor Anda, navigasikan ke halaman detail konektor Anda di AWS konsol dan pilih tombol Lihat kata sandi. Atau, Anda bisa mendapatkan kata sandi tantangan konektor dengan memanggil GetChallengeKata Sandi dan menyalin `Password` nilai dari respons.
12. Tempelkan kata sandi tantangan ke dalam bidang Verifikasi Tantangan.
13. Pilih Ukuran Kunci. Kami merekomendasikan ukuran kunci 2048 atau lebih tinggi.
14. (Opsional) Pilih Gunakan sebagai tanda tangan digital. Pilih ini untuk tujuan otentikasi untuk memberikan perangkat akses aman ke sumber daya seperti Wi-Fi dan VPN.
15. (Opsional) Pilih Gunakan untuk encipherment kunci.
16. (Opsional) Masukkan string hex di bidang Sidik Jari. Untuk petunjuk tentang cara membuat sidik jari CA pribadi Anda, lihat(Opsional) Tambahkan sidik jari CA.
17. Pilih Simpan.

Membuat dan mengunggah sertifikat penandatanganan profil

Untuk menggunakan Konektor untuk SCEP dengan Jamf Pro, Anda harus memberikan sertifikat penandatanganan dan CA untuk CA pribadi yang terkait dengan konektor Anda. Anda dapat melakukannya dengan mengunggah keystore sertifikat penandatanganan profil ke Jamf Pro yang berisi kedua sertifikat tersebut. Anda perlu membuat permintaan penandatanganan sertifikat (CSR) menggunakan proses internal Anda dan membuatnya ditandatangani AWS Private Certificate Authority. Petunjuk berikut menjelaskan cara membuat keystore sertifikat dan mengunggahnya ke Jamf Pro. Contoh berikut menggunakan OpenSSL, tetapi Anda dapat membuat permintaan penandatanganan sertifikat menggunakan metode pilihan Anda.

1. Menggunakan OpenSSL, buat kunci pribadi dengan menjalankan perintah berikut:

```
openssl genrsa -out local.key 2048
```

2. Buat permintaan penandatanganan sertifikat (CSR):

```
openssl req -new -key local.key -sha512 -out local.csr -
subj "/CN=MySigningCertificate/O=MyOrganization" -addext
 keyUsage=critical,digitalSignature,nonRepudiation
```

3. Dengan menggunakan AWS CLI, terbitkan sertifikat penandatanganan menggunakan CSR yang Anda buat di langkah kedua. Jalankan perintah berikut dan catat sertifikat ARN dalam tanggapan:

```
aws acm-pca issue-certificate --certificate-authority-arn <SAME CA AS USED ABOVE,
 SO IT'S TRUSTED> --csr fileb://local.csr --signing-algorithm SHA512WITHRSA --
validity Value=365,Type=DAYS
```

4. Dapatkan sertifikat penandatanganan dengan menjalankan perintah berikut menggunakan sertifikat ARN dari langkah 3:

```
aws acm-pca get-certificate --certificate-authority-arn <SAME CA AS USED ABOVE, SO
 IT'S TRUSTED> --certificate-arn <ARN OF NEW CERTIFICATE> | jq -r '.Certificate'
 >local.crt
```

5. Dapatkan sertifikat CA dengan menjalankan perintah berikut:

```
aws acm-pca get-certificate-authority-certificate --certificate-authority-arn <SAME
 CA AS USED ABOVE, SO IT'S TRUSTED> | jq -r '.Certificate' > ca.crt
```

6. Menggunakan OpenSSL, keluarkan keystore sertifikat penandatanganan dalam format p12. Anda akan menggunakan `crt` file yang Anda buat selama langkah empat dan lima. Jalankan perintah berikut:

```
openssl pkcs12 -export -in local.crt -inkey local.key -certfile ca.crt -name "CA
 Chain" -out local.p12
```

7. Saat diminta, masukkan kata sandi ekspor. Kata sandi ini adalah kata sandi keystore Anda, dan Anda harus menggunakannya nanti.
8. Di Jamf Pro, arahkan ke Template Sertifikat Manajemen dan buka panel CA Eksternal.
9. Di bagian bawah panel CA Eksternal, pilih Ubah Penandatanganan dan Sertifikat CA.
10. Ikuti petunjuk di layar untuk mengunggah sertifikat penandatanganan dan CA untuk CA eksternal.

(Opsional) Tambahkan sidik jari CA

Menambahkan sidik jari CA memungkinkan perangkat yang dikelola untuk memverifikasi CA dan hanya meminta sertifikat dari CA.

1. Dapatkan sertifikat CA pribadi baik dari AWS Private CA konsol atau dengan menggunakan GetCertificateAuthorityCertificate. Simpan sebagai `ca.pem` file.
2. Di OpenSSL, jalankan perintah berikut:

   ```
   openssl x509 -in ca.pem -sha256 -fingerprint
   ```

3. Salin dan tempel output ke bidang Sidik Jari sebagaimana dimaksud dalam prosedur sebelumnya.

(Opsional) Instal sertifikat selama pendaftaran yang dimulai pengguna

Untuk memasang sertifikat CA pribadi konektor Anda ke klien atau perangkat selama pendaftaran yang dimulai pengguna, konfigurasikan setelan pendaftaran yang dimulai pengguna Jamf Pro Anda. Ini membantu Jamf Pro untuk menginstal AWS Private CA sertifikat Anda ke klien atau perangkat saat mereka meminta sertifikat. Anda bertanggung jawab untuk menguji konfigurasi Anda untuk memastikan bahwa itu kompatibel dengan Konektor Anda untuk implementasi SCEP. Untuk informasi tentang pengaturan pendaftaran yang dimulai pengguna Jamf Pro, lihat Pengaturan Pendaftaran yang Dimulai Pengguna di dokumentasi Jamf Pro.

Memecahkan masalah kegagalan instalasi profil

Jika Anda mengalami kegagalan instalasi profil setelah mengaktifkan Gunakan Jamf Pro sebagai SCEP Proxy untuk pendaftaran komputer dan perangkat seluler, coba yang berikut ini.

Pesan kesalahan	Mitigasi
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:15001>`	Jika Anda menerima pesan galat ini saat mencoba mendaftar, coba lagi pendaftaran. Diperlukan beberapa kali percobaan sebelum pendaftaran berhasil.
`Profile installation failed. Unable to obtain certificate from`	Kata sandi tantangan Anda mungkin salah dikonfigurasi. Verifikasi bahwa kata sandi

Pesan kesalahan	Mitigasi
`SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:14006>`	tantangan di Jamf Pro cocok dengan kata sandi tantangan konektor Anda.

Menggunakan Konektor untuk SCEP untuk Microsoft Intune

Anda dapat menggunakan AWS Private CA sebagai otoritas sertifikat eksternal (CA) dengan sistem manajemen perangkat seluler Microsoft Intune (MDM). Panduan ini memberikan petunjuk tentang cara mengkonfigurasi Microsoft Intune setelah Anda membuat Konektor untuk SCEP untuk Microsoft Intune.

Prasyarat

Sebelum Anda membuat Konektor untuk SCEP untuk Microsoft Intune, Anda harus menyelesaikan prasyarat berikut.

- Buat ID Entra.
- Buat Penyewa Intune Microsoft.
- Buat Pendaftaran Aplikasi di ID Microsoft Entra Anda. Lihat Memperbarui izin yang diminta aplikasi di Microsoft Entra ID di dokumentasi Microsoft Entra untuk informasi tentang cara mengelola izin tingkat aplikasi untuk Pendaftaran Aplikasi Anda. Pendaftaran Aplikasi harus memiliki izin berikut:
  - Di bawah Intune setel scep_challenge_provider.
  - Untuk Microsoft Graph mengatur Application.Read.All dan User.Read.
- Anda harus memberikan aplikasi dalam persetujuan admin Pendaftaran Aplikasi Anda. Untuk selengkapnya, lihat Memberikan persetujuan admin seluruh penyewa untuk aplikasi dalam dokumentasi Microsoft Entra.

> **Tip**
>
> Saat Anda membuat Pendaftaran Aplikasi, catat ID Aplikasi (klien) dan ID Direktori (penyewa) atau domain utama. Saat Anda membuat Konektor untuk SCEP untuk Microsoft Intune, Anda akan memasukkan nilai-nilai ini. Untuk informasi tentang cara mendapatkan nilai ini, lihat Membuat aplikasi dan prinsip layanan Microsoft Entra yang dapat mengakses sumber daya dalam dokumentasi Microsoft Entra.

Memberikan AWS Private CA izin untuk menggunakan Aplikasi Microsoft Entra ID

Setelah Anda membuat Konektor untuk SCEP untuk Microsoft Intune, Anda harus membuat kredensi federasi di bawah Pendaftaran Aplikasi Microsoft sehingga Konektor untuk SCEP dapat berkomunikasi dengan Microsoft Intune.

Untuk mengkonfigurasi AWS Private CA sebagai CA eksternal di Microsoft Intune

1. Di konsol Microsoft Entra ID, navigasikan ke pendaftaran Aplikasi.
2. Pilih aplikasi yang Anda buat untuk digunakan dengan Connector for SCEP. ID aplikasi (klien) aplikasi yang Anda klik harus cocok dengan ID yang Anda tentukan saat Anda membuat konektor.
3. Pilih Sertifikat & rahasia dari menu tarik-turun Dikelola.
4. Pilih tab Kredensial Federasi.
5. Pilih Tambahkan kredensi.
6. Dari menu tarik-turun skenario kredensi Federasi, pilih Penerbit lain.
7. Salin dan tempel nilai penerbit OpenID dari detail Connector for SCEP for Microsoft Intune Anda ke dalam bidang Penerbit. Untuk melihat detail konektor, pilih konektor dari daftar Konektor untuk SCEP di AWS konsol. Atau, Anda bisa mendapatkan URL dengan menelepon GetConnectordan kemudian menyalin `Issuer` nilai dari respons.
8. Salin dan tempel nilai OpenID Audience dari detail Connector for SCEP for Microsoft Intune ke bidang Audience. Untuk melihat detail konektor, pilih konektor dari daftar Konektor untuk SCEP di AWS konsol. Atau, Anda bisa mendapatkan URL dengan menelepon GetConnectordan kemudian menyalin `Subject` nilai dari respons.
9. (Opsional) Masukkan nama instance di bidang Nama. Misalnya, Anda bisa menamainya AWS Private CA.
10. (Opsional) Masukkan deskripsi ke dalam bidang Deskripsi.
11. Pilih Edit (opsional) di bawah bidang Audiens. Salin dan tempel nilai subjek OpenID dari konektor Anda ke bidang Subjek. Anda dapat melihat nilai penerbit OpenID di halaman detail konektor di konsol. AWS Atau, Anda bisa mendapatkan URL dengan menelepon GetConnectordan kemudian menyalin `Audience` nilai dari respons.
12. Pilih Tambahkan.

Mengatur profil konfigurasi Microsoft Intune

Setelah Anda memberikan AWS Private CA izin untuk memanggil Microsoft Intune, Anda harus menggunakan Microsoft Intune untuk membuat profil konfigurasi Microsoft Intune yang menginstruksikan perangkat untuk menghubungi Connector for SCEP untuk penerbitan sertifikat.

1. Buat profil konfigurasi sertifikat tepercaya. Anda harus mengunggah sertifikat CA root dari rantai yang Anda gunakan dengan Connector for SCEP ke Microsoft Intune untuk membangun kepercayaan. Untuk informasi tentang cara membuat profil konfigurasi sertifikat tepercaya, lihat Profil sertifikat root tepercaya untuk Microsoft Intune di dokumentasi Microsoft Intune.
2. Buat profil konfigurasi sertifikat SCEP yang mengarahkan perangkat Anda ke konektor saat memerlukan sertifikat baru. Jenis Profil profil konfigurasi harus Sertifikat SCEP. Untuk sertifikat root profil konfigurasi, pastikan Anda menggunakan sertifikat tepercaya yang Anda buat pada langkah sebelumnya.

   Untuk URL Server SCEP, salin dan tempel URL SCEP Publik dari detail konektor Anda ke bidang URL Server SCEP. Untuk melihat detail konektor, pilih konektor dari daftar Konektor untuk SCEP. Atau, Anda bisa mendapatkan URL dengan menelepon GetConnector, dan kemudian menyalin `Endpoint` nilai dari respons. Untuk panduan cara membuat profil konfigurasi di Microsoft Intune, lihat Membuat dan menetapkan profil sertifikat SCEP di Microsoft Intune dalam dokumentasi Microsoft Intune.

   > **Note**
   >
   > Untuk perangkat non-mac OS dan iOS, jika Anda tidak menetapkan masa berlaku di profil konfigurasi, Connector for SCEP mengeluarkan sertifikat dengan validitas satu tahun. Jika Anda tidak menyetel nilai Extended Key Usage (EKU) di profil konfigurasi, Connector for SCEP mengeluarkan sertifikat dengan EKU yang disetel. `Client Authentication (Object Identifier: 1.3.6.1.5.5.7.3.2)` Untuk perangkat macOS atau iOS, Microsoft Intune tidak menghormati `ExtendedKeyUsage` atau `Validity` parameter dalam profil konfigurasi Anda. Untuk perangkat ini, Konektor untuk SCEP mengeluarkan sertifikat dengan masa berlaku satu tahun ke perangkat ini melalui otentikasi klien.

Verifikasi koneksi ke Konektor untuk SCEP

Setelah Anda membuat profil konfigurasi Microsoft Intune yang mengarah ke titik akhir Connector for SCEP, konfirmasikan bahwa perangkat yang terdaftar dapat meminta sertifikat. Untuk mengonfirmasi, pastikan tidak ada kegagalan penetapan kebijakan. Untuk mengonfirmasi, di portal Intune navigasikan ke Devices > Manage Devices > Configuration dan verifikasi bahwa tidak ada yang tercantum di bawah Kegagalan Penugasan Kebijakan Konfigurasi. Jika ada, konfirmasikan pengaturan Anda dengan informasi dari prosedur sebelumnya. Jika pengaturan Anda benar dan masih ada kegagalan, maka konsultasikan Kumpulkan data yang tersedia dari perangkat seluler.

Untuk informasi tentang pendaftaran perangkat, lihat Apa itu pendaftaran perangkat? dalam dokumentasi Microsoft Intune.

Pemecahan Masalah

Konsultasikan topik berikut jika Anda memiliki masalah dalam menggunakan AWS Private CA.

Topik

Membuat dan menandatangani sertifikat CA privat

Setelah membuat CA privat, Anda harus mengambil CSR dan mengirimkannya ke CA perantara atau akar di infrastruktur X.509 Anda. CA Anda menggunakan CSR untuk membuat sertifikat CA privat Anda dan kemudian menandatangani sertifikat tersebut sebelum mengembalikannya kepada Anda.

Sayangnya, tidak mungkin memberikan saran khusus tentang masalah yang terkait dengan pembuatan dan penandatanganan sertifikat CA privat Anda. Detail infrastruktur X.509 Anda dan hierarki CA di dalamnya berada di luar cakupan dokumentasi ini. AWS Private CA

Latensi dalam tanggapan OCSP

Responsif OCSP mungkin lebih lambat jika penelepon secara geografis jauh dari cache tepi regional atau dari Wilayah CA penerbit. Untuk informasi selengkapnya tentang ketersediaan cache edge regional, lihat Global Edge Network. Kami merekomendasikan untuk menerbitkan sertifikat di Wilayah dekat tempat mereka akan digunakan.

Mengonfigurasi Amazon S3 untuk memungkinkan pembuatan bucket CRL

CA privat Anda mungkin gagal membuat bucket CRL jika Blokir akses publik (pengaturan bucket) Amazon S3 diterapkan di akun Anda. Periksa pengaturan Amazon S3 Anda jika ini terjadi. Untuk informasi lebih lanjut, lihat Menggunakan Amazon S3 Block Public Access.

Membatalkan sertifikat CA yang ditandatangani sendiri

Anda tidak dapat mencabut sertifikat CA yang ditandatangani sendiri. Sebagai gantinya, Anda harus menghapus CA.

Menangani pengecualian

AWS Private CA Perintah mungkin gagal karena beberapa alasan. Untuk informasi tentang setiap pengecualian dan rekomendasi untuk mengatasinya, lihat tabel di bawah ini.

AWS Private CA Pengecualian

Pengecualian Dikembalikan oleh AWS Private CA	Deskripsi	Remediasi
`AccessDeniedException`	Izin yang diperlukan untuk menggunakan perintah yang diberikan belum didelegasikan oleh CA privat ke akun panggilan.	Untuk informasi tentang mendelegasikan izin AWS Private CA, lihat. Menetapkan izin perpanjangan sertifikat untuk ACM
`InvalidArgsException`	Pembuatan sertifikat atau permintaan perpanjangan dibuat dengan parameter yang tidak valid.	Periksa dokumentasi individual perintah untuk memastikan parameter input Anda valid. Jika Anda membuat sertifikat baru, pastikan bahwa algoritme penandatanganan yang diminta dapat digunakan dengan jenis kunci CA.

Pengecualian Dikembalikan oleh AWS Private CA	Deskripsi	Remediasi
`InvalidStateException`	CA privat terkait tidak dapat memperbarui sertifikat karena tidak dalam status `ACTIVE`.	Mencoba memulihkan CA privat. Jika CA privat berada di luar masa pemulihannya, CA tidak dapat dipulihkan dan sertifikat tidak dapat diperpanjang.
`LimitExceededException`	Setiap otoritas sertifikasi (CA) memiliki kuota sertifikat yang dapat diterbitkan. CA privat yang terkait dengan sertifikat yang ditunjuk telah mencapai kuotanya. Untuk informasi selengkapnya, lihat Service Quotas di Panduan. Referensi Umum AWS	Hubungi AWS Support Pusat untuk meminta kenaikan kuota.
`MalformedCSRException`	Permintaan penandatanganan sertifikat (CSR) yang diajukan AWS Private CA tidak dapat diverifikasi atau divalidasi.	Konfirmasikan bahwa CSR Anda dibuat dan dikonfigurasi dengan benar.
`OtherException`	Kesalahan internal telah menyebabkan permintaan gagal.	Coba untuk menjalankan perintah lagi. Jika masalah berlanjut, hubungi AWS Support Pusat.
`RequestFailedException`	Masalah jaringan di AWS lingkungan Anda menyebabkan permintaan gagal.	Coba lagi permintaannya. Jika kegagalan berlanjut, periksa konfigurasi Amazon VPC (VPC) Anda.
`ResourceNotFoundException`	CA privat yang mengeluarkan sertifikat telah dihapus dan tidak ada lagi.	Minta sertifikat baru dari CA aktif lainnya.

Pengecualian Dikembalikan oleh AWS Private CA	Deskripsi	Remediasi
`ThrottlingException`	Tindakan API yang diminta gagal karena melebihi kuota.	Konfirmasikan bahwa Anda tidak mengeluarkan lebih banyak panggilan daripada yang diizinkan oleh AWS Private CA. Kesalahan `ThrottlingException` juga dapat terjadi karena Anda mengalami kondisi sementara dan bukan dari kuota yang terlampaui. Jika Anda menemukan kesalahan dan Anda belum melakukan panggilan melebihi kuota, coba permintaan Anda lagi. Jika Anda kehabisan kuota, Anda mungkin dapat meminta peningkatan. Untuk informasi selengkapnya, lihat Service Quotas di Panduan. Referensi Umum AWS
`ValidationException`	Parameter masukan permintaan salah diformat, atau masa berlaku sertifikat akar berakhir sebelum masa berlaku sertifikat yang diminta.	Periksa persyaratan sintaksis dari parameter input perintah serta masa berlaku sertifikat akar CA Anda. Untuk informasi tentang mengubah masa berlaku, lihat Memperbarui CA privat Anda.

Menggunakan standar Matter

Standar konektivitas Matter menentukan konfigurasi sertifikat yang meningkatkan keamanan dan konsistensi perangkat internet of things (IoT). Sampel Java untuk membuat sertifikat CA root, CA perantara, dan entitas akhir yang sesuai dengan Matter dapat ditemukan di. Menggunakan AWS Private CA API untuk mengimplementasikan standar Matter (contoh Java)

Untuk membantu pemecahan masalah, pengembang Matter menyediakan alat verifikasi sertifikat yang disebut chip-cert. Kesalahan yang dilaporkan alat tercantum dalam tabel berikut dengan remediasi.

Kode kesalahar	Arti	Remediasi
0x000003 5	`BasicCons traints` ,`KeyUsage`, dan `ExtensionKeyUsage` ekstensi harus ditandai kritis.	Pastikan bahwa Anda telah memilih template yang benar untuk penggunaan Anda.
0x000000 0	Ekstensi pengenal kunci otoritas harus ada.	AWS Private CA tidak menyetel ekstensi pengenal kunci otorita sertifikat root. Anda harus menghasilkan AuthorityKeyIdentifier yang dikodekan Base64 menggunakan CSR dan kemudian me nnya melalui file. CustomExtension Lihat informasi yang lebih le di Aktifkan Root CA untuk Node Operational Certificates (NOC) Aktifkan Otoritas Pengesahan Produk (PAA).
0x000000 E	Sertifikat kedaluwarsa.	Pastikan sertifikat yang Anda gunakan belum kedaluwarsa.
0x000000 4	Kegagalan validasi rantai sertifikat.	Kesalahan ini mungkin terjadi jika Anda mencoba membuat ser entitas akhir yang sesuai dengan Matter tanpa menggunakan c Java yang disediakan, yang menggunakan AWS Private CA AP meneruskan konfigurasi dengan benar. KeyUsage Secara default, AWS Private CA menghasilkan nilai KeyUsage sembilan bit, dengan bit kesembilan menghasilkan byte tambah Matter mengabaikan byte ekstra selama konversi format, meny kegagalan validasi rantai. Namun, CustomExtensiondalam `API`

Kode kesalahar	Arti	Remediasi

		`rough` template dapat digunakan untuk mengatur jumlah byte tepat dalam `KeyUsage` nilai. Sebagai contoh, lihat Membuat No Operational Certificate (NOC).
		
		Jika Anda memodifikasi kode sampel atau menggunakan utilitas alternatif seperti OpenSSL, Anda perlu melakukan verifikasi ma untuk menghindari kesalahan validasi rantai.
		
		Untuk memverifikasi bahwa konversi lossless
		
		1. Gunakan openssl untuk memverifikasi bahwa sertifikat sert t node (end-entity) berisi rantai yang valid. Dalam contoh in `rcac.pem` adalah sertifikat CA root, `icac.pem` adalah ser perantara, dan `noc.pem` merupakan sertifikat node.
		
		`openssl verify -verbose -CAfile <(cat rcac.pem icac noc.pem`
		
		2. Gunakan chip-cert untuk mengonversi sertifikat node berfo PEM ke format TLV (tag, panjang, nilai) dan kembali lagi.
		
		`./chip-cert convert-cert noc.pem noc.chip -c`
		`./chip-cert convert-cert noc.chip noc_converted.pem`
		
		File `noc.pem` dan `noc_converted.pem` harus persis sa seperti yang dikonfirmasi oleh alat perbandingan string.

Konektor untuk kesalahan dan kegagalan AD

Gunakan informasi di sini untuk membantu Anda mendiagnosis dan memperbaiki kesalahan dan kegagalan pembuatan saat Anda bekerja dengan Connector for AD.

Topik

Kesalahan

Konektor untuk AD mengirim pesan kesalahan karena beberapa alasan. Untuk informasi tentang setiap kesalahan dan rekomendasi tentang menyelesaikannya, lihat tabel berikut. Anda dapat menerima kesalahan ini dengan berlangganan acara Amazon EventBridge Scheduler (sumber acara:`aws.pca-connector-ad`) atau dengan menggunakan pendaftaran manual di Windows.

Kode kesalahan	Arti	Remediasi
0x8FFFA000	Otentikasi Kerberos gagal.	Jika Anda menggunakan pendaftaran otomatis, perbaiki prinsip layanan AWS sumber daya Anda. Jika Anda menggunakan Active Directory UI untuk mendapatkan sertifikat, jalankan`gpupdate /force` .
0x8FFFA001	Pesan SOAP harus berisi header tindakan.	Tambahkan header tindakan.
0x8FFFA002	Konektor tidak memiliki akses ke CA pribadi yang terhubung dengannya.	Bagikan CA pribadi Anda dengan konektor dengan membuat AWS Resource Access Manager (RAM) untuk berbagi antara CA pribadi Anda dan Connector for AD service.
0x8FFFA003	CA pribadi untuk konektor ini tidak aktif.	Pindahkan CA pribadi ke status Aktif. Jika CA pribadi Anda dalam status sertifikat tertunda, maka instal sertifikat CA.

Kode kesalahan	Arti	Remediasi
0x8FFFA004	CA pribadi untuk konektor ini tidak ada.	Pindahkan otoritas sertifikat Anda ke status Aktif jika berada dalam status Dihapus. Jika CA pribadi Anda dihapus secara permanen, buat konektor baru dengan CA yang berbeda.
0x8FFFA005	Template menentukan `directoryGuid` atribut untuk subjek sertifikat atau nama alternatif subjek, tetapi atribut tidak ditemukan di objek AD untuk pemohon.	Active Directory tidak menghasilkan `directoryGuid` untuk direktori Anda. Memecahkan masalah di Active Directory.
0x8FFFA006	Template menentukan `dnsHostName` atribut untuk subjek sertifikat atau nama alternatif subjek, tetapi atribut tidak ditemukan di objek AD untuk pemohon.	Tambahkan `dnsHostName` atribut ke objek AD Anda.
0x8FFFA007	Template menetapkan atribut email yang akan disertakan dalam subjek sertifikat atau nama alternatif subjek, tetapi atribut tidak ditemukan di objek AD untuk pemohon.	Menambahkan atribut email ke objek AD

Kode kesalahan	Arti	Remediasi
0x8FFFA008	Pesan SOAP harus memiliki header tindakan salah satu `http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy/IPolicy/GetPolicies` atau`http://schemas.microsoft.com/windows/pki/2009/01/enrollment/RST/wstep` .	Perbarui header tindakan untuk menggunakan salah satu nilai yang ditentukan.
0x8FFFA009	BinarySecurityToken Harus dikodekan. `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary`	Perbarui jenis token keamanan biner.
0x8FFFA00A	BinarySecurityToken Itu tidak valid.	Periksa apakah CSR dihasilkan dengan benar.
0x8FFFA00B	BinarySecurityToken Harus memiliki jenis nilai salah satu `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#PKCS7` atau`http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10` .	Perbarui jenis nilai token keamanan biner ke nilai yang valid.

Kode kesalahan	Arti	Remediasi
0x8FFFA00C	CMS BinarySecurityToken yang terkandung tidak valid.	Base64 valid tetapi sintaks pesan kriptografi (CMS) tidak valid. Tinjau sintaks CMS.
0x8FFFA00D	Yang BinarySecurityToken berisi CSR yang tidak valid.	Periksa apakah CSR dihasilkan dengan benar.
0x8FFFA00E	CA pribadi tidak dapat mengeluar kan sertifikat menggunakan templat tertentu.	Tinjau pengecualian validasi dari AWS Private CA. Anda dapat melihat pengecualian validasi di Amazon EventBridge atau AWS CloudTrail.
0x8FFFA00F	Pesan SOAP harus memiliki jenis permintaan`http://do cs.oasis-open.org/ws-sx/ ws-trust/200512/Issue` .	Tetapkan jenis permintaan ke`http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .
0x8FFFA010	Pesan SOAP harus memiliki header to dari bidang konektor atau `CertificateEnrollm entPolicyServerEndpoint` bidang URI dalam respons XCEP.	Tetapkan header token keamanan permintaan ke `Certifica teEnrollmentPolicy ServerEndpoint` bidang atau bidang URI dalam respons XCEP.
0x8FFFA011	Pesan SOAP harus hanya memiliki satu header tindakan.	Tinjau header pesan SOAP dari token keamanan permintaan dan atur header dengan benar.
0x8FFFA012	Pesan SOAP harus hanya memiliki satu `messageId` header.	Tinjau header pesan SOAP dari token keamanan permintaan dan atur header dengan benar.

Kode kesalahan	Arti	Remediasi
0x8FFFA013	Pesan SOAP harus hanya memiliki satu ke header.	Tinjau header pesan SOAP dari token keamanan permintaan dan atur header dengan benar.
0x8FFFA014	Pemohon tidak memiliki akses ke template yang diminta.	Izinkan grup pemohon untuk mendaftar menggunakan template yang diminta dengan membuat Entri Kontrol Akses.
0x8FFFA015	Entah ekstensi `CertificateTemplateInformation` atau `CertificateTemplateName` ekstensi harus ada di BinarySecurityToken.	Tambahkan ekstensi keamanan ke CSR Anda.
0x8FFFA016	Template yang diminta tidak ditemukan untuk konektor yang diberikan.	Template adalah sumber daya anak untuk setiap konektor. Buat template untuk konektor menggunak an`createTemplate` .
0x8FFFA017	Permintaan ditolak karena throttling permintaan.	Memperlambat tingkat permintaan.
0x8FFFA018	Pesan SOAP harus berisi `to` header.	Tinjau header pesan SOAP.
0x8FFFA019	Tidak dapat memproses pesan SOAP karena header yang tidak dikenal.	Tinjau header pesan SOAP.

Kode kesalahan	Arti	Remediasi
0x8FFFA01A	Template menetapkan atribut UPN untuk disertakan dalam subjek sertifikat atau nama alternatif subjek, tetapi atribut tidak ditemukan di objek AD untuk pemohon.	Tambahkan UPN ke objek Active Directory.

Kegagalan pembuatan konektor

Pembuatan konektor dapat gagal karena berbagai alasan. Saat pembuatan konektor gagal, Anda akan menerima alasan kegagalan dalam respons API. Jika Anda menggunakan konsol, maka alasan kegagalan ditampilkan di halaman detail konektor di bawah bidang Detail status tambahan di dalam wadah detail konektor. Tabel berikut menjelaskan alasan kegagalan dan langkah-langkah yang direkomendasikan untuk resolusi.

Status kegagalan	Deskripsi	Remediasi
`CA_CERTIFICATE_REGISTRATION_FAILED`	Konektor untuk AD tidak dapat mengimpor sertifikat CA ke direktori Anda.	Tinjau halaman Prasyarat dan periksa apakah akun layanan Anda memiliki izin yang tepat. Setelah mendelegasikan izin yang benar ke akun layanan Anda, hapus konektor yang gagal dan buat yang baru. Untuk informasi tentang mendelegasikan izin, lihat Mendelegasikan hak istimewa ke akun layanan Anda di Panduan Administrasi.AWS Directory Service
`DIRECTORY_ACCESS_DENIED`	Konektor untuk AD tidak dapat mengakses direktori Anda.	Anda harus memberikan Konektor untuk akses AD

Status kegagalan	Deskripsi	Remediasi
		ke direktori Anda. Tinjau Kebijakan IAM bagian ini untuk memastikan bahwa kebijakan IAM yang terkait dengan AWS akun Anda memungkinkan Anda mengakses dan mendeskripsikan direktori. Setelah memberikan izin yang benar untuk AWS peran Anda, hapus konektor yang gagal dan buat yang baru.
`INTERNAL_FAILURE`	Konektor untuk AD mengalami kegagalan internal.	Coba lagi nanti. Hapus konektor yang gagal dan buat yang baru.

Status kegagalan	Deskripsi	Remediasi
`PRIVATECA_ACCESS_DENIED`	Konektor untuk AD tidak dapat mengakses CA pribadi Anda.	Tinjau halaman Prasyarat dan periksa apakah Anda memiliki izin untuk membuat konektor. Untuk informasi, lihat Kebijakan IAM. Jika Anda membuat konektor melalui AWS CLI atau API, tinjau halaman Prasyarat dan periksa apakah Anda telah membagikan CA pribadi dengan Connector for AD menggunakan. AWS Resource Access Manager Setelah memeriksa dan memperbaiki izin IAM dan berbagi AWS RAM sumber daya, hapus konektor yang gagal dan buat yang baru.
`PRIVATECA_RESOURCE_NOT_FOUND`	Konektor untuk AD tidak dapat menemukan CA pribadi yang ditentukan.	Pastikan Anda menentukan CA Amazon Resource Name (ARN) pribadi yang benar, lalu hapus konektor yang gagal dan buat yang baru menggunakan CA ARN pribadi yang Anda inginkan.

Status kegagalan	Deskripsi	Remediasi
SECURITY_GROUP_NOT_IN_VPC	Grup keamanan tidak ada di VPC yang meng-host direktori Anda.	Gunakan grup keamanan yang ada di VPC yang meng-host direktori Anda. Untuk informasi selengkapnya, lihat Grup keamanan. Hapus konektor yang gagal dan buat yang baru dengan grup keamanan yang ada di VPC.
VPC_ACCESS_DENIED	Konektor untuk AD tidak dapat mengakses VPC Amazon yang menghosting direktori Anda.	Periksa izin IAM Anda. Hapus konektor yang gagal dan buat yang baru. Untuk contoh kebijakan IAM yang menyertakan izin akses, lihat Kebijakan IAM
VPC_ENDPOINT_LIMIT_EXCEEDED	Konektor untuk AD tidak dapat membuat titik akhir di VPC Amazon Anda. Anda telah mencapai batas titik akhir VPC yang dapat Anda buat untuk akun Anda.	Hapus titik akhir Amazon VPC, atau minta peningkatan batas. Setelah Anda melakukan salah satu dari dua langkah, hapus konektor yang gagal dan buat yang baru. Untuk informasi tentang kuota, lihat kuota Layanan Amazon Virtual Private Cloud.
VPC_RESOURCE_NOT_FOUND	Konektor untuk AD tidak dapat menemukan VPC yang ditentukan.	Pastikan Anda menentukan VPC yang benar dan VPC ada. Kemudian hapus konektor yang gagal dan buat yang baru menggunakan ID VPC yang benar.

Kegagalan pembuatan SPN

Pembuatan nama utama layanan (SPN) dapat gagal karena berbagai alasan. Ketika pembuatan SPN gagal, Anda akan menerima alasan kegagalan dalam respons API. Jika Anda menggunakan konsol, maka alasan kegagalan ditampilkan di halaman Detail konektor di bawah bidang Detail status tambahan dalam wadah Nama utama layanan (SPN). Tabel berikut menjelaskan alasan kegagalan dan langkah-langkah yang direkomendasikan untuk resolusi.

Status kegagalan	Deskripsi	Remediasi
`DIRECTORY_ACCESS_DENIED`	Konektor untuk AD tidak dapat mengakses direktori Anda.	Berikan Konektor untuk akses AD ke direktori Anda. Untuk contoh kebijakan IAM yang menyertakan izin yang memberikan akses direktori, lihat. Kebijakan IAM
`DIRECTORY_NOT_REACHABLE`	Konektor untuk AD tidak dapat mengakses direktori Anda.	Periksa jaringan antara AWS dan direktori Anda, dan coba buat SPN lagi.
`DIRECTORY_RESOURCE_NOT_FOUND`	Konektor untuk AD tidak dapat menemukan direktori yang ditentukan.	Pastikan Anda menentukan ID direktori yang benar, lalu hapus konektor yang gagal dan buat yang baru menggunakan ID direktori yang Anda inginkan.
`INTERNAL_FAILURE`	Konektor untuk AD mengalami kegagalan internal.	Coba lagi nanti.
`SPN_EXISTS_ON_DIFFERENT_AD_OBJECT`	Nama utama layanan (SPN) ada pada objek Active Directory yang berbeda.	Hapus SPN dari objek Active Directory, dan coba buat SPN lagi.

Status kegagalan	Deskripsi	Remediasi
SPN_LIMIT_EXCEEDED	Konektor untuk AD tidak dapat membuat SPN karena Anda telah mencapai batas SPN per direktori. Jumlah maksimum SPN per direktori adalah 10.	Hapus satu atau beberapa SPN dari akun Anda, dan coba buat SPN lagi.

Konektor untuk kesalahan kegagalan pembuatan konektor AD

Konektor untuk pembuatan konektor AD dapat gagal karena berbagai alasan. Saat pembuatan konektor gagal, Anda akan menerima alasan kegagalan dalam respons API. Jika Anda menggunakan konsol, maka alasan kegagalan ditampilkan di halaman Detail konektor di bawah bidang Detail status tambahan di dalam wadah Detail konektor. Tabel berikut menjelaskan alasan kegagalan dan langkah-langkah yang direkomendasikan untuk resolusi.

Istilah dan Konsep

Istilah dan konsep berikut dapat membantu Anda saat Anda bekerja dengan AWS Private Certificate Authority (AWS Private CA).

Topik

Percaya

Agar peramban web mempercayai identitas situs web, peramban harus dapat memverifikasi sertifikat situs web. Namun, peramban hanya mempercayai sejumlah kecil sertifikat yang dikenal sebagai sertifikat akar CA. Pihak ketiga tepercaya, yang dikenal sebagai otoritas sertifikasi (CA), memvalidasi identitas situs web dan menerbitkan sertifikat digital yang ditandatangani ke operator situs web. Peramban kemudian dapat memeriksa tanda tangan digital untuk memvalidasi identitas situs web. Jika validasi berhasil, peramban menampilkan ikon kunci di bilah alamat.

Sertifikat server TLS

Transaksi HTTPS memerlukan sertifikat server untuk mengautentikasi server. Sertifikat server adalah struktur data X.509 v3 yang mengikat kunci publik dalam sertifikat ke subjek sertifikat. Sertifikat

TLS ditandatangani oleh otoritas sertifikat (CA). Ini berisi nama server, masa berlaku, kunci publik, algoritme tanda tangan, dan banyak lagi.

Tanda tangan sertifikat

Tanda tangan digital adalah hash yang dienkripsi melalui sertifikat. Tanda tangan digunakan untuk menegaskan integritas data sertifikat. CA privat Anda membuat tanda tangan dengan menggunakan fungsi hash kriptografi seperti SHA256 atas konten sertifikat ukuran variabel. Fungsi hash ini menghasilkan string data ukuran tetap yang efektif tidak dapat ditempa. String ini disebut hash. CA kemudian mengenkripsi nilai hash dengan kunci privat dan menggabungkan hash yang dienkripsi dengan sertifikat.

Otoritas sertifikat

Otoritas sertifikasi (CA) menerbitkan dan jika perlu mencabut sertifikat digital. Jenis sertifikat yang paling umum didasarkan pada standar ISO X.509. Sertifikat X.509 menegaskan identitas subjek sertifikat dan mengikat identitas tersebut ke kunci publik. Subjek dapat berupa pengguna, aplikasi, komputer, atau perangkat lain. CA menandatangani sertifikat dengan melakukan hash pada konten dan kemudian mengenkripsi hash dengan kunci privat yang terkait dengan kunci publik dalam sertifikat. Aplikasi klien seperti peramban web yang perlu menegaskan identitas subjek menggunakan kunci publik untuk mendekripsi tanda tangan sertifikat. Ini kemudian melakukan hash pada isi sertifikat dan membandingkan nilai hash dengan tanda tangan yang didekripsi untuk menentukan apakah cocok. Untuk informasi selengkapnya tentang penandatanganan sertifikat, lihat Tanda tangan sertifikat.

Anda dapat menggunakan AWS Private CA untuk membuat CA pribadi dan menggunakan CA pribadi untuk menerbitkan sertifikat. CA privat Anda hanya menerbitkan sertifikat SSL/TLS pribadi untuk digunakan dalam organisasi Anda. Untuk informasi lebih lanjut, lihat Sertifikat pribadi. CA privat Anda juga memerlukan sertifikat sebelum dapat menggunakannya. Untuk informasi selengkapnya, lihat Sertifikat CA.

CA akar

Blok bangunan kriptografi dan akar kepercayaan tempat sertifikat dapat diterbitkan. Ini terdiri dari kunci privat untuk menandatangani (menerbitkan) sertifikat dan sertifikat akar yang mengidentifikasi CA akar dan mengikat kunci privat ke nama CA. Sertifikat akar didistribusikan ke penyimpanan kepercayaan setiap entitas di lingkungan. Administrator membuat penyimpanan kepercayaan untuk

menyertakan hanya CA yang dipercayai. Administrator memperbarui atau membangun penyimpanan kepercayaan ke dalam sistem operasi, instans, dan citra mesin host entitas di lingkungannya. Ketika sumber daya mencoba untuk terhubung satu sama lain, mereka memeriksa sertifikat yang disajikan setiap entitas. Klien memeriksa sertifikat untuk validitas dan apakah ada rantai dari sertifikat ke sertifikat root yang diinstal di penyimpanan kepercayaan. Jika kondisi tersebut terpenuhi, "jabat tangan" dilakukan antara sumber daya. Jabat tangan ini secara kriptografis membuktikan identitas masing-masing entitas dengan yang lain dan menciptakan saluran komunikasi terenkripsi (TLS/SSL) di antaranya.

Sertifikat CA

Sertifikat otoritas sertifikasi (CA) menegaskan identitas CA dan mengikatnya ke kunci publik yang ada dalam sertifikat.

Anda dapat menggunakan AWS Private CA untuk membuat CA root pribadi atau CA bawahan pribadi, masing-masing didukung oleh sertifikat CA. Sertifikat CA bawahan ditandatangani oleh sertifikat CA lain yang lebih tinggi dalam rantai kepercayaan. Tetapi dalam kasus CA akar, sertifikat ditandatangani sendiri. Anda juga dapat membuat otoritas akar eksternal (dihosting on premise, misalnya). Anda kemudian dapat menggunakan otoritas root Anda untuk menandatangani sertifikat CA root bawahan yang dihosting olehAWS Private CA.

Contoh berikut menunjukkan bidang tipikal yang terkandung dalam sertifikat CA AWS Private CA X.509. Perhatikan bahwa untuk sertifikat CA, nilai `CA:` dalam bidang `Basic Constraints` diatur ke `TRUE`.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4121 (0x1019)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=Washington, L=Seattle, O=Example Company Root CA, OU=Corp,
 CN=www.example.com/emailAddress=corp@www.example.com
        Validity
            Not Before: Feb 26 20:27:56 2018 GMT
            Not After : Feb 24 20:27:56 2028 GMT
        Subject: C=US, ST=WA, L=Seattle, O=Examples Company Subordinate CA,
 OU=Corporate Office, CN=www.example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
```

```
                Modulus:
                    00:c0: ... a3:4a:51
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                F8:84:EE:37:21:F2:5E:0B:6C:40:C2:9D:C6:FE:7E:49:53:67:34:D9
            X509v3 Authority Key Identifier:
                keyid:0D:CE:76:F2:E3:3B:93:2D:36:05:41:41:16:36:C8:82:BC:CB:F8:A0

            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Digital Signature, CRL Sign
    Signature Algorithm: sha256WithRSAEncryption
         6:bb:94: ... 80:d8
```

Sertifikat Root CA

Otoritas sertifikasi (CA) biasanya ada dalam struktur hierarkis yang berisi beberapa CA lain dengan hubungan induk-turunan yang jelas di antaranya. CA turunan atau bawahan disertifikasi oleh CA induknya, membuat rantai sertifikat. CA di bagian atas hierarki disebut sebagai CA akar, dan sertifikatnya disebut sertifikat akar. Sertifikat ini biasanya ditandatangani sendiri.

Sertifikat entitas akhir

Sertifikat entitas akhir mengidentifikasi sumber daya, seperti server, instans, kontainer, atau perangkat. Tidak seperti sertifikat CA, sertifikat entitas akhir tidak dapat digunakan untuk menerbitkan sertifikat. Istilah umum lainnya untuk sertifikat entitas akhir adalah sertifikat “klien” atau “daun”.

Sertifikat yang ditandatangani sendiri

Sertifikat yang ditandatangani oleh penerbit, bukan CA yang lebih tinggi. Tidak seperti sertifikat yang dikeluarkan dari akar aman yang dikelola oleh CA, sertifikat yang ditandatangani sendiri bertindak sebagai akar mereka sendiri, dan akibatnya sertifikat tersebut memiliki batasan yang signifikan: Sertifikat tersebut dapat digunakan untuk menyediakan enkripsi kabel tetapi tidak untuk memverifikasi identitas, dan sertifikat tersebut tidak dapat dicabut. Mereka tidak dapat diterima dari perspektif keamanan. Tetapi organisasi tetap menggunakannya karena mudah dibuat, tidak memerlukan keahlian atau infrastruktur, dan banyak aplikasi menerimanya. Tidak ada kontrol untuk menerbitkan sertifikat yang ditandatangani sendiri. Organisasi yang menggunakannya mengalami

risiko pemadaman yang lebih besar yang disebabkan oleh masa berlaku sertifikat karena tidak memiliki cara untuk melacak tanggal kedaluwarsa.

Sertifikat pribadi

AWS Private CAsertifikat adalah sertifikat SSL/TLS pribadi yang dapat Anda gunakan dalam organisasi Anda, tetapi tidak dipercaya di internet publik. Gunakan mereka untuk mengidentifikasi sumber daya seperti klien, server, aplikasi, layanan, perangkat, dan pengguna. Saat membuat saluran komunikasi terenkripsi yang aman, setiap sumber daya menggunakan sertifikat seperti berikut ini serta teknik kriptografi untuk membuktikan identitasnya ke sumber daya lain. Titik akhir API internal, server web, pengguna VPN, perangkat IoT, dan banyak aplikasi lainnya menggunakan sertifikat privat untuk membuat saluran komunikasi terenkripsi yang diperlukan untuk operasi aman mereka. Secara default, sertifikat privat tidak dipercaya secara publik. Administrator internal harus secara eksplisit mengonfigurasi aplikasi untuk memercayai sertifikat privat dan mendistribusikan sertifikat.

```
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
```

Jalur sertifikat

Klien yang mengandalkan sertifikat memvalidasi bahwa jalur ada dari sertifikat entitas akhir, mungkin melalui rantai sertifikat perantara, ke root tepercaya. Klien memeriksa bahwa setiap sertifikat sepanjang jalur sudah valid (tidak dicabut). Ini juga memeriksa bahwa sertifikat entitas akhir belum kedaluwarsa, memiliki integritas (belum dirusak atau dimodifikasi), dan bahwa kendala dalam sertifikat diberlakukan.

Kendala panjang jalur

Kendala dasar pathLenConstraintuntuk sertifikat CA menetapkan jumlah sertifikat CA bawahan yang mungkin ada dalam rantai di bawahnya. Misalnya, sertifikat CA dengan batasan panjang jalur nol tidak dapat memiliki CA bawahan. CA dengan batasan panjang jalur satu mungkin memiliki hingga satu tingkat CA bawahan di bawahnya. RFC 5280 mendefinisikan ini sebagai, “jumlah maksimum sertifikat non-self-issued perantara yang dapat mengikuti sertifikat ini di jalur sertifikasi yang valid.” Nilai panjang jalur tidak termasuk sertifikat entitas akhir, meskipun bahasa informal tentang “panjang” atau “kedalaman” rantai validasi dapat menyertakannya... yang menyebabkan kebingungan.

Riwayat Dokumen

Tabel berikut menjelaskan perubahan signifikan pada dokumentasi ini sejak Januari 2018. Selain perubahan besar yang ditampilkan di sini, kami juga sering memperbarui dokumentasi untuk memperbaiki deskripsi dan contoh, serta membahas umpan balik yang Anda kirimkan kepada kami. Agar menerima pemberitahuan tentang perubahan signifikan, gunakan tautan di sudut kanan atas untuk berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
AWS Private CA sekarang mendukung Konektor untuk SCEP (Pratinjau)	Gunakan Konektor untuk SCEP untuk menautkan AWS Private CA ke klien dan perangkat berkemampuan SCEP Anda.	Juni 11, 2024
Panduan pemecahan masalah konektor baru	Menambahkan dua bagian baru pada konektor pemecahan masalah dan kegagalan pembuatan SPN.	April 4, 2024
Menambahkan ekstensi CDP untuk Matter	Menambahkan dukungan untuk ekstensi Certificate Revocation List Distribution Point (CDP) untuk Matter.	Januari 25, 2024
AWS Private CA Dukungan API untuk mDL	Menambahkan dukungan API untuk membuat sertifikat yang sesuai dengan standar ISO/IEC untuk SIM seluler (mDL).	Januari 16, 2024
AWS Private CA Konektor untuk Active Directory	Panduan pengguna, API, dan dukungan CLI untuk Konektor untuk AD. Untuk informasi selengkapnya, lihat Konektor untuk dokumentasi AD.	24 Agustus 2023

Mengubah nama kebijakan keamanan agar sesuai dengan nama layanan baru	Adopsi nama baru untuk kebijakan IAM AWS terkelola yang menentukan izin standar pada. AWS Private CA Untuk informasi selengkapnya, lihat Kebijakan terkelola AWS.	13 Februari 2023
Menambahkan pelacak perubahan untuk kebijakan AWS terkelola	Dokumentasi ditambahkan untuk melacak perubahan pada kebijakan IAM AWS terkelola yang menentukan izin standar. AWS Private CA Untuk informasi selengkapnya, lihat Memperbarui kebijakan AWS terkelola untuk AWS Private CA.	11 November 2022
Dukungan API dan CLI untuk CA yang mengeluarkan sertifikat berumur pendek	Dengan diperkenalkannya mode penggunaan CA, CA dapat dikonfigurasi untuk mengeluarkan sertifikat tujuan umum atau hanya berumur pendek. Untuk informasi selengkapnya, lihat Mode otoritas sertifikat.	24 Oktober 2022
Rebranding layanan dan pembaruan konsol	Layanan ini berganti nama menjadi AWS Private Certificate Authority (AWS Private CA). AWS Private CA Konsol mendapatkan peningkatan kegunaan termasuk panel bantuan terintegrasi yang menautkan ke dokumentasi lengkap.	September 27, 2022

Dukungan sertifikat yang sesuai dengan materi	Tiga templat sertifikat baru menambahkan dukungan untuk CA yang sesuai dengan Matter dan sertifika t entitas akhir. Untuk informasi selengkapnya, lihat Memahami templat sertifikat.	20 Juli 2022
Dukungan wilayah baru	Endpoint ditambahkan untuk Asia Pasifik (Jakarta). Untuk daftar lengkap titik akhir, lihat AWS Private CA Titik Akhir dan Kuota Otoritas Sertifikat Pribadi ACM.	4 Mei, 2022
Support untuk Atribut dan Ekstensi Kustom	Gunakan CustomAttributeobj ek untuk mengonfigurasi CA dan sertifikat yang disesuaik an, dan CustomExtension objek untuk mengonfigurasi sertifikat yang disesuaikan.	16 Maret 2022
Support untuk OCSP Terkelola	Lihat Menyiapkan metode pencabutan sertifikat untuk opsi pencabutan termasuk OCSP.	18 Agustus 2021
Dukungan untuk fitur S3 Block Public Access untuk CRL	Lihat Mengaktifkan fitur Blokir Akses Publik S3.	27 Mei 2021
Contoh implementasi Java yang baru dan diperbarui	Lihat Menggunakan ACM Private CA API (Contoh Java).	9 September 2020

Dukungan wilayah baru	Titik akhir ditambahkan untuk Africa (Cape Town) dan Europe (Milan). Untuk daftar lengkap titik akhir, lihat AWS Private CA Titik Akhir dan Kuota Otoritas Sertifikat AWS Certificate Manager Pribadi.	27 Agustus 2020
Akses CA pribadi lintas akun didukung	AWS Certificate Manager pengguna dapat diberi wewenang untuk mengeluar kan sertifikat menggunak an CA pribadi yang tidak mereka miliki. Untuk informasi selengkapnya, lihat Akses Lintas Akun ke CA Privat.	17 Agustus 2020
Dukungan titik akhir VPC () PrivateLink	Menambahkan dukungan untuk penggunaan titik akhir VPC (AWS PrivateLink) untuk keamanan jaringan yang ditingkatkan. Untuk informasi selengkapnya, lihat Titik Akhir AWS PrivateLink VPC CA Pribadi ACM ().	26 Maret 2020
Bagian keamanan khusus ditambahkan	Dokumentasi keamanan untuk AWS telah dikonsoli dasikan ke dalam bagian keamanan khusus. Untuk informasi tentang keamanan, lihat Keamanan di Otoritas Sertifikat AWS Certificate Manager Pribadi.	26 Maret 2020

Template ARN ditambahkan ke laporan audit.	Untuk informasi selengkapnya, lihat Membuat Laporan Audit untuk CA Privat Anda.	6 Maret 2020
CloudFormation dukungan	Support ditambahkan untuk AWS CloudFormation. Untuk informasi selengkapnya, lihat Referensi Jenis Sumber Daya ACMPCA dalam Panduan Pengguna AWS CloudFormation .	22 Januari 2020
CloudWatch Integrasi acara	Integrasi dengan CloudWatch Acara untuk acara asinkron, termasuk pembuatan CA, penerbitan sertifikat, dan pembuatan CRL. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch Acara.	23 Desember 2019
Titik akhir FIPS	Titik akhir FIPS ditambahkan untuk AWS GovCloud (AS-Timur) dan AWS GovCloud (AS-Barat). Untuk daftar lengkap titik akhir, lihat AWS Private CA Titik Akhir dan Kuota Otoritas Sertifikat AWS Certificate Manager Pribadi.	13 Desember 2019

Izin berbasis tag	Izin berbasis tag didukung menggunakan API `TagResource` , `UntagResource` , dan `ListTagsForResource` baru. Untuk informasi umum tentang kontrol berbasis tag, lihat Mengontrol Akses ke dan untuk Pengguna dan Peran IAM Menggunakan Tag Sumber Daya IAM.	5 November 2019
Penegakan batasan nama	Menambahkan dukungan untuk menegakkan kendala nama subjek pada sertifikat CA yang diimpor. Untuk informasi selengkapnya, lihat Menegakkan Kendala Nama pada CA Privat.	28 Oktober 2019
Templat sertifikat baru	Template sertifikat baru ditambahkan, termasuk templat untuk penandatanganan kode AWS Signer. Untuk informasi selengkapnya, lihat Menggunakan templat.	1 Oktober 2019
Merencanakan CA Anda	Bagian baru ditambahkan pada perencanaan PKI Anda menggunakan AWS Private CA. Untuk informasi selengkapnya, lihat Merencanakan Deployment ACM Private CA Anda.	30 September 2019

Ditambahkan dukungan wilayah	Menambahkan dukungan wilayah untuk Wilayah AWS Asia Pasifik (Hong Kong). Untuk daftar lengkap wilayah yang didukung, lihat Titik Akhir dan Kuota Otoritas Sertifikat AWS Certificate Manager Pribadi.	24 Juli 2019
Menambahkan dukungan hierarki CA pribadi lengkap	Dukungan untuk membuat dan meng-host akar CA tidak akan memerlukan induk eksternal.	20 Juni 2019
Ditambahkan dukungan wilayah	Menambahkan dukungan wilayah untuk Wilayah AWS GovCloud (AS-Barat dan AS-Timur). Untuk daftar lengkap wilayah yang didukung, lihat AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	8 Mei 2019
Ditambahkan dukungan wilayah	Menambahkan dukungan wilayah untuk Wilayah AWS Asia Pasifik (Mumbai dan Seoul), AS Barat (California N.), dan Uni Eropa (Paris dan Stockholm). Untuk daftar lengkap wilayah yang didukung, lihat Titik Akhir dan Kuota Otoritas Sertifikat AWS Certificate Manager Pribadi.	4 April 2019

Menguji alur kerja pembaruan sertifikat	Pelanggan kini dapat menguji konfigurasi secara manual alur kerja pembaruan terkelola ACM mereka. Untuk informasi selengkapnya, lihat Menguji Konfigurasi Pembaruan Terkelola ACM.	14 Maret 2019
Ditambahkan dukungan wilayah	Menambahkan dukungan wilayah untuk Wilayah AWS UE (London). Untuk daftar lengkap wilayah yang didukung, lihat AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	1 Agustus 2018
Kembalikan CA yang dihapus	Pemulihan CA privat memungkinkan pelanggan untuk memulihkan otoritas sertifikasi (CA) untuk hingga 30 hari setelah dihapus. Untuk informasi selengkapnya, lihat Memulihkan CA Privat Anda.	20 Juni 2018

Pembaruan Sebelumnya

Tabel berikut menjelaskan riwayat rilis dokumentasi AWS Private Certificate Authority sebelum Juni 2018.

Perubahan	Deskripsi	Tanggal
Panduan baru	Rilis ini memperkenalkan AWS Private Certificate Authority.	4 April 2018

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.