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 root dan bawahan CAs, tanpa biaya investasi dan pemeliharaan pengoperasian CA lokal. Pribadi Anda CAs dapat menerbitkan 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 Konsol Manajemen AWS, menggunakan AWS Private CA API, atau menggunakan AWS CLI.

Topik

Ketersediaan regional untuk AWS Private Certificate Authority

Seperti kebanyakan AWS sumber daya, otoritas sertifikat swasta (CAs) adalah sumber daya Regional. Untuk menggunakan privat CAs di lebih dari satu Wilayah, Anda harus membuat CAs di Wilayah tersebut. Anda tidak dapat menyalin pribadi CAs 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 private CAs ke Amazon Elastic Kubernetes Service untuk memberikan penerbitan sertifikat di dalam klaster Kubernetes. Untuk informasi selengkapnya, lihat Amankan Kubernetes dengan AWS Private Certificate Authority.

> **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.

Algoritma kriptografi yang didukung di AWS Private Certificate Authority

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

Algoritme yang didukung

Algoritme kunci privat	Algoritme penandatanganan
ML_DSA_44	ML_DSA_44
ML_DSA_65	ML_DSA_65

Algoritme kunci privat	Algoritme penandatanganan
ML_DSA_87	ML_DSA_87
RSA_2048	SHA256WITHRSA
RSA_3072	SHA384WITHRSA
RSA_4096	SHA512WITHRSA
EC_prime256v1	SHA256DENGANECDSA
EC_secp384r1	SHA384DENGANECDSA
EC_secp521r1	SHA512DENGANECDSA
SM2 (Hanya Wilayah Tiongkok)	SM3WITHSM2

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**
>
> Untuk RSA atau ECDSA, keluarga algoritma penandatanganan yang ditentukan harus cocok dengan keluarga algoritma kunci kunci privat CA.
> Untuk ML-DSA, fungsi hash didefinisikan sebagai bagian dari algoritma itu sendiri. Tidak ada pilihan untuk memilih fungsi hash yang berbeda dengan ML-DSA. Untuk mempertahankan kompatibilitas mundur dengan APIs, nilai yang sama digunakan untuk algoritma kunci dan algoritma penandatanganan.

Kepatuhan RFC 5280 di AWS Private Certificate Authority

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 bertuliskan `Not After` tanggal lebih lambat dari 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 (pathLenConstraint) menentukan berapa banyak bawahan yang CAs 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 untukCAs.
- 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 OIDs; setiap pasangan termasuk issuerDomainPolicy dan asubjectDomainPolicy.
- 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 untuk AWS Private Certificate Authority

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.

Syarat dan konsep untuk AWS Private CA

Istilah dan konsep berikut dapat membantu Anda saat Anda bekerja dengannya AWS Private Certificate Authority.

Topik

- Percaya
- Sertifikat server TLS
- Tanda tangan sertifikat
- Otoritas sertifikat
- CA akar

- Sertifikat CA
- Sertifikat Root CA
- Sertifikat entitas akhir
- Sertifikat yang ditandatangani sendiri
- Sertifikat pribadi
- Jalur sertifikat
- Kendala panjang jalur

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 pribadi Anda membuat tanda tangan dengan menggunakan fungsi hash kriptografi seperti SHA256 di atas konten sertifikat berukuran 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 membangun toko kepercayaan untuk memasukkan hanya CAs yang mereka percayai. 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 oleh AWS 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 sertifikat (CA) biasanya ada dalam struktur hierarkis yang berisi beberapa lainnya CAs dengan hubungan orangtua-anak yang jelas di antara mereka. Anak atau bawahan CAs disertifikasi

oleh orang tua mereka CAs, 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 CA sertifikat 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 bawahan. CAs CA dengan batasan panjang jalur satu mungkin memiliki hingga satu tingkat bawahan di bawahnya. CAs 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.

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 https://console.aws.amazon.com/acm-pca/konsol.

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, lihat Panduan Pengguna AWS Certificate Manager.

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 hubungan Anda CAs dan mereka
- 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 root pada umumnya hanya boleh digunakan untuk menerbitkan sertifikat untuk perantara CAs. Hal ini memungkinkan CA root disimpan di luar bahaya sementara perantara CAs 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) 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 kendali atas akun Anda CAs . Kerugiannya adalah Anda tidak dapat menggunakan Konsol Manajemen AWS 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 sama Akun 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 CA menawarkan dua opsi standar untuk pencabutan terkelola: Protokol Status Sertifikat Online (OCSP), dan daftar pencabutan sertifikat (). CRLs Untuk informasi selengkapnya, lihat Rencanakan metode pencabutan AWS Private CA sertifikat Anda.

Nyalakan 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 Konsol Manajemen AWS, the AWS SDKs, the AWS Command Line Interface, dan layanan tingkat yang lebih tinggi AWS . 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 Pencatatan panggilan AWS Private Certificate Authority API menggunakan AWS 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 yang tidak digunakan CAs

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 Hapus CA pribadi Anda.

> **Note**
>
> AWS menagih Anda untuk CA sampai dihapus.

Blokir akses publik ke CRLs

AWS Private CA merekomendasikan penggunaan fitur Amazon S3 Block Public Access (BPA) pada bucket yang berisi. CRLs 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 Aktifkan 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. Amankan Kubernetes dengan AWS Private Certificate Authority

Gunakan AWS Private CA dengan AWS SDK untuk 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 alat baris perintah AWS SDKs dan untuk berinteraksi AWS 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 untuk 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 untuk Java di Blog AWS Pengembang.

AWS Private CA Contoh API

Contoh kode berikut menunjukkan cara menggunakan tindakan AWS Private CA API dan tipe data tertentu dengan AWS SDK untuk Java.

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 CAs dan sertifikat dengan nama subjek khusus
- Buat sertifikat dengan ekstensi khusus

Membuat dan mengaktifkan CA akar secara terprogram

Contoh Java ini menunjukkan cara mengaktifkan root CA menggunakan tindakan AWS Private CA API berikut:

- 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 algoritma penandatanganan yang digunakan CA untuk menandatangani Permintaan Penandatanganan Sertifikat sendiri, CRLs, dan tanggapan OCSP
- 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 NTAuth toko Enterprise Microsoft Active Directory (AD).

Operasi membuat otoritas sertifikat root pribadi (CA) menggunakan pengidentifikasi objek kustom (OIDs). 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 memberikan izin akses dari CA pribadi ke kepala AWS layanan 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.auth.AWSStaticCredentialsProvider;

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

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

public class DeletePolicy {

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

        // Retrieve your credentials from the C:\Users\name\.aws\credentials file
        // in Windows or the .aws/credentials file in Linux.
```

```
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load your credentials from file.",
 e);
        }

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

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

        // Create the request object.
        DeletePolicyRequest req = new DeletePolicyRequest();

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

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

```
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (AWSACMPCAException ex) {
            throw ex;
        }
    }
}
```

DescribeCertificateAuthority

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 dalam AWS 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 Gunakan templat AWS Private CA 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

Contoh Java berikut menunjukkan cara menggunakan ListCertificateAuthoritiesoperasi.

Operasi ini mencantumkan otoritas sertifikat pribadi (CAs) yang Anda buat menggunakan CreateCertificateAuthorityoperasi.

```
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. IzinIssueCertificate, termasukGetCertificate,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 Anda CAs. 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 diizinkan oleh kebijakan, kepala sekolah yang berada di AWS akun lain dapat menerbitkan dan memperbarui sertifikat entitas akhir pribadi menggunakan CA pribadi 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 CA menulis 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 AWS sumber daya 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 mereka CAs, Anda dapat menerapkan tag yang sama ke beberapa pribadi CAs. 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 CAs dan sertifikat dengan nama subjek khusus

CustomAttributeObjek memungkinkan administrator untuk meneruskan pengidentifikasi objek kustom (OIDs) ke CAs privat dan sertifikat. Kustom OIDs dapat digunakan untuk membuat hierarki nama subjek 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 AWS Private CA varietas template. Untuk informasi selengkapnya tentang penggunaan atraksi khusus, lihat dan. Menerbitkan sertifikat entitas akhir pribadi Buat CA pribadi di AWS Private CA

Anda tidak dapat menggunakan `StandardAttributes` dalam hubungannya dengan`CustomAttributes`. Namun, Anda dapat melewati standar OIDs sebagai bagian dari a`CustomAttributes`. Nama subjek default OIDs 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

Nama subjek	ID Objek
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);
    }
}
```

Buat sertifikat dengan ekstensi khusus

CustomExtensionObjek ini 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 AWS Private CA varietas template. Untuk informasi selengkapnya tentang menggunakan ekstensi kustom, lihatMenerbitkan sertifikat entitas akhir pribadi.

Topik

- Aktifkan CA bawahan dengan ekstensi NameConstraints
- Menerbitkan sertifikat dengan ekstensi pernyataan QC

Aktifkan CA bawahan dengan ekstensi NameConstraints

```
package com.amazonaws.samples;

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

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

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

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

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

```
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;

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

import lombok.SneakyThrows;

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

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

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

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

      // Define a certificate revocation list configuration.
      CrlConfiguration crlConfigure = new CrlConfiguration();
```

```
      crlConfigure.withEnabled(true);
      crlConfigure.withExpirationInDays(365);
      crlConfigure.withCustomCname(null);
      crlConfigure.withS3BucketName("your-bucket-name");

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

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

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

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

      // Create a client that you can use to make requests.
      AWSACMPCA client = AWSACMPCAClientBuilder.standard()
```

```
            .withEndpointConfiguration(endpoint)
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();

        return client;
    }

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

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

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

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

        return rootCertificate;
    }

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

        // Create the request object.
```

```
      CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
      createCARequest.withCertificateAuthorityConfiguration(configCA);
      createCARequest.withRevocationConfiguration(revokeConfig);
      createCARequest.withIdempotencyToken("1234");
      createCARequest.withCertificateAuthorityType(caType);

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

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

      return subordinateCAArn;
  }

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

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

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

```
        } catch(AWSACMPCAException e) {
            //Unexpected service exception.
        }

        // Retrieve the CSR.
        GetCertificateAuthorityCsrResult csrResult = null;
        try {
            csrResult = client.getCertificateAuthorityCsr(csrRequest);
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

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

        return csr;
    }

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

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

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

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

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

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

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

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

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

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

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

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

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

      return subordinateCertificateArn;
  }

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

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

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

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

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

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

      // Create waiter to wait on successful creation of the certificate file.
```

```
      Waiter<GetCertificateRequest> getCertificateWaiter =
client.waiters().certificateIssued();
      try {
          getCertificateWaiter.run(new WaiterParameters<>(certificateRequest));
      } catch (WaiterUnrecoverableException e) {
          //Explicit short circuit when the recourse transitions into
          //an undesired state.
      } catch (WaiterTimedOutException e) {
          //Failed to transition into desired state even after polling.
      } catch (AWSACMPCAException e) {
          //Unexpected service exception.
      }

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

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

      return subordinateCertificate;
  }

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

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

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

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

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

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

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

Menerbitkan 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);
    }
}
```

Gunakan AWS Private CA untuk mengimplementasikan sertifikat Matter

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.

Matter 1.2, dirilis pada Oktober 2023, mendukung pencabutan DAC menggunakan Daftar Pencabutan Sertifikat (). CRLs Untuk membantu Anda menyesuaikan dengan standar Matter saat ini, saat Anda mengaktifkan pencabutan CRL untuk masalah CAs itu, sertifikat Matter, di `CrlConfiguration` objek, dalam `CrlDistributionPointExtensionConfiguration` struktur, atur ke. `OmitExtension` `true`

Biasanya, CAs sematkan 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 CDPs dari Matter Distributed Compliance Ledger (DCL), penyimpanan data Matter tepercaya. Anda harus mengunggah URI CDP ke Matter DCL sehingga dapat ditemukan saat

memvalidasi. DACs Untuk informasi selengkapnya tentang menentukan URI CDP, lihatMenentukan URI Titik Distribusi CRL (CDP). Untuk informasi lebih lanjut tentang Materi, lihat halaman beranda standar Matter.

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 Root CACertificate _ 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 Memecahkan masalah kesalahan sertifikat yang sesuai dengan AWS Private CA 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 BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1definisi 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 Memecahkan masalah kesalahan sertifikat yang sesuai dengan AWS Private CA 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 Memecahkan masalah kesalahan sertifikat yang sesuai dengan AWS Private CA 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 Root CACertificate _ APIPassthrough / V1 template untuk membuat dan menginstal sertifikat Matter Root CA untuk diterbitkan NOCs. 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 Memecahkan masalah kesalahan sertifikat yang sesuai dengan AWS Private CA 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 BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1definisi template untuk menerbitkan dan menginstal sertifikat Matter Subordinate CA untuk diterbitkan NOCs. 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 Memecahkan masalah kesalahan sertifikat yang sesuai dengan AWS Private CA 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 Memecahkan masalah kesalahan sertifikat yang sesuai dengan AWS Private CA 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);
    }
}
```

Gunakan AWS Private CA untuk mengimplementasikan sertifikat mDL

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.

Topik

- Aktifkan sertifikat otoritas otoritas penerbitan (IACA)
- Buat sertifikat penandatangan dokumen

Aktifkan sertifikat otoritas otoritas penerbitan (IACA)

Contoh Java ini menunjukkan cara menggunakan BlankRootCACertificatePathLen_ APIPassthrough 0_/V1 definisi 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);
    }
}
```

Arsitek solusi Anda untuk AWS Private CA

AWS Private CA memberi Anda kontrol lengkap berbasis cloud atas PKI pribadi organisasi Anda (infrastruktur kunci publik), mulai dari otoritas sertifikat root (CA), melalui bawahan CAs, 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

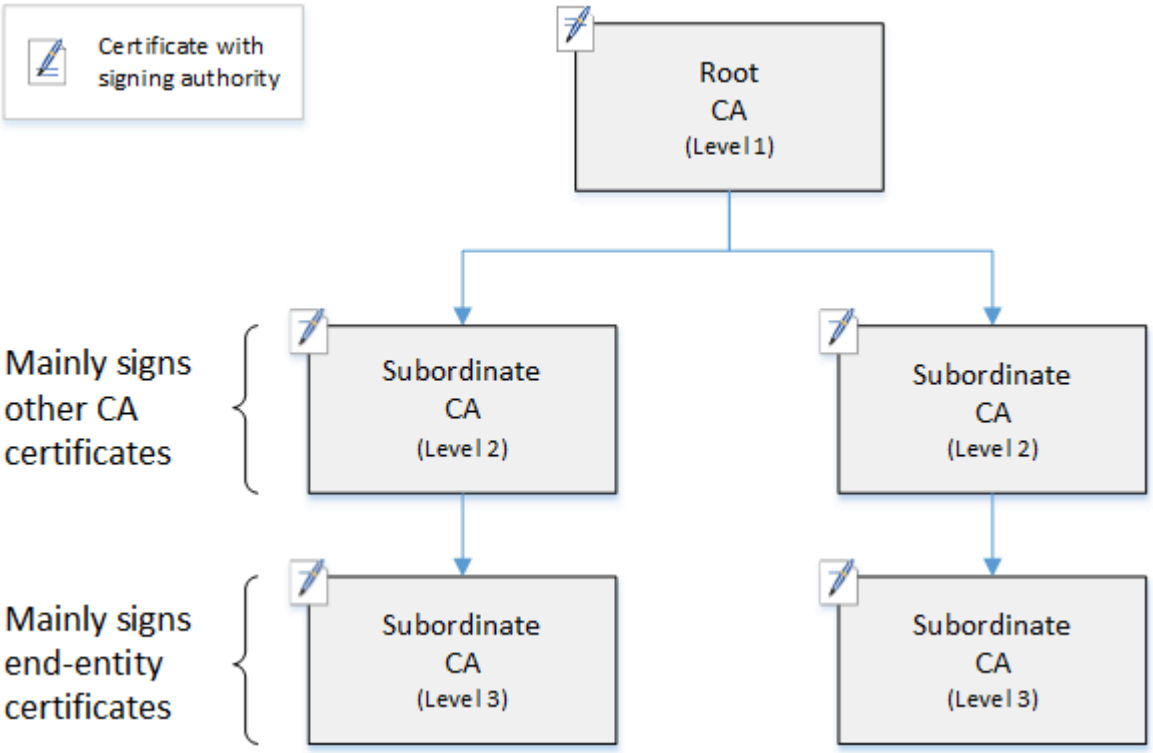
Rancang hierarki CA

Dengan AWS Private CA, Anda dapat membuat hierarki otoritas sertifikat hingga lima level. CA akar, di bagian atas pohon hierarki, dapat memiliki sejumlah cabang. Akar CA dapat memiliki sebanyak empat tingkat bawahan CAs pada 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 CAs dengan kepercayaan terbatas dan dapat dibatalkan untuk operasi harian
- 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 mereka dapat menandatangani sertifikat lain yang berada di bawah mereka. CAs 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. Ini CAs, pada gilirannya, menandatangani sertifikat untuk CAs 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. Root CA anchor dipercaya untuk semua bawahan CAs 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 bawahan tingkat tinggi CAs 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
>
> Root dan tingkat tinggi lainnya CAs membutuhkan proses operasional yang sangat aman dan protokol kontrol akses.

Topik

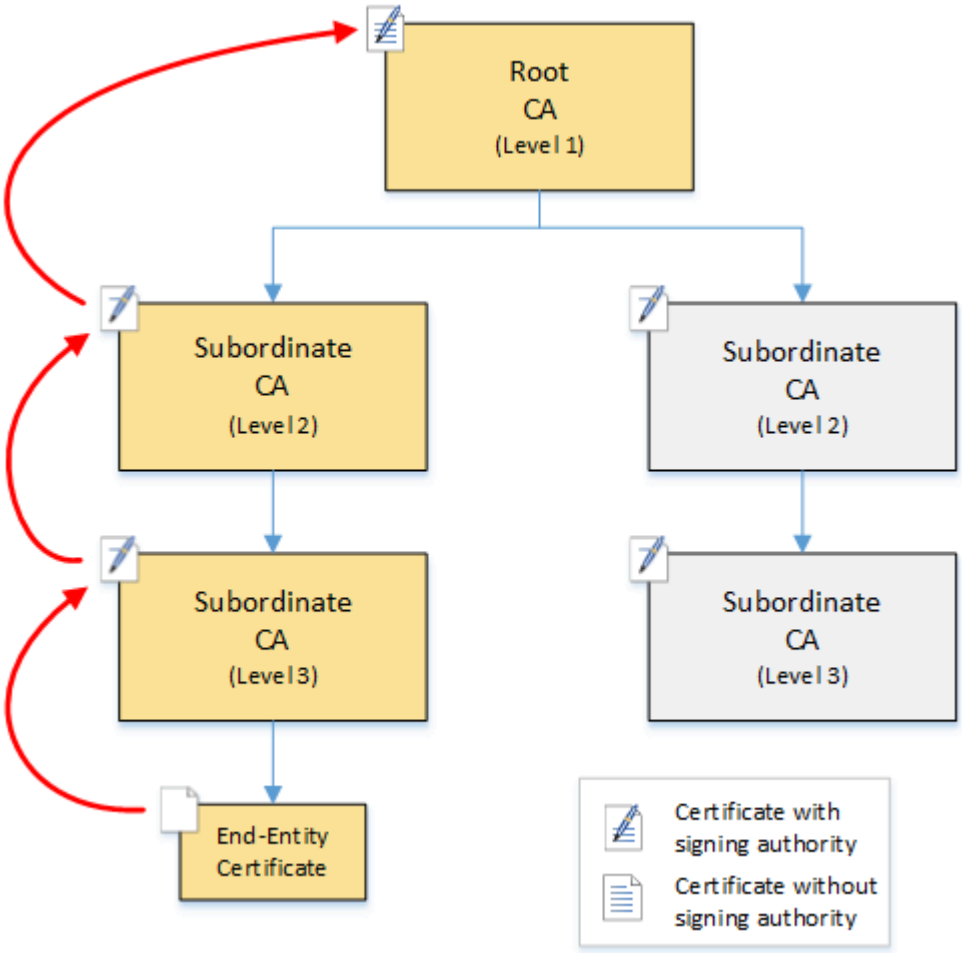
Validasi sertifikat entitas akhir

Sertifikat entitas akhir memperoleh kepercayaan mereka dari jalur sertifikasi yang mengarah kembali melalui bawahan CAs ke root CA. 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.

Toko kepercayaan adalah perpustakaan tepercaya CAs yang berisi browser 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 bawahan CAs 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.

Rencanakan 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, akar CA, yang tidak memiliki pathLenConstraintparameter, dapat mengotorisasi tingkat CAs bawahan yang tidak terbatas. CA bawahan dapat memiliki bawahan anak CAs 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 bawahan CAs
- Struktur hierarkis yang melindungi tingkat yang lebih tinggi CAs 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 bawahan CAs yang melaksanakan 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

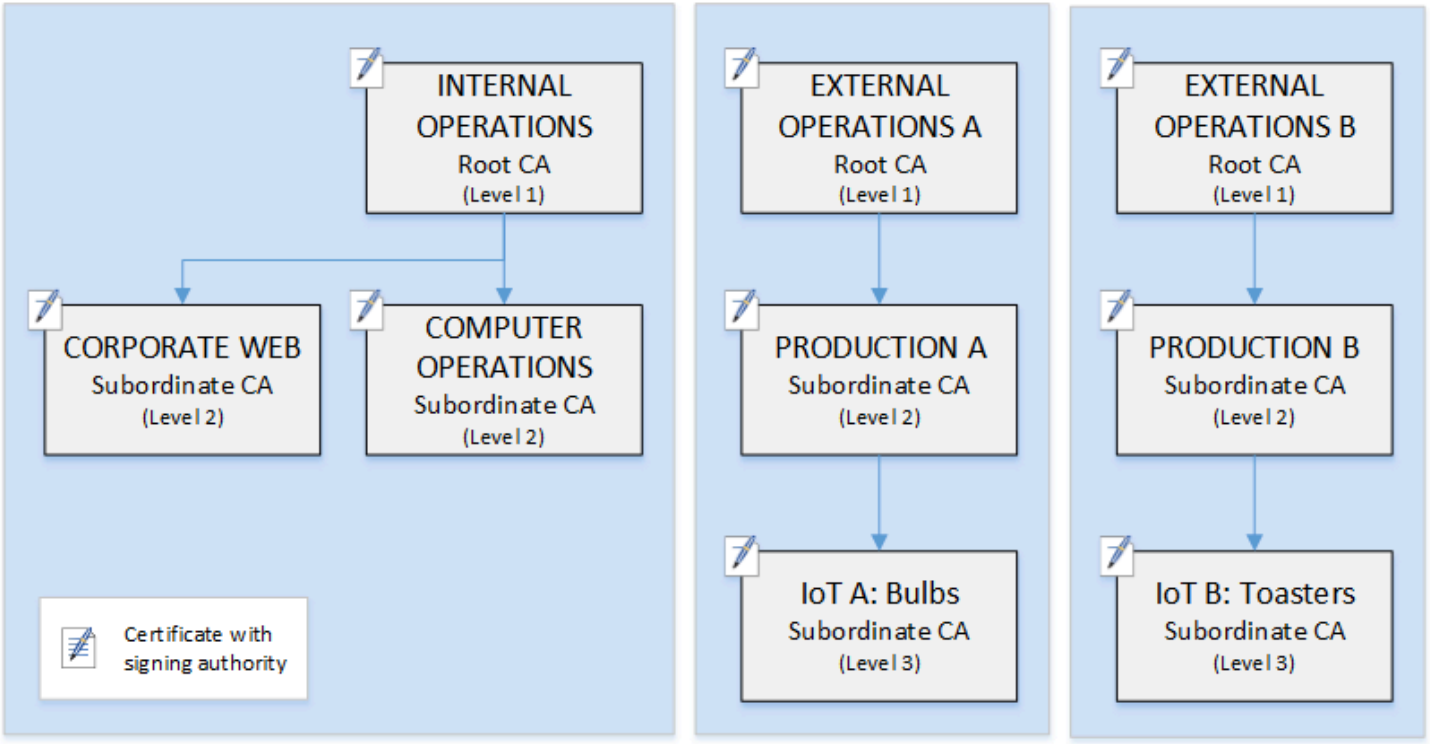
  Struktur ini biasanya digunakan untuk developer dan pengujian ketika rantai kepercayaan penuh tidak diperlukan. Digunakan dalam produksi, itu tidak lazim. Selain itu, ini melanggar praktik terbaik mempertahankan kebijakan keamanan terpisah untuk root CA dan CAs yang mengeluarkan sertifikat entitas akhir.

  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 root terpisah CAs 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, akar CAs memiliki tingkat isolasi yang tinggi dari level 3 CAs yang bertanggung jawab untuk penandatanganan massal sertifikat untuk ribuan atau jutaan barang manufaktur.

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.

Tetapkan batasan panjang pada jalur sertifikasi

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

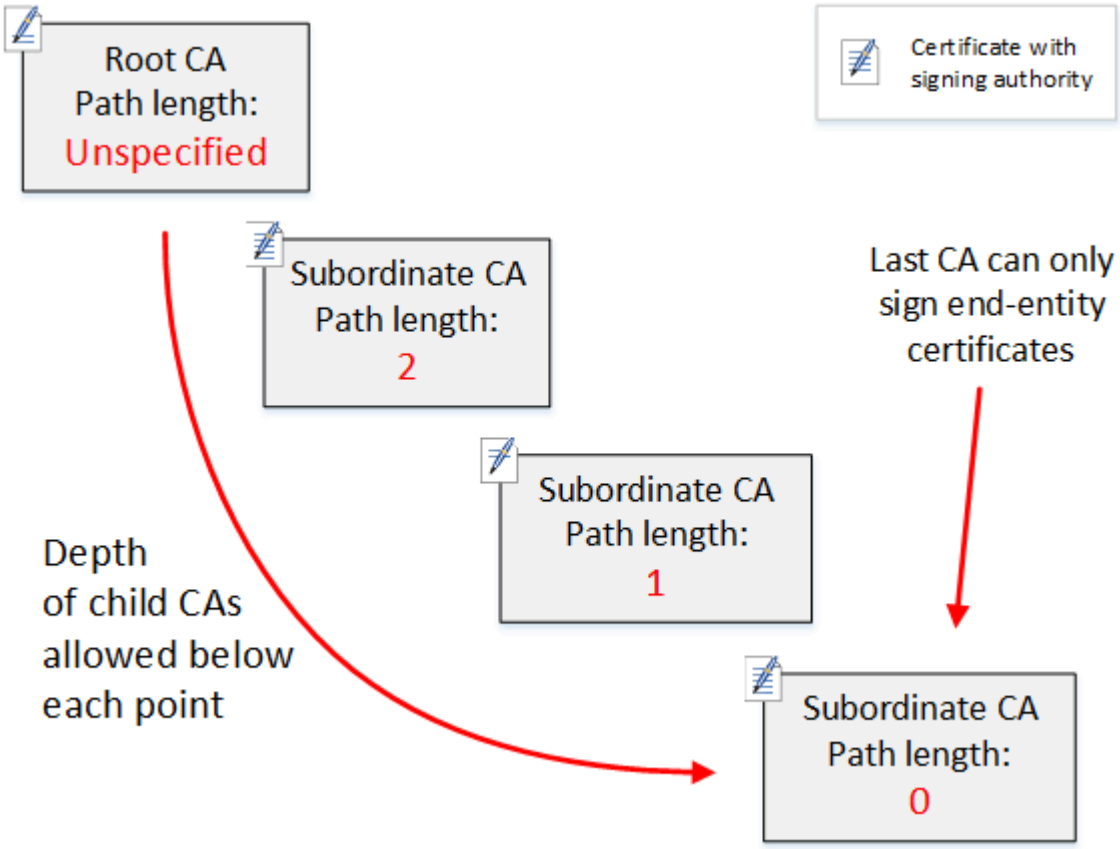
- `cA` – Apakah sertifikat menentukan CA. Jika nilai ini SALAH (default), maka sertifikat tersebut adalah sertifikat entitas akhir.
- `pathLenConstraint`— Jumlah maksimum bawahan tingkat bawah CAs yang dapat eksis dalam rantai kepercayaan yang valid. Sertifikat entitas akhir tidak dihitung karena bukan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

Bawahan CAs memiliki `pathLenConstraint` nilai yang sama dengan atau lebih besar dari nol, tergantung pada lokasi dalam penempatan hierarki dan fitur yang diinginkan. Misalnya, dalam hierarki dengan tiga CAs, tidak ada batasan jalur yang ditentukan untuk root CA. CA bawahan pertama memiliki panjang jalur 1 dan karenanya dapat menandatangani anak CAs. Masing-masing anak ini CAs harus memiliki `pathLenConstraint` nilai nol. Ini berarti bahwa mereka dapat menandatangani sertifikat entitas akhir, tetapi tidak dapat menerbitkan sertifikat CA tambahan. Membatasi kekuatan untuk menciptakan yang baru CAs 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 `pathLenConstraint` nilai 2, yang membatasi anaknya untuk melangkah lebih CAs dari dua tingkat lebih dalam. 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.

Kelola panjang jalur dengan templat

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:

- CACertificateAkar/V1
- Bawahan CACertificate _ 0/V1 PathLen
- Bawahan CACertificate _ 1/V1 PathLen
- Bawahan CACertificate _ 2/V1 PathLen

- Bawahan CACertificate _ 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 Gunakan templat AWS Private CA sertifikat.

Otomatiskan pengaturan 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 CloudFormation .

Mengelola siklus hidup CA pribadi

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

Pilih periode validitas

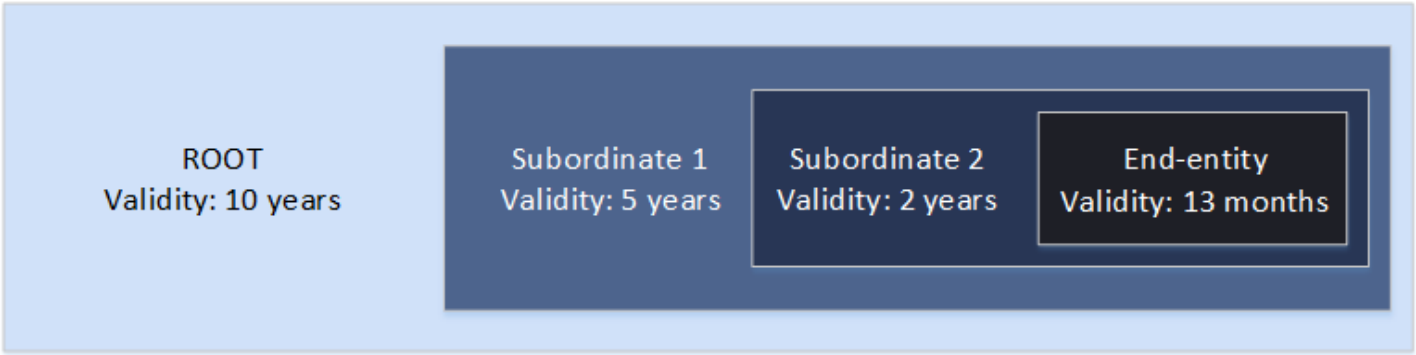
Masa berlaku sertifikat X.509 adalah bidang sertifikat dasar yang diperlukan. Menentukan rentang waktu saat menerbitkan CA yang disertifikasi yang menyatakan bahwa sertifikat dapat dipercaya, pembatasan pencabutan. (Sertifikat akar, yang ditandatangani sendiri, menyatakan masa berlakunya sendiri.)

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, sertifikat anak CAs dan entitas akhir tidak dapat bertahan 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 kisaran bawahan CAs antara 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.

Kelola suksesi CA

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

Ganti 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 sertifikat lama yang dikeluarkan dari CA, dan, jika dikonfigurasi untuk melakukannya, ia terus memvalidasi sertifikat melalui daftar pencabutan sertifikat OCSP and/or (). CRLs 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 bawahan CAs, Anda juga harus menggantinya, karena bawahan CAs kedaluwarsa pada saat yang sama atau sebelum CA orang tua mereka. Mulailah dengan mengganti CA tertinggi dalam hierarki yang perlu diganti. Kemudian buat bawahan pengganti baru CAs di setiap tingkat bawah berikutnya.

AWS merekomendasikan agar Anda menyertakan pengenal generasi CA dalam nama sesuai CAs 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 CAs keduanya 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 sebanding dengan frekuensi penggunaan kunci: CAs yang mengeluarkan lebih banyak sertifikat harus diputar 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 dalamMemasang 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.

Cabut 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.

Rencanakan metode pencabutan AWS Private CA sertifikat Anda

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 (CRLs), atau keduanya.

> **Note**
>
> Jika Anda membuat CA tanpa mengonfigurasi pencabutan, Anda selalu dapat mengonfigurasinya nanti. Untuk informasi selengkapnya, lihat Perbarui CA pribadi di AWS Private Certificate Authority.

- 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 yang baru atau yang sudah ada CAs dengan satu operasi menggunakan AWS Private CA konsol, API, CLI, atau melalui. CloudFormation Sedangkan CRLs disimpan dan diproses pada titik akhir dan dapat menjadi basi, persyaratan penyimpanan dan pemrosesan OCSP 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. CAs dibuat di AWS Private CA konsol menggunakan algoritma tanda tangan SHA256 WITHRSA secara default. Algoritma lain yang didukung dapat ditemukan di dokumentasi CertificateAuthorityConfigurationAPI.
  - APIPassthrough dan templat CSRPassthrough sertifikat 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 () CRLs

  Daftar pencabutan sertifikasi (CRL) adalah file yang berisi daftar sertifikat yang dicabut sebelum tanggal kedaluwarsa yang dijadwalkan. CRL berisi daftar sertifikat yang seharusnya tidak lagi dipercaya, alasan pencabutan, dan informasi relevan lainnya.

  Saat mengonfigurasi otoritas sertifikat (CA), Anda dapat memilih apakah AWS Private CA membuat CRL lengkap atau terpartisi. Pilihan Anda menentukan jumlah maksimum sertifikat yang dapat diterbitkan dan dicabut oleh otoritas sertifikat. Untuk informasi lebih lanjut, lihat AWS Private CA kuota.

  Pertimbangan CRL

  - Pertimbangan memori dan bandwidth: CRLs membutuhkan lebih banyak memori daripada OCSP karena persyaratan unduhan dan pemrosesan lokal. Namun, CRLs mungkin mengurangi bandwidth jaringan dibandingkan dengan OCSP dengan caching daftar pencabutan alih-alih memeriksa status per koneksi. Untuk perangkat yang dibatasi memori, seperti perangkat IoT tertentu, pertimbangkan untuk menggunakan partisi. CRLs
  - Mengubah jenis CRL: Saat mengubah dari CRL lengkap ke partisi, AWS Private CA buat partisi baru sesuai kebutuhan dan tambahkan ekstensi IDP ke semua, termasuk yang asli. CRLs Mengubah dari dipartisi untuk menyelesaikan pembaruan hanya satu CRL dan mencegah pencabutan sertifikat di masa depan yang terkait dengan partisi sebelumnya.

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

Topik

- Siapkan CRL untuk AWS Private CA
- Kustomisasi URL OCSP untuk AWS Private CA

Siapkan CRL untuk AWS Private CA

Sebelum Anda dapat mengonfigurasi daftar pencabutan sertifikat (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 Certificate revocation options dan. Kustomisasi URL OCSP untuk AWS Private CA

Topik

Jenis CRL

- Selesai - Pengaturan default. AWS Private CA memelihara file CRL tunggal yang tidak dipartisi untuk semua sertifikat yang belum kedaluwarsa yang dikeluarkan oleh CA yang telah dicabut. Setiap sertifikat yang AWS Private CA diterbitkan terikat pada CRL tertentu melalui ekstensi CRL distribution point (CDP), sebagaimana didefinisikan dalam RFC 5280. Anda dapat memiliki hingga 1 juta sertifikat pribadi untuk setiap CA dengan CRL lengkap diaktifkan. Untuk informasi lebih lanjut, lihat AWS Private CA kuota.
- Dipartisi - Dibandingkan dengan yang lengkap CRLs, dipartisi CRLs secara dramatis meningkatkan jumlah sertifikasi yang dapat dikeluarkan CA pribadi Anda, dan menyelamatkan Anda dari sering memutar. CAs

> **Important**
>
> Saat menggunakan partisi CRLs, Anda harus memvalidasi bahwa URI titik distribusi penerbitan terkait (IDP) CRL cocok dengan URI CDP sertifikasi untuk memastikan CRL yang tepat telah diambil. AWS Private CA menandai ekstensi IDP sebagai hal yang penting, yang harus dapat diproses oleh klien Anda.

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
```

CRLs memiliki format 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 CRLs 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.

> **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 CRLs sebagai metode pencabutan sertifikat, AWS Private CA buat CRL dan terbitkan 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 menerapkan kebijakan yang kurang permisif seperti berikut ini, yang membatasi akses ke AWS akun tertentu dan CA pribadi tertentu. Atau, Anda dapat menggunakan kunci kondisi aws: SourceOrg ID untuk membatasi akses ke organisasi tertentu di AWS Organizations. Untuk informasi selengkapnya tentang kebijakan bucket, lihat Kebijakan Bucket untuk Amazon Simple Storage Service.

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

Aktifkan 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 CRLs di internet publik (yaitu, saat tidak masuk ke AWS akun). Bagian ini menjelaskan cara mengonfigurasi solusi PKI pribadi menggunakan Amazon CloudFront, jaringan pengiriman konten (CDN), untuk melayani CRLs 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 CloudFront untuk BPA

Buat CloudFront distribusi yang akan memiliki akses ke bucket S3 pribadi Anda, dan dapat melayani CRLs 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. 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 Buat CA pribadi di AWS Private CA.

  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.

Menentukan URI Titik Distribusi CRL (CDP)

Jika Anda perlu menggunakan URI Titik Distribusi CRL (CDP) di alur kerja Anda, Anda dapat mengeluarkan sertifikat menggunakan URI CRL pada sertifikat tersebut atau menggunakan

metode berikut. Ini hanya berfungsi untuk lengkap CRLs. Dipartisi CRLs memiliki GUID acak yang ditambahkan padanya.

Jika Anda menggunakan bucket S3 sebagai CRL Distribution Point (CDP) untuk CA Anda, URI CDP dapat berada dalam salah satu format berikut.

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

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

Secara default, AWS Private CA tulis ekstensi CDP menggunakan titik akhir regional, IPv4 - only`amazonaws.com`. Untuk menggunakan CRLs over IPv6, lakukan salah satu langkah berikut sehingga CDPs ditulis dengan titik URLs itu ke titik akhir dualstack S3:

- Tetapkan nama kustom CRL Anda ke domain endpoint dualstack S3. Sebagai contoh, `bucketname.s3.dualstack.region-code.amazonaws.com`.
- Siapkan catatan DNS CNAME Anda sendiri yang menunjuk ke titik akhir dualstack S3 yang relevan, lalu gunakan sebagai nama kustom CRL Anda

Kustomisasi URL OCSP untuk AWS Private CA

> **Note**
>
> Topik ini ditujukan untuk pelanggan yang ingin menyesuaikan URL publik dari titik akhir responder Online Certificate Status Protocol (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.

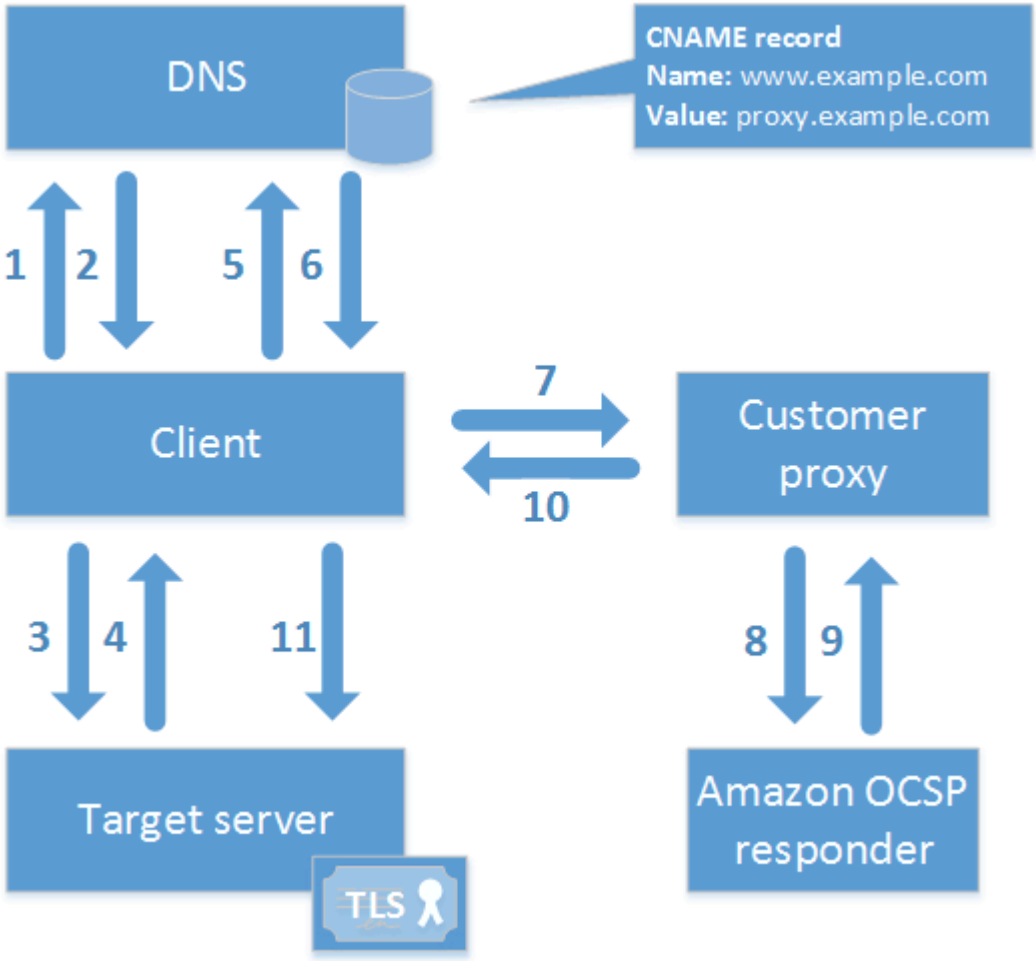
Secara default, saat Anda mengaktifkan OCSP AWS Private CA, setiap sertifikat yang Anda terbitkan berisi URL untuk responden AWS OCSP. Hal ini memungkinkan klien meminta koneksi kriptografis aman untuk mengirim kueri validasi OCSP langsung ke. AWS Namun, dalam beberapa kasus, mungkin lebih baik untuk 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 yang 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 dalamBuat CA pribadi di AWS Private CA.
- 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 Buat CA pribadi di AWS Private CA.
- 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.
>   - Tetapkan `ocsp.acm-pca.<region>.amazonaws.com` sebagai asal.
>   - Untuk menggunakan IPv6 koneksi, gunakan titik akhir dualstack `acm-pca-ocsp.<region>.api.aws`
>   - 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

Menggunakan OCSP di atas IPv6

URL responder AWS Private CA OCSP default adalah IPv4 -only. Untuk menggunakan OCSP over IPv6, konfigurasikan URL OCSP kustom untuk CA Anda. URL dapat berupa:

- FQDN dari responder PCA OCSP dualstack, yang mengambil bentuk `acm-pca-ocsp.region-name.api.aws`
- Catatan CNAME yang telah Anda konfigurasikan untuk menunjuk pada responder OCSP dualstack, seperti yang dijelaskan di atas.

Memahami mode AWS Private CA CA

AWS Private CA mendukung pembuatan otoritas sertifikat (CA) di salah satu dari dua mode. Mode, sertifikat tujuan umum dan berumur pendek, memengaruhi masa berlaku yang diizinkan dari sertifikat yang dikeluarkan oleh CA.

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

Tujuan umum (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.

Sertifikat berumur pendek

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.

Sertifikat berumur pendek harus menjadi CA terakhir dalam hierarki sertifikat. Ada biaya overhead yang signifikan karena CA pribadi harus diperbarui setiap tujuh hari.

CAs dengan mode sertifikat berumur pendek biaya kurang dari tujuan umum CAs. Untuk informasi selengkapnya, silakan lihat Harga AWS Private Certificate Authority.

Untuk membuat CA yang mengeluarkan sertifikat berumur pendek, tetapkan `UsageMode` parameter ke sertifikat berumur pendek menggunakan prosedur create a CA untuk membuat CA.

> **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

Rencanakan ketahanan di AWS Private CA

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 redundan CAs 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 root CAs 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 root CAs redundan meningkatkan kompleksitas operasional: Anda perlu mendistribusikan kedua sertifikat root CA ke toko kepercayaan browser dan sistem operasi di lingkungan Anda.
2. Anda juga dapat membuat bawahan redundan CAs 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.

Otoritas sertifikat di AWS Private CA

Dengan menggunakan AWS Private Certificate Authority, Anda dapat membuat hierarki root dan subordinate certificate authority (CAs) yang sepenuhnya AWS dihosting untuk penggunaan internal oleh organisasi Anda. Untuk mengelola pencabutan sertifikat, Anda dapat mengaktifkan Protokol Status Sertifikat Online (OCSP), daftar pencabutan sertifikat (CRLs), atau keduanya. AWS Private CA menyimpan dan mengelola sertifikat CA Anda, CRLs, dan tanggapan OCSP, dan kunci pribadi untuk otoritas root Anda disimpan dengan aman oleh. 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 Konsol Manajemen AWS, 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 Gunakan AWS Private CA dengan AWS SDK untuk Java.

Setelah Anda membuat CA pribadi aktif dan mengkonfigurasi akses ke sana, Anda dapat mengeluarkan dan mengambil sertifikat, seperti yang dijelaskan dalamMenerbitkan dan mengelola sertifikat di AWS Private CA.

Topik

- Perbarui CA pribadi di AWS Private Certificate Authority
- Hapus CA pribadi Anda
- Kembalikan CA pribadi
- Gunakan sertifikat CA pribadi yang ditandatangani secara eksternal

Siapkan untuk digunakan AWS Private CA

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.

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/pendaftaran.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada 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. Kapan saja, Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan masuk https://aws.amazon.comke/ 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 Konsol Manajemen AWSsebagai 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.

Buat CA pribadi di AWS Private CA

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

Untuk informasi tentang memperbarui konfigurasi CA yang telah Anda buat, lihatPerbarui CA pribadi di AWS Private Certificate Authority.

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

- Contoh CLI untuk membuat CA pribadi

Console

Untuk membuat CA privat menggunakan konsol

1. Selesaikan langkah-langkah berikut untuk membuat CA pribadi menggunakan file Konsol Manajemen AWS.

   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 pribadi CAs, 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 Anda CAs. Pilih Buat CA.

2. Di bawah Opsi mode, pilih mode kedaluwarsa sertifikat yang diterbitkan 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.

3. 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. Ini berfungsi sebagai otoritas penandatanganan utama untuk sertifikat entitas lain CAs dan akhir dalam hierarki.
   - Memilih Bawahan menciptakan CA yang harus ditandatangani oleh CA induk di atasnya dalam hierarki. Bawahan CAs biasanya digunakan untuk membuat bawahan lain CAs 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 Instal sertifikat CA bawahan yang ditandatangani oleh CA induk eksternal.

4. 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 APIPassthrough templat pada saat penerbitan. Untuk informasi lebih lanjut dan contoh terperinci, lihatMenerbitkan sertifikat dengan nama subjek kustom menggunakan APIPassthrough templat.

   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.

5.

Di bawah opsi Algoritma kunci, pilih algoritma kunci dan kekuatan algoritma. Nilai defaultnya adalah RSA 2048. Anda dapat memilih dari algoritma berikut:

- ML-DSA-44
- ML-DSA-65
- ML-DSA-87
- RSA 2048
- RSA 3072
- RSA 4096
- ECDSA P256
- ECDSA P384
- ECDSA P521

6. 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 Rencanakan metode pencabutan AWS Private CA sertifikat Anda 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 Perbarui CA pribadi di AWS Private Certificate Authority.

   Untuk mengonfigurasi opsi pencabutan Sertifikat, lakukan langkah-langkah berikut.

   a. Di bawah opsi pencabutan sertifikat, pilih Aktifkan distribusi CRL.

b. Di bawah URI bucket S3, pilih bucket yang sudah ada dari daftar.

Saat 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 CRLs. Gunakan salah satu pola kebijakan yang dijelaskan dalam Kebijakan akses untuk CRLs di Amazon S3. Untuk informasi selengkapnya, lihat Menambahkan kebijakan bucket menggunakan konsol Amazon S3.

c. Perluas pengaturan CRL untuk opsi konfigurasi tambahan.

- Pilih Aktifkan partisi untuk mengaktifkan partisi. CRLs Jika Anda tidak mengaktifkan partisi, CA Anda tunduk pada jumlah maksimum sertifikat yang dicabut. Untuk informasi lebih lanjut, lihat AWS Private Certificate Authority kuota. Untuk informasi selengkapnya tentang dipartisi CRLs, lihat tipe CRL.
- 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. Untuk menggunakan CRLs over IPv6, setel ini ke titik akhir S3 dualstack bucket Anda seperti yang dijelaskan dalam Using over. CRLs IPv6
- Tambahkan jalur Kustom untuk membuat alias DNS untuk jalur file di bucket Amazon S3 Anda.
- Ketik Validitas dalam beberapa hari CRL Anda akan tetap valid. Nilai default-nya adalah 7 hari. Untuk online CRLs, masa berlaku 2-7 hari adalah umum. AWS Private CA mencoba meregenerasi CRL pada titik tengah periode yang ditentukan.

7. Untuk opsi pencabutan Sertifikat, pilih Aktifkan OCSP.

- Di bidang endpoint OCSP Kustom - opsional, Anda dapat memberikan nama domain yang memenuhi syarat (FQDN) untuk titik akhir OCSP non-Amazon. Untuk menggunakan OCSP di atas IPv6, atur bidang ini ke titik akhir dualstack seperti yang dijelaskan dalam Menggunakan OCSP over. IPv6

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. Kustomisasi URL OCSP untuk AWS Private CA

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

Nama catatan	Tipe	Kebijakan perutean	Pembeda	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://”.

8. 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 CAs setelah pembuatan, lihatTambahkan 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.

9. 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.

10. 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.

11. 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:
    >
    > - Instal sertifikat CA root

- Instal sertifikat CA bawahan yang dihosting oleh AWS Private CA
- Instal sertifikat CA bawahan yang ditandatangani oleh CA induk eksternal

CLI

Gunakan create-certificate-authorityperintah untuk membuat CA pribadi. Anda harus menentukan konfigurasi CA (berisi algoritma dan informasi nama subjek), konfigurasi pencabutan (jika Anda berencana untuk menggunakan OCSP CRL), dan and/or 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 CRLs di Amazon S3.

File konfigurasi CA menentukan informasi berikut:

- Nama algoritme
- Ukuran kunci yang akan digunakan untuk membuat kunci privat CA
- Jenis algoritma penandatanganan yang digunakan CA untuk menandatangani Permintaan Penandatanganan Sertifikat sendiri, CRLs, dan tanggapan OCSP
- 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 Aktifkan 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”.
- (Opsional) Jalur khusus untuk CRL di bucket S3.
- (Opsional) CrlTypeNilai yang menentukan apakah CRL akan lengkap atau dipartisi. Jika tidak disediakan, CRL akan default untuk menyelesaikan.

> **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)

Lihat bagian berikut untuk contoh CLI.

Contoh CLI untuk membuat CA pribadi

Contoh berikut mengasumsikan bahwa Anda telah menyiapkan direktori `.aws` konfigurasi dengan Region, endpoint, dan kredensial default yang valid. Untuk informasi tentang mengonfigurasi AWS CLI lingkungan Anda, lihat Konfigurasi dan pengaturan file kredenal. 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. Kustomisasi URL OCSP untuk AWS Private CA

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

Nama catatan	Tipe	Kebijakan perutean	Pembeda	Nilai/Rutekan lalu lintas ke
alternati f.example.com	CNAME	Sederhana	-	proxy.exa mple.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":"amzn-s3-demo-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": "amzn-s3-demo-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":"amzn-s3-demo-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": "amzn-s3-demo-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 NTAuth toko Enterprise 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 NTAuth toko Perusahaan.

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 CRLs, dan mengonfigurasi CA untuk menghilangkan URL CDP default dari sertifikat yang dikeluarkan.

Berkas: revoke_config.txt

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

Memasang 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 Pribadi CAs.

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

  Warisan Wilayah AWS meliputi:

Nama wilayah	Lokasi geografis

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	Africa (Cape Town)
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)

Nama wilayah	Lokasi geografis
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.
- Di setiap Wilayah AWS, aturan berikut berlaku untuk EDCSA:
  - CA induk dengan algoritma penandatanganan EC_Secp521R1 dapat mengeluarkan sertifikat dengan ECDSA P521.

Instal sertifikat CA root

Anda dapat menginstal sertifikat CA root dari Konsol Manajemen AWS 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/rumah. 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 lebih lanjut, lihatAlgoritma penandatanganan yang kompatibel,Algoritma kriptografi yang didukung di AWS Private Certificate Authority, dan SigningAlgorithmmasuk CertificateAuthorityConfiguration.
     - 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": "amzn-s3-demo-bucket"
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

Instal sertifikat CA bawahan yang dihosting oleh AWS Private CA

Anda dapat menggunakan Konsol Manajemen AWS 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/rumah. 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 ditampilkan CAs 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 CAs bawahan tambahan untuk itu.
   - 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 Gunakan templat AWS Private CA 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.

Instal sertifikat CA bawahan yang ditandatangani oleh CA induk eksternal

Setelah Anda membuat CA pribadi bawahan seperti yang dijelaskan dalamBuat CA pribadi di AWS Private CA, Anda memiliki opsi untuk 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 Gunakan sertifikat CA pribadi yang ditandatangani secara eksternal.

Kontrol akses ke CA pribadi

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
- Lampirkan kebijakan untuk akses lintas akun

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) AWS Private Certificate Authority

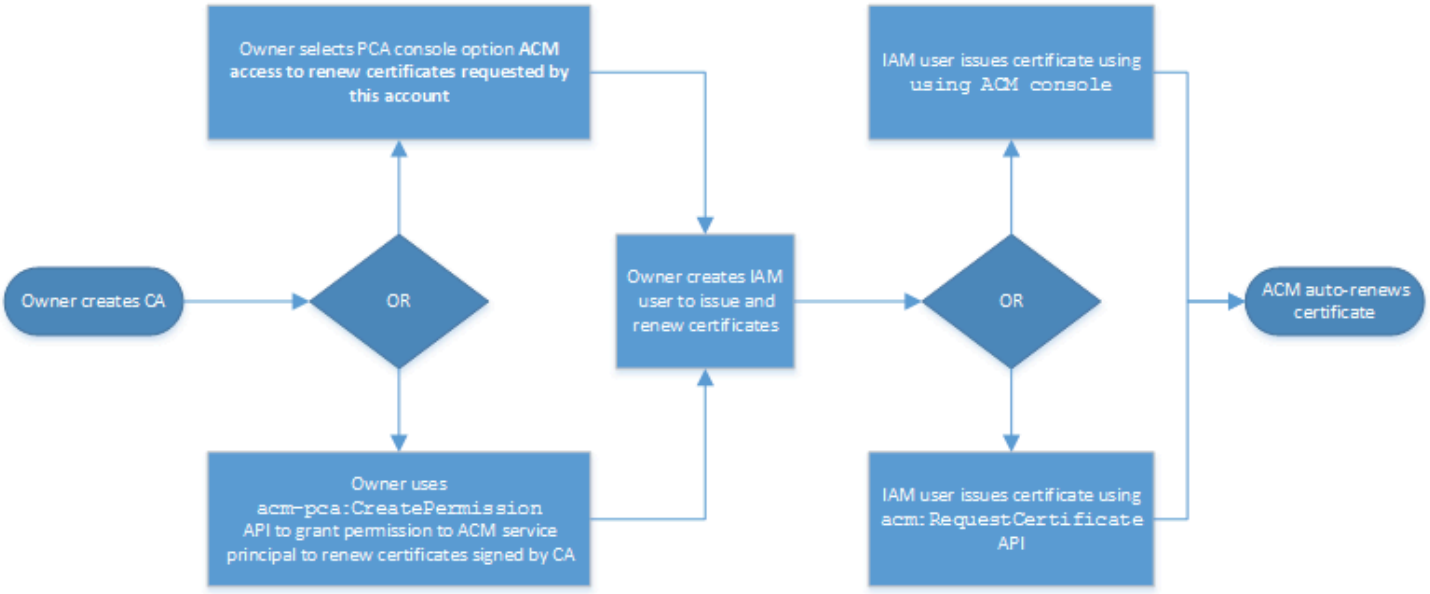
Kasus akun tunggal 1: Menerbitkan sertifikat yang tidak dikelola

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/rumah.
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.

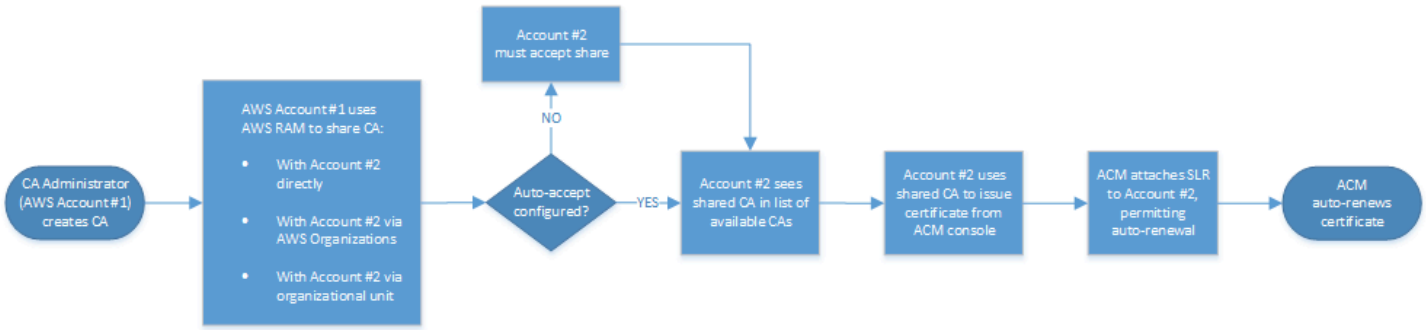
> **Note**
>
> Anda dapat dengan mudah membuka konsol RAM dengan memilih CA dan kemudian memilih Tindakan, Kelola berbagi sumber daya.

- Secara terprogram, menggunakan PCA APIs 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.



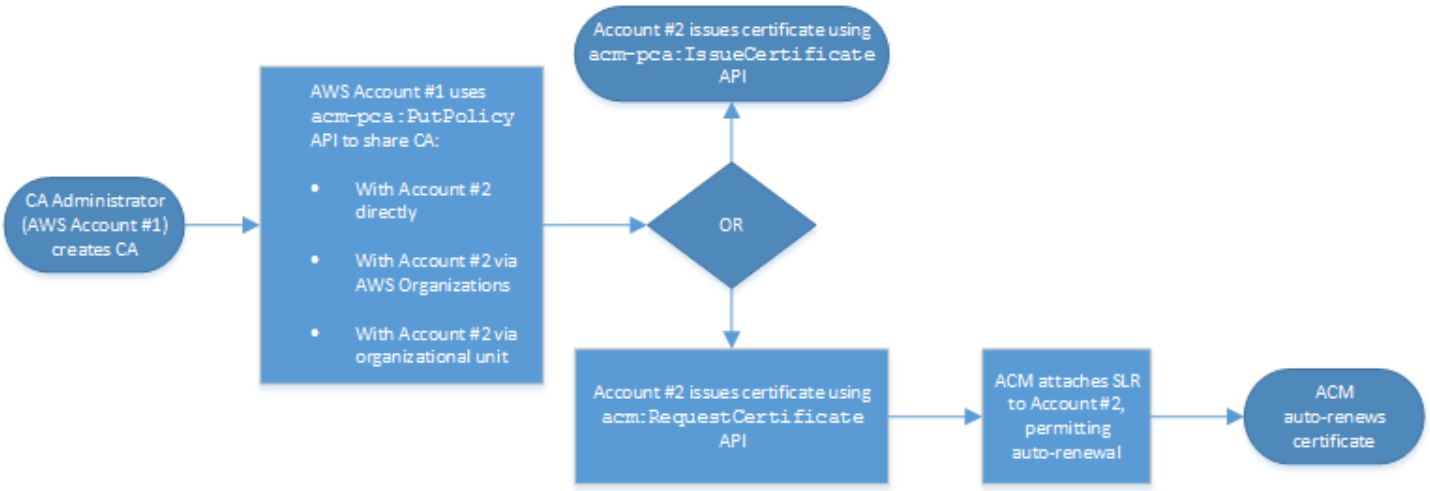
Setelah RAM berbagi sumber daya AWS Organizations, prinsipal penerima harus menerima sumber daya agar dapat diterapkan. Penerima dapat mengonfigurasi AWS Organizations untuk menerima saham yang ditawarkan secara otomatis.

> **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.

Daftar pribadi CAs

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

Untuk daftar yang tersedia CAs menggunakan konsol

1. Masuk ke AWS akun Anda dan buka AWS Private CA konsol di https://console.aws.amazon.com/acm-pca/rumah.
2. Tinjau informasi dalam daftar otoritas sertifikat swasta. Anda dapat menavigasi melalui beberapa halaman CAs 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 ML_DSA_44, ML_DSA_65, ML_DSA_87, RSA_2048, RSA_3072, RSA_4096, EC_Prime256v1, EC_secp384R1, dan EC_secp521R1.
- 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 ML_DSA_44, ML_DSA_65, ML_DSA_87, WITHRSA, WITHRSA, WITHRSA, WITHECDSA, WITHECDSA, dan WITHECDSA. SHA256 SHA384 SHA512 SHA256 SHA384 SHA512

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

Untuk daftar yang tersedia CAs menggunakan AWS CLI

Gunakan list-certificate-authoritiesperintah untuk daftar yang tersedia CAs 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
            }
         }
      }
 ...
   ]
```

```
}
```

Lihat 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 memperbaruiCAs, lihatPerbarui CA pribadi di AWS Private Certificate Authority.

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/rumah.
2. Tinjau daftar otoritas sertifikat swasta. Anda dapat menavigasi melalui beberapa halaman CAs 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 Subject distinguished name. 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 ML-DSA-44, ML-DSA-65, ML-DSA-87, RSA 2048, RSA 3072, RSA 4096, ECDSA P256, ECDSA P384, dan ECDSA P521.
   - Algoritma penandatanganan — Algoritma yang digunakan CA untuk menandatangani Permintaan Penandatanganan Sertifikat sendiri CRLs,, dan tanggapan OCSP (Jangan bingung dengan `SigningAlgorithm` parameter yang digunakan dalam IssueCertificate

API.) Nilai yang mungkin adalah ML-DSA-44, ML-DSA-65, ML-DSA-87, RSA, RSA, dan RSA, ECDSA, ECDSA, SHA256 ECDSA SHA384 SHA512 SHA256 SHA384 SHA512

- Standar keamanan penyimpanan utama — Tingkat kesesuaian Standar Pemrosesan Informasi Federal (FIPS). Anda dapat memilih dari nilai-nilai ini: FIPS 140-2 level 2 atau lebih tinggi, FIPS 140-2 level 3 atau lebih tinggi, dan CCPC Level 1 atau lebih tinggi. Parameter ini bervariasi menurut AWS Wilayah.

> **Note**
>
> Mulai 26 Januari 2023, AWS Private CA melindungi semua kunci pribadi CA di wilayah non-China menggunakan modul keamanan perangkat keras (HSMs) yang sesuai dengan FIPS PUB 140-2 Level 3.

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).

Tambahkan 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 pribadi CAs.

Anda dapat menambahkan atau menghapus tag khusus untuk CA pribadi Anda kapan saja. Misalnya, Anda dapat menandai privat CAs dengan pasangan kunci-nilai 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 EC2 Sumber Daya Amazon Anda di EC2 Panduan Pengguna Amazon.

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 terhadap 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 Konsol Manajemen AWS.

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/rumah.
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 tag-certificate-authorityperintah untuk menambahkan tag ke CA pribadi 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 untag-certificate-authorityperintah untuk menghapus tag dari CA pribadi.

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

Memahami status AWS Private CA 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 CRLs	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
`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`.						Tidak

Status	Sertifikat penerbitan	Validasi sertifikat dengan OCSP	Menghapus CRLs	Hasilkan audit	Anda dapat memperbarui sertifikat CA	Sertifikat dapat dicabut	Anda ditagih untuk CA
	• Untuk informasi lebih lanjut tentang penghapusan CA, lihat Hapus CA pribadi Anda.						

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 Memasang sertifikat CA.

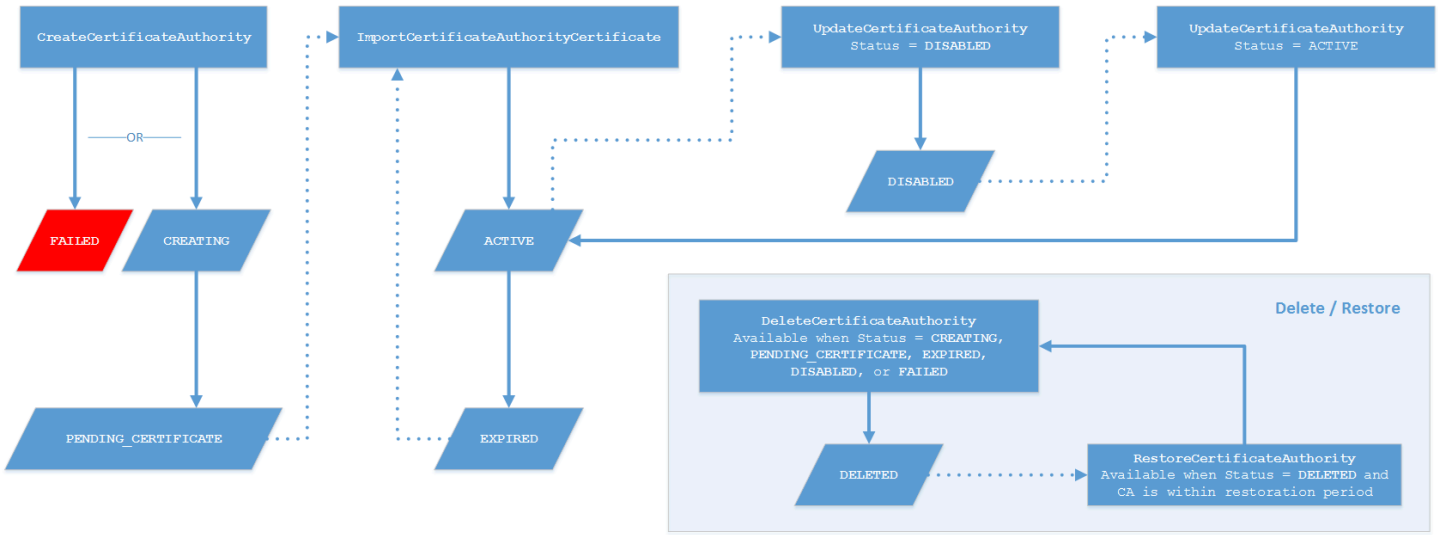
Anda tidak dapat langsung mengubah status CA yang kedaluwarsa. Jika Anda mengimpor sertifikat baru untuk CA, AWS Private CA setel ulang statusnya `ACTIVE` kecuali telah disetel ke `DISABLED` sebelum sertifikat kedaluwarsa.

Pertimbangan tambahan tentang sertifikat CA yang kedaluwarsa:

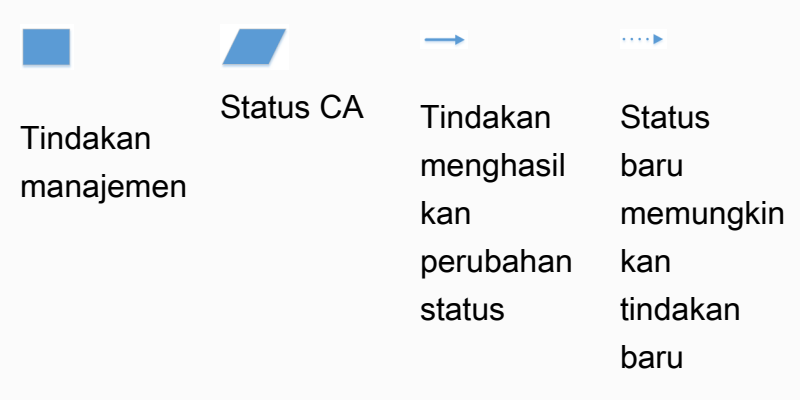
- Sertifikat CA tidak diperbarui secara otomatis. Untuk informasi tentang mengotomatisasi pembaruan melalui AWS Certificate Manager, lihat. Menetapkan izin perpanjangan sertifikat untuk ACM
- Jika Anda mencoba untuk menerbitkan sertifikat baru dengan CA yang kedaluwarsa, API `IssueCertificate` mengembalikan `InvalidStateException`. CA akar yang kedaluwarsa harus menandatangani sendiri sertifikat CA akar yang baru sebelum dapat menerbitkan sertifikat bawahan baru.
- The `ListCertificateAuthorities`dan `DescribeCertificateAuthority` APIs mengembalikan status `EXPIRED` jika sertifikat CA kedaluwarsa, terlepas dari apakah status CA disetel 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.

Hubungan antara 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.

Perbarui CA pribadi di AWS Private Certificate Authority

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. CAs

Perbarui CA (konsol)

Prosedur berikut menunjukkan cara memperbarui konfigurasi CA yang ada menggunakan. Konsol Manajemen AWS

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/rumah
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:

- Jenis CRL (lengkap atau dipartisi)
- 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/rumah.
2. Pada halaman Private Certificate Authority, 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 Rencanakan metode pencabutan AWS Private CA sertifikat Anda 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 CRLs. Gunakan salah satu pola kebijakan yang dijelaskan dalam Kebijakan akses untuk CRLs 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.

- Pilih Aktifkan partisi untuk mengaktifkan partisi. CRLs Jika Anda tidak mengaktifkan partisi, CA Anda tunduk pada jumlah maksimum sertifikat yang dicabut, yang ditunjukkan pada kuota.AWS Private Certificate Authority Untuk informasi selengkapnya tentang dipartisi CRLs, lihat tipe CRL.
- 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. Untuk menggunakan CRLs over IPv6, setel ini ke titik akhir S3 dualstack bucket Anda seperti yang dijelaskan dalam Using over. CRLs IPv6
- Tambahkan jalur Kustom untuk membuat alias DNS untuk jalur file di bucket Amazon S3 Anda.
- Ketik Validitas dalam beberapa hari CRL Anda akan tetap valid. Nilai default-nya adalah 7 hari. Untuk online CRLs, 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. Untuk menggunakan OCSP di atas IPv6, atur bidang ini ke titik akhir dualstack seperti yang dijelaskan dalam Menggunakan OCSP over. IPv6

   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. Kustomisasi URL OCSP untuk AWS Private CA

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

Nama catatan	Tipe	Kebijakan perutean	Pembeda	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 aturACTIVE. 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": "amzn-s3-demo-bucket"
   },
            "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": "amzn-s3-demo-bucket"
            },
            "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": "amzn-s3-demo-bucket"
   }
}
```

> **Note**
>
> Saat memperbarui CA pengesahan perangkat Matter untuk mengaktifkan CRLs, 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": "amzn-s3-demo-bucket"
      "CrlDistributionPointExtensionConfiguration":{
         "OmitExtension": true
      }
   }
}
```

3. Gunakan update-certificate-authorityperintah dan file konfigurasi pencabutan 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_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T13:46:50-08:00",
```

```
        "NotAfter": "2022-03-08T14:46:50-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "S3BucketName": "amzn-s3-demo-bucket",
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

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 update-certificate-authorityperintah dan file konfigurasi pencabutan 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.

Hapus CA pribadi Anda

Anda dapat menghapus CA pribadi dari Konsol Manajemen AWS 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 Kembalikan 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 AWS Private Certificate Authority.

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/rumah.
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 DescribeCerticateAuthorityatau ListCertificateAuthoritiesAPI.

Untuk menghapus CA privat (AWS CLI)

Gunakan delete-certificate-authorityperintah untuk menghapus CA pribadi.

```
$ 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
```

Kembalikan 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 Hapus CA pribadi 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 untuk AWS Private Certificate Authority.

Memulihkan CA privat (konsol)

Anda dapat menggunakan Konsol Manajemen AWS 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/rumah.
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.

Kembalikan CA (AWS CLI) pribadi

Gunakan restore-certificate-authorityperintah untuk memulihkan CA pribadi yang dihapus yang ada di `DELETED` negara bagian. 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 delete-certificate-authorityperintah untuk menghapus CA pribadi. Jika status CA pribadi adalah `DISABLED` ataupun`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 restore-certificate-authorityperintah untuk mengembalikan CA pribadi. 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 update-certificate-authorityperintah untuk mengubah status CA pribadi menjadi`ACTIVE`.

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

Gunakan sertifikat CA pribadi yang ditandatangani secara eksternal

Jika akar kepercayaan hierarki CA pribadi Anda harus berupa CA di luar AWS 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. Kendala 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 di AWS Private CA.

Prosedur berikut menjelaskan cara menyelesaikan tugas-tugas ini menggunakan salah satu Konsol Manajemen AWS atau AWS 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/rumah. 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 menyalin 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 layar Anda, gunakan `--output text` opsi untuk menghilangkan CR/LF karakter 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 dalam AWS Private CA.

```
$ aws acm-pca import-certificate-authority-certificate \
--certificate-authority-arn arn:aws:acm-pca:region:account:\
certificate-authority/12345678-1234-1234-1234-123456789012 \
--certificate file://C:\example_ca_cert.pem \
--certificate-chain file://C:\example_ca_cert_chain.pem
```

Menerbitkan dan mengelola sertifikat di AWS Private CA

Setelah Anda membuat dan mengaktifkan otoritas sertifikat pribadi (CA) dan mengkonfigurasi akses ke sana, Anda atau pengguna yang berwenang dapat menerbitkan dan mengelola sertifikat. Jika Anda belum menyiapkan kebijakan AWS Identity and Access Management (IAM) untuk CA, Anda dapat mempelajari lebih lanjut 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 Kontrol akses ke CA pribadi.

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 Gunakan templat AWS Private CA 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 menggunakan AWS Private CA, Anda dapat mengimpornya ke ACM dan menggunakannya dengan layanan yang didukung AWS .

> **Note**
>
> Sertifikat yang dibuat dengan prosedur di bawah ini, menggunakan issue-certificate perintah, atau dengan tindakan IssueCertificateAPI, tidak dapat langsung diekspor untuk digunakan di luar AWS. 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. CSRs 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 CA segera 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 APIPassthrough templat

Dalam contoh ini, sertifikat dikeluarkan yang berisi elemen nama subjek yang disesuaikan. Selain menyediakan CSR seperti yang ada di[Menerbitkan sertifikat standar (AWS CLI)](), Anda meneruskan dua argumen tambahan ke issue-certificate perintah: ARN APIPassthrough template, dan file konfigurasi JSON yang menentukan atribut kustom dan pengidentifikasi objek (). OIDs Anda tidak dapat menggunakan `StandardAttributes` dalam hubungannya dengan`CustomAttributes`. Namun, Anda dapat lulus standar OIDs sebagai bagian dari. `CustomAttributes` Nama subjek default OIDs 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.

Menerbitkan sertifikat dengan ekstensi khusus menggunakan APIPassthrough templat

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

file konfigurasi JSON yang menentukan ekstensi kustom, dan CSR seperti yang ditunjukkan di. [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
```

Ambil sertifikat pribadi

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 AWS CLI perintah get-certificate untuk mengambil sertifikat end-entity pribadi. 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 Gunakan laporan audit dengan CA pribadi 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 pribadi

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, lihatGunakan laporan audit dengan CA pribadi 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"
}
```

Ekspor sertifikat pribadi dan kunci rahasianya

AWS Private CA tidak 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 pribadi

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 CA menyediakan dua mekanisme yang dikelola sepenuhnya untuk mendukung pemeriksaan status pencabutan: Protokol Status Sertifikat Online (OCSP) dan daftar pencabutan sertifikat (). CRLs 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 CRLs 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 melalui AWS Private CA, lihat. Rencanakan metode pencabutan AWS Private CA sertifikat Anda

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

> **Note**
>
> Untuk penelepon lintas akun, pembagian dengan `AWSRAMRevokeCertificateCertificateAuthority` izin diperlukan. Izin pencabutan tidak termasuk dalam. `AWSRAMDefaultPermissionCertificateAuthority` 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 CRLs 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 CRLs, lihatSiapkan CRL untuk AWS Private CA.

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 Gunakan laporan audit dengan CA pribadi 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.

Gunakan templat AWS Private CA sertifikat

AWS Private CA menggunakan 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
- Bawahan CACertificate _ 0/V1 PathLen

Untuk informasi selengkapnya, lihat Kebijakan berbasis sumber daya.

Topik

- AWS Private CA varietas template
- AWS Private CA urutan template operasi
- AWS Private CA definisi template

AWS Private CA varietas template

AWS Private CA mendukung empat jenis template.

- Template dasar

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

- CSRPassthrough template

  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 AWS Private CA urutan template operasi.

- APIPassthrough template

  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 AWS Private CA urutan template 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 AWS Private CA urutan template operasi.

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

> **Note**
>
> Untuk informasi tentang template ARNs 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
OCSPSigningSertifikat/V1	`arn:aws:acm-pca:::template/OCSPSigningCertificate/V1`	Penandatanganan OCSP
CACertificateAkar/V1	`arn:aws:acm-pca:::template/RootCACertificate/V1`	CA
Bawahan CACertificate _ 0/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1`	CA
Bawahan CACertificate _ 1/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1/V1`	CA
Bawahan CACertificate _ 2/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2/V1`	CA

Nama Templat	ARN Templat	Jenis Sertifikat
Bawahan CACertificate _ 3/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3/V1`	CA

CSRPassthrough template

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_PathLen0_CSRPassthrough/V1	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1`	CA
BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1`	CA
BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1	`arn:aws:acm-pca:::template/BlankSubo`	CA

Nama Templat	ARN Templat	Jenis Sertifikat
	rdinateCACertificate_PathLen2_CSRPassthrough/V1	
BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1	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
OCSPSigningSertifikat/V1 CSRPassthrough	arn:aws:acm-pca:::template/OCSPSigningCertificate_CSRPassthrough/V1	Penandatanganan OCSP

Nama Templat	ARN Templat	Jenis Sertifikat
Bawahan CACertificate _ PathLen 0_/V1 CSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
Bawahan CACertificate _ PathLen 1_/V1 CSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
Bawahan CACertificate _ PathLen 2_/V1 CSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
Bawahan CACertificate _ PathLen 3_ /V1 CSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA

APIPassthrough template

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/BlankEndE ntityCertificate_C riticalBasicConstr aints_APIPassthrough/ V1	Entitas akhir
CodeSigningCertificate_ APIPassthrough /V1	arn:aws:acm-pca::: template/CodeSigni ngCertificate_APIP assthrough/V1	Penandatanganan kode
EndEntityCertificate_ APIPassth rough /V1	arn:aws:acm-pca::: template/EndEntity Certificate_APIPas sthrough/V1	Entitas akhir
EndEntityClientAuthCertificate_ APIPassthrough /V1	arn:aws:acm-pca::: template/EndEntity ClientAuthCertific ate_APIPassthrough/V1	Entitas akhir
EndEntityServerAuthCertificate_ APIPassthrough /V1	arn:aws:acm-pca::: template/EndEntity ServerAuthCertific ate_APIPassthrough/V1	Entitas akhir
OCSPSigningSertifikat/V1 APIPassthrough	arn:aws:acm-pca::: template/OCSPSigni ngCertificate_APIP assthrough/V1	Penandatanganan OCSP
Akar CACertificate _ APIPassth rough /V1	arn:aws:acm-pca::: template/RootCACer tificate_APIPassth rough/V1	CA

Nama Templat	ARN Templat	Jenis Sertifikat
BlankRootCACertificate_ APIPassthrough /V1	arn:aws:acm-pca::: template/BlankRoot CACertificate_APIP assthrough/V1	CA
BlankRootCACertificate_ PathLen 0_ /V1 APIPassthrough	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len0_APIPassthrough/V1	CA
BlankRootCACertificate_ PathLen 1_ APIPassthrough /V1	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len1_APIPassthrough/V1	CA
BlankRootCACertificate_ PathLen 2_ APIPassthrough /V1	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len2_APIPassthrough/V1	CA
BlankRootCACertificate_ PathLen 3_ APIPassthrough /V1	arn:aws:acm-pca::: template/BlankRoot CACertificate_Path Len3_APIPassthrough/V1	CA
Bawahan CACertificate _ PathLen 0_/V1 APIPassthrough	arn:aws:acm-pca::: template/Subordina teCACertificate_Pa thLen0_APIPassthrough/ V1	CA
BlankSubordinateCACertifica te_PathLen0_APIPassthrough/V1	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen0_APIPas sthrough/V1	CA

Nama Templat	ARN Templat	Jenis Sertifikat
Bawahan CACertificate _ PathLen 1_/V1 APIPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
Bawahan CACertificate _ PathLen 2_/V1 APIPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
Bawahan CACertificate _ PathLen 3_ /V1 APIPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APIPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1	CA

APICSRPassthrough template

Nama Templat	ARN Templat	Jenis Sertifikat
BlankEndEntityCertificate_ APICSRPassthrough /V1	arn:aws:acm-pca::: template/BlankEndE ntityCertificate_A PICSRPassthrough/V1	Entitas akhir
BlankEndEntityCertificate_ CriticalBasicConstraints _ APICSRPassthrough /V1	arn:aws:acm-pca::: template/BlankEndE ntityCertificate_C riticalBasicConstr aints_APICSRPassth rough/V1	Entitas akhir
CodeSigningCertificate_ APICSRPassthrough /V1	arn:aws:acm-pca::: template/CodeSigni ngCertificate_APIC SRPassthrough/V1	Penandatanganan kode
EndEntityCertificate_ APICSRPas sthrough /V1	arn:aws:acm-pca::: template/EndEntity Certificate_APICSR Passthrough/V1	Entitas akhir
EndEntityClientAuthCertificate_ APICSRPassthrough /V1	arn:aws:acm-pca::: template/EndEntity ClientAuthCertific ate_APICSRPassthrough/ V1	Entitas akhir
EndEntityServerAuthCertificate_ APICSRPassthrough /V1	arn:aws:acm-pca::: template/EndEntity ServerAuthCertific	Entitas akhir

Nama Templat	ARN Templat	Jenis Sertifikat
	ate_APICSRPassthrough/V1	
OCSPSigningSertifikat/V1 APICSRPassthrough	arn:aws:acm-pca:::template/OCSPSigningCertificate_APICSRPassthrough/V1	Penandatanganan OCSP
Bawahan CACertificate _ PathLen0_/V1 APICSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
Bawahan CACertificate _ PathLen1_/V1 APICSRPassthrough	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA
BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA

Nama Templat	ARN Templat	Jenis Sertifikat
Bawahan CACertificate _ PathLen 2_APICSRPassthrough/PathLen3_ V1 APIPassthrough	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APICSRPassthrough/V1`	CA
BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1`	CA
Bawahan CACertificate _ PathLen 3_ /V1 APICSRPassthrough	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA
BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA

AWS Private CA urutan template 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 ketika Anda menggunakan template 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 Nama Alternatif Subjek (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.

AWS Private CA definisi template

Bagian berikut memberikan rincian konfigurasi tentang template AWS Private CA sertifikat yang didukung.

BlankEndEntityCertificate_ APIPassthrough /V1 definisi

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,, c, encipherOnly, dan DecipherOnly) dan EKU dapat berupa salah satu nilai yang didukung (ServerAuth, ClientAuth keyCertSign, comendesainRLSign, EmailProtection, stempel waktu, dan) ditambah ekstensi khusus. OCSPSigning

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 /V1 definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

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 /V1 definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

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 /V1 definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

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 /V1 definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

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 /V1 definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

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.

BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1

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.

BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen0_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]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen0_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: 0`
Pengidentifikasi kunci otoritas	[SKI dari Sertifikat CA]
Pengidentifikasi kunci subjek	[Berasal dari CSR]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen1_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]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen1_CSRPassthrough/V1

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.

BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen1_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]

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.

BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen2_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]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinateCACertificate_PathLen2_CSRPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen2_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]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen2_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]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA atau CSR]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen3_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]
Titik distribusi CRL*	[Passthrough dari konfigurasi CA]

* Titik distribusi CRL disertakan dalam templat hanya jika CA dikonfigurasi dengan generasi CRL diaktifkan.

BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen3_CSRPassthrough/V1

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.

BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1definisi

Untuk informasi umum tentang templat kosong, lihat BlankEndEntityCertificate_ APIPassthrough /V1 definisi.

BlankSubordinateCACertificate_PathLen3_APICSRPassthrough

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 Kode AWS 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 /V1 definisi

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.

CodeSigningCertificate_ APIPassthrough /V1 definisi

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 /V1 definisi

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 /V1 definisi

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.

EndEntityCertificate_ APIPassthrough /V1 definisi

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 /V1 definisi

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 /V1 definisi

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.

EndEntityClientAuthCertificate_ APIPassthrough /V1 definisi

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 /V1 definisi

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 /V1 definisi

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.

EndEntityServerAuthCertificate_ APIPassthrough /V1 definisi

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 /V1 definisi

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.

OCSPSigningDefinisi sertifikat/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.

OCSPSigningSertifikat/V1

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.

OCSPSigningDefinisi sertifikat_/V1 APICSRPassthrough

Template ini memperluas OCSPSigning Certificate/V1 untuk mendukung nilai passthrough API dan CSR.

OCSPSigningAPICSRPassthroughSertifikat_/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.

OCSPSigningDefinisi sertifikat_/V1 APIPassthrough

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.

OCSPSigningAPIPassthroughSertifikat_/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.

OCSPSigningDefinisi sertifikat_/V1 CSRPassthrough

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.

OCSPSigningCSRPassthroughSertifikat_/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 CACertificate Root/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 ekspansi 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.

CACertificateAkar/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 Root CACertificate _ APIPassthrough /V1

Template ini memperluas CACertificate Root/V1 untuk mendukung nilai passthrough API.

Akar CACertificate _ 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

BlankRootCACertificate_ APIPassthrough /V1 definisi

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]

BlankRootCACertificatePathLen_ APIPassthrough 0_/V1 definisi

Untuk informasi umum tentang templat CA root kosong, lihat???.

BlankRootCACertificate_ PathLen 0_ /V1 APIPassthrough

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]

BlankRootCACertificatePathLen_ APIPassthrough 1_/V1 definisi

Untuk informasi umum tentang templat CA root kosong, lihat???.

BlankRootCACertificate_ 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 subjek	[Berasal dari CSR]

BlankRootCACertificatePathLen_ APIPassthrough 2_/V1 definisi

Untuk informasi umum tentang templat CA root kosong, lihat???.

BlankRootCACertificate_ 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 subjek	[Berasal dari CSR]

BlankRootCACertificatePathLen_ APIPassthrough 3_/V1 definisi

Untuk informasi umum tentang templat CA root kosong, lihat???.

BlankRootCACertificate_ PathLen 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 subjek	[Berasal dari CSR]

Definisi bawahan CACertificate _ PathLen 0/V1

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.

Bawahan CACertificate _ 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.

Definisi bawahan CACertificate PathLen _ 0_/V1 APICSRPassthrough

Template ini memperluas Subordinate CACertificate _ PathLen 0/V1 untuk mendukung nilai passthrough API dan CSR.

Bawahan CACertificate _ PathLen 0_/V1 APICSRPassthrough

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.

Definisi bawahan CACertificate PathLen _ 0_/V1 APIPassthrough

Template ini memperluas Bawahan CACertificate _ PathLen 0/V1 untuk mendukung nilai passthrough API.

Bawahan CACertificate _ PathLen 0_/V1 APIPassthrough

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.

Definisi bawahan CACertificate PathLen _ 0_/V1 CSRPassthrough

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

Bawahan CACertificate _ PathLen 0_/V1 CSRPassthrough

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.

Definisi bawahan CACertificate _ PathLen 1/V1

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.

Bawahan CACertificate _ 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.

Definisi bawahan CACertificate PathLen _ 1_/V1 APICSRPassthrough

Template ini memperluas Subordinate CACertificate _ PathLen 1/V1 untuk mendukung nilai passthrough API dan CSR.

Bawahan CACertificate _ PathLen 1_/V1 APICSRPassthrough

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.

Definisi bawahan CACertificate PathLen _ 1_/V1 APIPassthrough

Template ini memperluas Bawahan CACertificate _ PathLen 0/V1 untuk mendukung nilai passthrough API.

Bawahan CACertificate _ PathLen 1_/V1 APIPassthrough

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.

Definisi bawahan CACertificate PathLen _ 1_/V1 CSRPassthrough

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

Bawahan CACertificate _ PathLen 1_/V1 CSRPassthrough

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.

Definisi bawahan CACertificate _ PathLen 2/V1

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.

Bawahan CACertificate _ 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.

Definisi bawahan CACertificate PathLen _ 2_/V1 APICSRPassthrough

Template ini memperluas Subordinate CACertificate _ PathLen 2/V1 untuk mendukung nilai passthrough API dan CSR.

Bawahan CACertificate _ PathLen 2_/V1 APICSRPassthrough

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.

Definisi bawahan CACertificate PathLen _ 2_/V1 APIPassthrough

Template ini memperluas Bawahan CACertificate _ PathLen 2/V1 untuk mendukung nilai passthrough API.

Bawahan CACertificate _ PathLen 2_/V1 APIPassthrough

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.

Definisi bawahan CACertificate PathLen _ 2_/V1 CSRPassthrough

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

Bawahan CACertificate _ PathLen 2_/V1 CSRPassthrough

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.

Definisi bawahan CACertificate _ PathLen 3/V1

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.

Bawahan CACertificate _ 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.

Definisi bawahan CACertificate PathLen _ 3_/V1 APICSRPassthrough

Template ini memperluas Subordinate CACertificate _ PathLen 3/V1 untuk mendukung nilai passthrough API dan CSR.

Bawahan CACertificate _ PathLen 3_ /V1 APICSRPassthrough

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.

Definisi bawahan CACertificate PathLen _ 3_/V1 APIPassthrough

Template ini memperluas Bawahan CACertificate _ PathLen 3/V1 untuk mendukung nilai passthrough API.

Bawahan CACertificate _ PathLen 3_ /V1 APIPassthrough

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.

Definisi bawahan CACertificate PathLen _ 3_/V1 CSRPassthrough

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

Bawahan CACertificate _ PathLen 3_ /V1 CSRPassthrough

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.

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) AWS Private Certificate Authority

Akses ke AWS Private CA memerlukan kredensil yang AWS dapat digunakan untuk mengautentikasi permintaan Anda. Topik berikut memberikan rincian tentang bagaimana Anda dapat menggunakan AWS Identity and Access Management (IAM) untuk membantu mengamankan otoritas sertifikat pribadi Anda (CAs) 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 kredensil 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 CAs.
  > - 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 dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

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

- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Buat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
  - (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam 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.

AWSPrivateCAFullAkses

Memberikan kontrol administratif yang tidak terbatas.

Untuk daftar JSON tentang detail kebijakan, lihat AWSPrivateCAFullAkses.

AWSPrivateCAReadHanya

Memberikan akses terbatas pada operasi API hanya-baca.

Untuk daftar JSON tentang detail kebijakan, lihat AWSPrivateCAReadHanya.

AWSPrivateCAPrivilegedPengguna

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.

Untuk daftar JSON tentang detail kebijakan, lihat AWSPrivateCAPrivilegedPengguna.

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

Untuk daftar JSON tentang detail kebijakan, lihat AWSPrivateCAUser.

AWSPrivateCAAuditor

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

Untuk daftar JSON tentang detail kebijakan, lihat AWSPrivateCAAuditor.

AWSPrivateCAConnectorForKubernetesPolicy

Memberikan izin penting untuk AWS Private CA Konektor untuk Kubernetes.

Untuk daftar JSON tentang detail kebijakan, lihat AWSPrivateCAConnectorForKubernetesPolicy.

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

Ubah	Deskripsi	Date
Kebijakan Baru: AWS Pribadi CAConnector ForKubern etesPolicy	Kebijakan terkelola baru diperkenalkan untuk digunakan dengan AWS Private CA Connector untuk Kubernetes.	19 Mei 2025
AWS CAPrivilegedPenggu na Pribadi dan AWS Pribadi CAUser - Kebijakan yang diperbarui	Diganti `StringLik e` dengan`ArnLike`, dan `StringNotLike` dengan`ArnNotLike` . Template arn yang diperbaru i untuk memasukkan kartu	Januari 22, 2025

Ubah	Deskripsi	Date
	liar `arn:aws:acm-pca:::template` `kearn:aws:acm-pca:*:*:template` .	
Nama kebijakan baru: • `AWS PrivateCA FullAccess` • `AWS PrivateCA ReadOnly` • `AWS PrivateCA PrivilegedUser` • `AWS PrivateCAAuditor` • `AWS PrivateCAUser`	Awalan nama kebijakan diubah dari `AWS CertificateManagerPrivateCA` menjadi. `AWS PrivateCA` 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.

   JSON

```
{
   "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 Konsol Manajemen AWS, 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

Daftar pribadi CAs

Kebijakan berikut memungkinkan pengguna untuk mencantumkan semua pribadi CAs di akun.

JSON

```
{
   "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.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:GetCertificateAuthorityCertificate",
      "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID/certificate/certificate_ID"
   }
}
```

Mengimpor sertifikat CA privat

Kebijakan berikut memungkinkan pengguna untuk mengimpor sertifikat CA privat.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:ImportCertificateAuthorityCertificate",
      "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID/certificate/certificate_ID"
   }
}
```

Menghapus CA privat

Kebijakan berikut memungkinkan pengguna untuk menghapus CA privat tertentu.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:DeleteCertificateAuthority",
      "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID/certificate/certificate_ID"   }
}
```

Tag-on-create: Melampirkan tag ke CA pada saat pembuatan

Kebijakan berikut memungkinkan pengguna untuk menerapkan tag selama pembuatan CA.

JSON

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

Tag-on-create: Penandaan terbatas

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

JSON

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

Mengontrol akses ke CA Pribadi menggunakan tag

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

JSON

```
{
   "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.

JSON

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

Akses penuh ke AWS Private CA

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

JSON

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

Praktik terbaik keamanan untuk akses lintas akun ke pribadi CAs

AWS Private CA Administrator dapat berbagi CA dengan prinsipal (pengguna, peran, dll.) di tempat lain. Akun 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 tentang AWS Organizations, lihat AWS Organizations Panduan 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.

Kontrol akses ke CA pribadiBagian panduan ini menyediakan alur kerja untuk memberikan akses ke CAs 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 PCA APIs PutPolicy, dan GetPolicy, 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

- Bawahan CACertificate _ 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 yang ARNs dapat dikonfigurasi oleh pelanggan.

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

   Ganti *555555555555* 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.

   JSON

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

```
    "Statement": [{
            "Sid": "ExampleStatementID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "555555555555"
            },
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListPermissions",
                "acm-pca:ListTags"
            ],
            "Resource": "arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID"
        },
        {
            "Sid": "ExampleStatementID2",
            "Effect": "Allow",
            "Principal": {
                "AWS": "555555555555"
            },
            "Action": [
                "acm-pca:IssueCertificate"
            ],
            "Resource": "arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID",
            "Condition": {
                "StringEquals": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
EndEntityCertificate/V1"
                }
            }
        }
    ]
}
```

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

   Ganti *o-a1b2c3d4z5* dengan AWS Organizations ID.

   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.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleStatementID3",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "acm-pca:IssueCertificate",
            "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID",
            "Condition": {
                "StringEquals": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
EndEntityCertificate/V1",
                    "aws:PrincipalOrgID": "o-a1b2c3d4z5"
                },
                "StringNotEquals": {
                    "aws:PrincipalAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "ExampleStatementID4",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListPermissions",
                "acm-pca:ListTags"
            ],
            "Resource":"arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/CA_ID",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "o-a1b2c3d4z5"
```

```
                },
                "StringNotEquals": {
                    "aws:PrincipalAccount": "111122223333"
                }
            }
        }
    ]
}
```

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 kredensil 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 AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.

- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-3.

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 SDKs. 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 pribadi CAs disimpan dalam modul keamanan perangkat keras AWS terkelola (HSMs). HSMs 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 berdasarkan 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 Anda CRLs
- Panduan Quick Start Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan menyediakan langkah untuk deployment lingkungan dasar yang berfokus pada keamanan dan kepatuhan di 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 dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub CSPM— 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.

Gunakan laporan audit dengan CA pribadi 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 `amzn-s3-demo-bucket`. `CA_ID` adalah pengenal unik penerbitan CA. `UUID` adalah pengenal unik laporan audit.

```
amzn-s3-demo-bucket/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.

Siapkan bucket Amazon S3 untuk laporan audit

Important

AWS Private CA tidak mendukung penggunaan Amazon S3 Object Lock. Jika Anda mengaktifkan Object Lock di bucket, Anda AWS Private CA tidak dapat menulis laporan audit ke bucket.

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 memungkinkan AWS Private CA untuk mengakses dan menulis ke bucket S3 yang Anda tentukan. 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. Atau, Anda dapat menggunakan kunci kondisi aws: SourceOrg ID untuk membatasi akses ke organisasi tertentu di

AWS Organizations. Untuk informasi selengkapnya tentang kebijakan bucket, lihat Kebijakan Bucket untuk Amazon Simple Storage Service.

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/rumah.
2. Pada halaman Private Certificate Authories, 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. Kebijakan default menyertakan kunci `aws:SourceAccount` kondisi, yang membatasi akses ke AWS akun tertentu. Jika ingin membatasi akses lebih lanjut, Anda dapat menambahkan kunci kondisi lain ke kebijakan seperti pada contoh sebelumnya.

   Jika Anda memilih Tidak, Anda harus melampirkan kebijakan ke bucket agar dapat membuat laporan audit. Gunakan pola kebijakan yang dijelaskan dalamSiapkan 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 dalamSiapkan 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 amzn-s3-demo-bucket \
--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 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
```

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.

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. 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 (Site-to-Site VPN) atau. Direct Connect Untuk selengkapnya Site-to-Site VPN, lihat Koneksi VPN di Panduan Pengguna Amazon VPC. Untuk informasi tentang Direct Connect, lihat Membuat hubungan di Panduan Pengguna Direct Connect .

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 VPC endpoint AWS Private CA

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 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 (Hyderabad)
- Asia Pasifik (Jakarta)
- Asia Pasifik (Melbourne)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- (Canada (Central)
- Kanada Barat (Calgary)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Milan)
- Eropa (Paris)
- Eropa (Spanyol)
- Eropa (Stockholm)
- Eropa (Zürich)
- Israel (Tel Aviv)
- Timur Tengah (Bahrain)
- Timur Tengah (UEA)

- 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 atau. https://console.aws.amazon.com/vpc/ 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-nya 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**
>
> Nilai tersebut *region* 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 padanya tindakan dapat dilakukan

Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan Titik Akhir VPC di Panduan VPC Amazon.

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"
            ]
         }
      ]
   }
```

Dukungan titik akhir tumpukan ganda

AWS Private Certificate Authority menyediakan titik akhir publik dual-stack yang mendukung keduanya IPv4 dan klien. IPv6 Titik akhir dual-stack memungkinkan klien untuk berkomunikasi dengan AWS Private CA menggunakan salah satu atau IPv4 alamat. IPv6 AWS Private CA untuk Active Directory dan AWS Private CA Connector untuk SCEP juga mendukung titik akhir dual-stack.

Titik akhir publik AWS Private CA dual-stack di `https://acm-pca.your-region.api.aws` mendukung keduanya IPv4 dan klien. IPv6 AWS Private CA juga dapat diakses secara pribadi melalui IPv4 dan IPv6 dari virtual private cloud (VPC) Anda menggunakan. AWS PrivateLink Untuk informasi selengkapnya tentang membuat titik akhir VPC antarmuka pribadi, lihat. AWS Private CAAWS Private CA Titik akhir VPC ()AWS PrivateLink

Untuk informasi selengkapnya, lihat sumber daya berikut:

- Pengalamatan IP untuk Anda VPCs dan subnet
- IPv6 dukungan untuk VPC Anda

Menggunakan IPv6 alamat di IAM dan AWS Private CA

Sebelum mencoba mengakses AWS Private Certificate Authority lebih IPv6, pastikan setiap kebijakan IAM yang berisi pembatasan alamat IP diperbarui untuk menyertakan rentang IPv6 alamat. Kebijakan berbasis IP yang tidak diperbarui untuk menangani IPv6 alamat dapat mengakibatkan klien salah kehilangan atau mendapatkan akses ketika mereka mulai menggunakan IPv6. Untuk mempelajari lebih lanjut tentang AWS Private CA dan dukungan dual-stack, lihat. Dukungan titik akhir tumpukan ganda

> **Important**
>
> Pernyataan ini tidak mengizinkan tindakan apa pun. Gunakan pernyataan ini dalam kombinasi dengan pernyataan lain yang memungkinkan tindakan tertentu.

Pernyataan berikut secara eksplisit menolak akses ke semua AWS Private CA izin untuk permintaan yang berasal dari rentang alamat. `192.0.2.*` IPv4 Alamat IP apa pun di luar rentang ini tidak secara eksplisit ditolak AWS Private CA izin. Karena semua IPv6 alamat berada di luar rentang yang ditolak, pernyataan ini tidak secara eksplisit menolak AWS Private CA izin untuk alamat apa pun. IPv6

```
{
    "Sid": "DenyPrivateCAPermissions",
    "Effect": "Deny",
    "Action": [
        "acm-pca:*"
    ],
    "Resource": "*",
    "Condition": {
        "NotIpAddress": {
            "aws:SourceIp": [
                "192.0.2.0/24"
            ]
        }
    }
}
```

Anda dapat memodifikasi `Condition` elemen untuk menolak rentang alamat IPv4 IPv6 (`192.0.2.0/24`2001:db8::/32) dan () seperti yang ditunjukkan pada contoh berikut:

```
{
    "Sid": "DenyPrivateCAPermissions",
    "Effect": "Deny",
    "Action": [
        "acm-pca:*"
    ],
    "Resource": "*",
    "Condition": {
        "NotIpAddress": {
```

```
            "aws:SourceIp": [
                "192.0.2.0/24",
                "2001:db8::/32"
            ]
        }
    }
}
```

AWS Private Certificate Authority CP/CPS Kerangka Pelanggan

AWS Private Certificate Authority menyediakan layanan infrastruktur yang memungkinkan Anda membuat hierarki otoritas sertifikat (CA), termasuk root dan bawahan CAs, tanpa biaya investasi dan pemeliharaan pengoperasian CA on-premise. Ketika Anda menggunakan AWS Private CA untuk membuat hierarki CA Anda, ada tanggung jawab bersama antara Anda dan AWS Private CA. Model tanggung jawab bersama dapat membantu meringankan beban operasional Anda saat AWS mengoperasikan, mengelola, dan mengendalikan keamanan fisik fasilitas tempat layanan beroperasi. Anda memikul tanggung jawab dan pengelolaan otoritas sertifikat (termasuk pembuatan dan penghapusan sumber daya CA; mendistribusikan jangkar kepercayaan; pembuatan hierarki PKI; kebijakan dan praktik sertifikasi; konfigurasi untuk mengizinkan atau menolak berbagi CA di seluruh Akun AWS; kebijakan untuk penggunaan templat; audit; kontrol akses, termasuk pemisahan tugas; dan konfigurasi dan kebijakan CA lainnya). Anda harus mempertimbangkan dengan cermat layanan yang Anda pilih karena tanggung jawab Anda bervariasi tergantung pada layanan yang digunakan, integrasi layanan tersebut ke dalam lingkungan TI Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya, lihat Model Tanggung Jawab Bersama AWS Cloud Keamanan.

Membuat kebijakan sertifikat (CP) atau pernyataan praktik sertifikasi (CPS) untuk otoritas sertifikat pribadi Anda adalah bagian penting dalam mengelola infrastruktur kunci publik (PKI) Anda. CP mendefinisikan semua requirements/rules untuk PKI Anda dan CPS menjelaskan bagaimana Anda memenuhi persyaratan CP. Anda bertanggung jawab untuk membuat CP dan CPS sebagai otoritas sertifikat PKI Anda. AWS Private CA memberi Anda dokumentasi AWS kontrol dan kepatuhan, seperti Laporan Kontrol AWS Sistem dan Organisasi (SOC) 2, yang dapat Anda gunakan untuk membantu membuat CP dan CPS Anda dan untuk melakukan evaluasi kontrol dan prosedur verifikasi sesuai kebutuhan. AWS Laporan SOC adalah laporan pemeriksaan pihak ketiga independen yang menunjukkan bagaimana AWS mencapai kontrol dan tujuan kepatuhan utama. Tujuan dari laporan ini adalah untuk membantu Anda dan auditor Anda memahami AWS kontrol yang ditetapkan untuk mendukung operasi dan kepatuhan.

Dokumen ini menyajikan kerangka kerja yang selaras dengan RFC 3647 untuk membantu Anda menulis CP dan CPS Anda dan mengidentifikasi tanggung jawab bersama antara Anda dan. AWS Private CA Bagian dari CP/CPS persyaratan yang AWS Private CA memiliki tanggung jawab kepatuhan diidentifikasi dengan “Dibagikan” atau “AWS Private Certificate Authority” dan “Informasi Tambahan” yang sesuai disediakan untuk membantu Anda memahami bagaimana AWS Private CA memenuhi persyaratan terkait CP/CPS . Misalnya, Persyaratan 5 (4.5.1) adalah AWS Private CA tanggung jawab dan Anda dapat menemukan bahasa kontrol yang sesuai di Bagian D.6 Laporan AWS SOC 2 untuk membantu menyelesaikan CP/CPS Anda. Untuk informasi lebih lanjut tentang Laporan AWS SOC dan bagaimana Anda dapat meminta akses ke Laporan SOC, silakan kunjungi halaman FAQsSOC kami.

Persyaratan dan Tanggung Jawab CP/CPS

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
1. Pendahuluan (Semua)	Anda	Anda bertanggung jawab untuk mendokumentasikan ikhtisar, nama dan identifik asi dokumen, peserta PKI, penggunaan sertifikat, administrasi kebijakan, dan definisi dan akronim yang terkait dengan PKI Anda.
2. Tanggung Jawab Publikasi dan Repositori (Semua)	Anda	Anda bertanggung jawab untuk mendokumentasikan definisi yang terkait dengan PKI Anda.
3. Identifikasi dan Otentikasi (Semua)	Anda	Anda bertanggung jawab untuk mendokumentasikan prosedur yang digunakan untuk mengotentikasi identitas atribut and/or lain dari pemohon sertifikat pengguna akhir ke CA atau Registrat

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
		ion Authority (RA) sebelum penerbitan sertifikat.
4. Persyaratan Operasional Siklus Hidup Sertifikat (4.4.1 — 4.4.6, 4.4.9 — 4.4.11)	Bersama	Anda bertanggung jawab untuk menentukan persyaratan yang dikenakan pada penerbitan CA, subjek CAs, RAs, pelanggan, atau peserta lain sehubungan dengan siklus hidup sertifikat. AWS Private CA memberi Anda dua mekanisme yang dikelola sepenuhnya untuk membantu mendukung pemeriksaan status pencabutan: Protokol Status Sertifikat Online (OCSP) dan daftar pencabutan sertifikat (CRLs) untuk membantu Anda memenuhi 4.4.9 dan 4.4.10.
4. Persyaratan Operasional Siklus Hidup Sertifikat (4.4.7, 4.4.8, 4.4.12)	N/A	AWS Private CA tidak mendukung Kunci Ulang Sertifikat, Modifikasi Sertifikat, atau Escrow Kunci dan Pemulihan.

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
5. Fasilitas, Manajemen, dan Kontrol Operasional (4.5.1)	AWS Private CA	Anda mewarisi kontrol akses yang membantu Anda memenuhi persyaratan di bagian ini yang berada dalam lingkup Laporan AWS Private CA SOC 2 Tipe 2 (lihat Bagian D.6 Keamanan Fisik dan Perlindungan Lingkungan). **Note** Anda bertanggung jawab atas keamanan fisik dan klasifikasi data CA data yang diekspor atau ditransfer ke luar AWS lingkungan tetapi tidak untuk keamanan fisik data CA yang disimpan di AWS.
5. Fasilitas, Manajemen, dan Pengendalian Operasional (4.5.2)	Bersama	Anda bertanggung jawab untuk memenuhi persyaratan di bagian ini khusus untuk menentukan peran tepercaya untuk operasi lingkungan PKI Anda. AWS Private CA mempertahankan peran tepercaya khusus untuk akses fisik modul kriptografi.

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
5. Fasilitas, Manajemen, dan Pengendalian Operasional (4.5.3)	Bersama	Anda bertanggung jawab untuk memenuhi persyaratan dalam bagian ini khusus untuk pemeriksaan latar belakang, pelatihan, dan prosedur tindakan disipliner untuk orang tepercaya Anda. Anda mewarisi kontrol yang terkait dengan pemeriksaan latar belakang, pelatihan, dan prosedur tindakan disipliner untuk AWS karyawan yang berada dalam lingkup Laporan AWS Private CA SOC 2 Tipe 2 (lihat Bagian A. Kebijakan , Lingkungan Kontrol A.1, B. Komunikasi, dan Organisas i Keamanan D.1 dan Akses Pengguna Karyawan D.2).

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
5. Fasilitas, Manajemen, dan Pengendalian Operasional (4.5.4)	Bersama	Anda bertanggung jawab untuk mengaktifkan, mengonfigurasi retensi, dan melindungi serta mengaudit log CloudTrail dan CloudWatch peringatan pelaporan. Selain itu, Anda bertanggung jawab untuk membuat prosedur pemrosesan log dan melakukan penilaian kerentanan atas penggunaan Anda atas AWS Private CA layanan yang memenuhi persyaratan di bagian ini. Anda mewarisi kontrol yang terkait dengan ketersediaan log, access/site keamanan fisik, manajemen CA/RA konfigurasi, keamanan log infrastruktur, dan penilaian kerentanan AWS AWS infrastruktur yang berada dalam lingkup Laporan AWS Private CA SOC 2 Tipe 2 (lihat Bagian A.1 Lingkungan Kontrol, Bagian C.1 Komitmen Layanan, Akses Pengguna Karyawan D.2, Keamanan Logis D.3, Keamanan Fisik dan Perlindungan Lingkungan D.6, Manajemen Perubahan D.7, Integritas Data D.8, Ketersediaan, dan

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
		Redundansi, dan E.1 Kegiatan Pemantauan).
5. Fasilitas, Manajemen, dan Kontrol Operasional (4.5.5)	Bersama	Anda bertanggung jawab untuk mengonfigurasi periode pencadangan dan retensi yang memenuhi persyaratan di bagian ini. Anda mewarisi kontrol yang terkait dengan ketersedi aan log Anda (saat Anda mengonfigurasi) yang berada dalam cakupan Laporan AWS Private CA SOC 2 Tipe 2 (lihat D.8 Integritas Data, Ketersedi aan, dan Redundansi).
5. Fasilitas, Manajemen, dan Kontrol Operasional (4.5.6)	N/A	AWS Private CA tidak mendukung Pergantian Kunci.

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
5. Fasilitas, Manajemen, dan Pengendalian Operasional (4.5.7)	Bersama	Anda bertanggung jawab untuk menerapkan prosedur penanganan insiden dan kompromi khusus untuk penggunaan Anda AWS Private CA yang memenuhi persyaratan dalam bagian ini. Anda mewarisi insiden, prosedur penanganan kompromi, kelangsun gan bisnis, dan prosedur pemulihan bencana khusus untuk perumahan lokasi fisik dan operasi infrastruktur yang membantu Anda memenuhi persyaratan di bagian ini yang berada dalam lingkup Laporan Privasi AWS Private CA SOC 2 Tipe 2 (lihat D.8 Integritas Data, Ketersediaan, dan Redundansi dan Privasi Bagian D.10).
5. Fasilitas, Manajemen, dan Pengendalian Operasional (4.5.8)	Anda	Anda diharuskan untuk mendokumentasikan persyarat an yang berkaitan dengan prosedur penghentian dan penghentian CA atau RA, termasuk identitas penjaga catatan arsip CA dan RA.

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
6. Kontrol Teknis (4.6.1)	Bersama	Anda bertanggung jawab untuk mendokumentasikan generasi kunci dan kebutuhan instalasi untuk PKI Anda. AWS Private CA memberi Anda modul kriptografi yang bersertifikat FIPS 140-3 level 3 untuk pembuatan kunci CA.
6. Kontrol Teknis (4.6.2)	Bersama	Anda bertanggung jawab untuk mendokumentasikan perlindungan kunci pribadi dan kontrol rekayasa modul kriptografi seperti persyaratan standar kriptografi dan kontrol multi-orang. AWS Private CA memberi Anda modul kriptografi yang bersertifikat FIPS 140-3 level 3 untuk pembuatan kunci CA dan kontrol akses fisik dua pihak ke. HSMs
6. Kontrol Teknis (4.6.3)	Anda	Anda bertanggung jawab untuk mendokumentasikan aspek-aspek lain dari manajemen key pair seperti pengarsipan kunci publik dan periode operasional sertifikat Anda.

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
6. Kontrol Teknis (4.6.4)	N/A	AWS AWS Private CA HSMs selalu online dan tidak memiliki gagasan tentang “data aktivasi”.

> **Note**
>
> Anda bertanggung jawab untuk menerapkan kontrol akses pengguna ke CA Pribadi Anda untuk membatasi kemampuan membuat CA dan menerbitkan sertifikat dengan tepat.

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
6. Kontrol Teknis (4.6.5)	Bersama	Anda bertanggung jawab untuk mendokumentasikan kontrol keamanan komputer untuk penggunaan Private CA Anda. Anda mewarisi kontrol yang terkait dengan akses logis AWS karyawan, kontrol keamanan jaringan dan komputer dari AWS infrastru ktur, dan kontrol parameter kata sandi akun AWS karyawan yang berada dalam lingkup Laporan AWS Private CA SOC 2 Tipe 2 (lihat Bagian D.2 Akses Pengguna Karyawan, Keamanan Logis D.3, dan Keamanan Fisik D.6 dan Perlindungan Lingkunga n).

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
6. Kontrol Teknis (4.6.6)	Bersama	Anda bertanggung jawab untuk mendokumentasikan kontrol manajemen keamanan yang terkait dengan penggunaan Private CA Anda. Anda mewarisi kontrol yang terkait dengan kontrol pengembangan sistem AWS Private CA layanan yang berada dalam lingkup Laporan AWS Private CA SOC 2 Tipe 2 (lihat Bagian D.7 Manajemen Perubahan).
6. Kontrol Teknis (4.6.7)	Bersama	Anda bertanggung jawab untuk mendokumentasikan kontrol keamanan jaringan untuk penggunaan Private CA jika berlaku untuk lingkungan PKI Anda. Anda mewarisi kontrol yang terkait dengan kontrol keamanan jaringan AWS infrastruktur yang berada dalam lingkup Laporan AWS Private CA SOC 2 Tipe 2 (lihat Bagian C.1 Komitmen Layanan, Keamanan Logis D.3, dan Aktivitas Pemantaua n E.1).

Persyaratan CP/CPS	Tanggung jawab	Informasi Tambahan
6. Kontrol Teknis (4.6.8)	AWS Private CA	AWS Private CA menggunakan sumber waktu tepercaya untuk stempel waktu data CA.
7. Profil Sertifikat, CRL, dan OCSP (Semua)	Bersama	Anda bertanggung jawab untuk mendokumentasikan persyaratan profil dan masukan sertifikat yang memenuhi kebutuhan lingkungan PKI Anda. AWS Private CA menyediakan Anda dengan template profil untuk membantu memenuhi persyaratan profil Anda.
8. Audit Kepatuhan dan Penilaian Lainnya (Semua)	Bersama	Anda bertanggung jawab untuk mendokumentasikan audit kepatuhan dan penilaian lainnya. AWS Private CA memberi Anda Laporan SOC 2 untuk membantu Anda dan auditor Anda memahami AWS kontrol yang ditetapkan untuk mendukung operasi dan kepatuhan.
9. Masalah Bisnis dan Hukum Lainnya	Anda	Anda bertanggung jawab untuk mendokumentasikan bisnis umum dan masalah hukum yang mencakup CA Pribadi Anda.

Pantau AWS Private CA sumber daya

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja AWS Private CA dan AWS solusi Anda yang lain. AWS menyediakan alat pemantauan berikut untuk menonton AWS Private CA, melaporkan ketika ada sesuatu yang salah, dan mengambil tindakan otomatis bila perlu:

- Amazon CloudWatch memantau AWS sumber daya Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat CloudWatch melacak penggunaan CPU atau metrik lain dari EC2 instans Amazon Anda dan secara otomatis meluncurkan instans baru bila diperlukan. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.
- Amazon CloudWatch Logs memungkinkan Anda memantau, menyimpan, dan mengakses file log Anda dari EC2 instans Amazon CloudTrail, dan sumber lainnya. CloudWatch Log dapat memantau informasi dalam file log dan memberi tahu Anda ketika ambang batas tertentu terpenuhi. Anda juga dapat mengarsipkan data log dalam penyimpanan yang sangat durabel. Untuk informasi selengkapnya, lihat Panduan Pengguna Amazon CloudWatch Logs.
- AWS CloudTrail merekam panggilan API dan kejadian terkait yang dilakukan oleh atau atas Akun AWS Anda dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun yang memanggil AWS, alamat IP asal panggilan dilakukan, dan waktu panggilan terjadi. Untuk informasi selengkapnya, lihat Panduan Pengguna AWS CloudTrail.
- Amazon EventBridge adalah layanan bus acara tanpa server yang memudahkan untuk menghubungkan aplikasi Anda dengan data dari berbagai sumber. EventBridge mengirimkan aliran data real-time dari aplikasi Anda sendiri, aplikasi Software-as-a-Service (SaaS), AWS dan layanan dan rute data tersebut ke target seperti Lambda. Hal ini memungkinkan Anda memantau kejadian yang terjadi dalam layanan, dan membangun arsitektur yang didorong kejadian. Untuk informasi selengkapnya, lihat Panduan EventBridge Pengguna Amazon.

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

AWS Private CA CloudWatch metrik

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 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

Monitor AWS Private CA dengan 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

Setelah berhasil, operasi mengembalikan CA dan sertifikat baru. ARNs

```
{
   "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

Setelah berhasil, operasi mengembalikan ARNs CA dan sertifikat yang 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"
   }
}
```

Pencatatan panggilan AWS Private Certificate Authority API menggunakan AWS CloudTrail

AWS Private Certificate Authority terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di AWS Private CA. CloudTrail menangkap panggilan API dan operasi penandatanganan untuk AWS Private CA as event. Panggilan yang diambil termasuk panggilan dari AWS Private CA konsol dan panggilan kode ke operasi AWS Private CA API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk. AWS Private CA Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat AWS Private CA, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

AWS Private CA informasi di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas terjadi di AWS Private CA, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di situs Anda Akun AWS. Untuk informasi selengkapnya, lihat Melihat peristiwa dengan Riwayat CloudTrail acara.

Untuk catatan acara yang sedang berlangsung di Anda Akun AWS, termasuk acara untuk AWS Private CA, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran umum untuk membuat jejak
- CloudTrail layanan dan integrasi yang didukung
- Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail
- Menerima file CloudTrail log dari beberapa wilayah dan Menerima file CloudTrail log dari beberapa akun

Semua AWS Private CA tindakan dicatat oleh CloudTrail dan didokumentasikan dalam referensi AWS Private CA API. Misalnya, panggilan ke`ImportCACertificate`, `IssueCertificate` dan `CreateAuditReport` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan itu dibuat dengan kredenal pengguna root atau AWS Identity and Access Management (IAM).
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi selengkapnya, lihat Elemen userIdentity CloudTrail .

AWS Private CA acara manajemen

AWS Private CA terintegrasi dengan CloudTrail untuk merekam tindakan API yang dibuat oleh pengguna, peran, atau AWS layanan di AWS Private CA. Anda dapat menggunakan CloudTrail untuk memantau permintaan AWS Private CA API secara real time dan menyimpan log di Amazon Simple Storage Service, Amazon CloudWatch Logs, dan Amazon CloudWatch Events. AWS Private CA mendukung pencatatan tindakan dan operasi berikut sebagai peristiwa dalam file CloudTrail log:

- CreateCertificateAuthority
- CreateCertificateAuthorityAuditReport
- CreatePermission
- DeleteCertificateAuthority
- DeletePermission
- DeletePolicy
- DescribeCertificateAuthority
- DescribeCertificateAuthorityReport
- GetCertificate
- GetCertificateAuthorityCertificate
- GetCertificateAuthorityCsr
- GetPolicy
- ImportCertificateAuthorityCertificate
- IssueCertificate
- ListCertificateAuthorities
- ListPermissions
- ListTags
- PutPolicy
- RestoreCertificateAuthority
- RevokeCertificate
- TagCertificateAuthority
- UntagCertificateAuthority
- UpdateCertificateAuthority
- `GenerateOCSPResponse`- Dipicu saat AWS Private CA menghasilkan respons OCSP.

- `SignCertificate`- Dihasilkan saat klien Anda menelepon IssueCertificate.
- `SignOCSPResponse`- Dihasilkan saat AWS Private CA menandatangani respons OCSP.
- `GenerateCRL`- Dihasilkan saat AWS Private CA menghasilkan daftar pencabutan sertifikat (CRL).
- `SignCACSR`- Dihasilkan saat AWS Private CA menandatangani permintaan penandatanganan sertifikat otoritas sertifikat (CA) (CSR).
- `SignCRL`- Dihasilkan saat AWS Private CA menandatangani CRL.

Contoh AWS Private CA peristiwa

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga file tersebut tidak muncul dalam urutan tertentu.

Berikut ini adalah contoh AWS Private CA CloudTrail peristiwa.

Contoh 1: Acara manajemen, **`IssueCertificate`**

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `IssueCertificate` tindakan.

```
{
   "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"
   }
```

```
}
```

Contoh 2: Acara manajemen, **ImportCertificateAuthorityCertificate**

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `ImportCertificateAuthorityCertificate` tindakan.

```
{
   "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"
}
```

Memecahkan masalah dengan AWS Private Certificate Authority

Konsultasikan topik berikut jika Anda memiliki masalah dalam menggunakan AWS Private Certificate Authority.

Topik

Memecahkan masalah pencabutan AWS Private CA sertifikat

Latensi respons OCSP

Responsivitas 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.

Pencabutan sertifikat yang ditandatangani sendiri

Anda tidak dapat mencabut sertifikat CA yang ditandatangani sendiri. Untuk mencabut sertifikat secara fungsional, hapus CA.

Memecahkan masalah pesan pengecualian AWS Private Certificate Authority

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.
`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 Dukungan Pusat untuk meminta kenaikan kuota.

Pengecualian Dikembalikan oleh AWS Private CA	Deskripsi	Remediasi
`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 Dukungan 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 Perbarui CA pribadi di AWS Private Certificate Authority.

Memecahkan masalah kesalahan sertifikat yang sesuai dengan AWS Private CA 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. Gunakan AWS Private CA untuk mengimplementasikan sertifikat Matter

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 Untuk informasi selengkapn lihat Aktifkan Root CA untuk Node Operational Certificates (NO 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

Kode kesalahar	Arti	Remediasi

		kegagalan validasi rantai. Namun, CustomExtensiondalam `API 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.

Amankan Kubernetes dengan AWS Private Certificate Authority

Anda dapat menggunakan AWS Private Certificate Authority untuk memberikan sertifikat untuk otentikasi aman dan enkripsi melalui TLS dan mTL. AWS Private CA menyediakan plugin open source, AWS Private CA Connector for Kubernetes, (`aws-privateca-issuer`) untuk add-on cert-manager yang diadopsi secara luas ke Kubernetes yang meminta sertifikat, mendistribusikannya ke rahasia Kubernetes, dan mengotomatiskan pembaruan sertifikat.

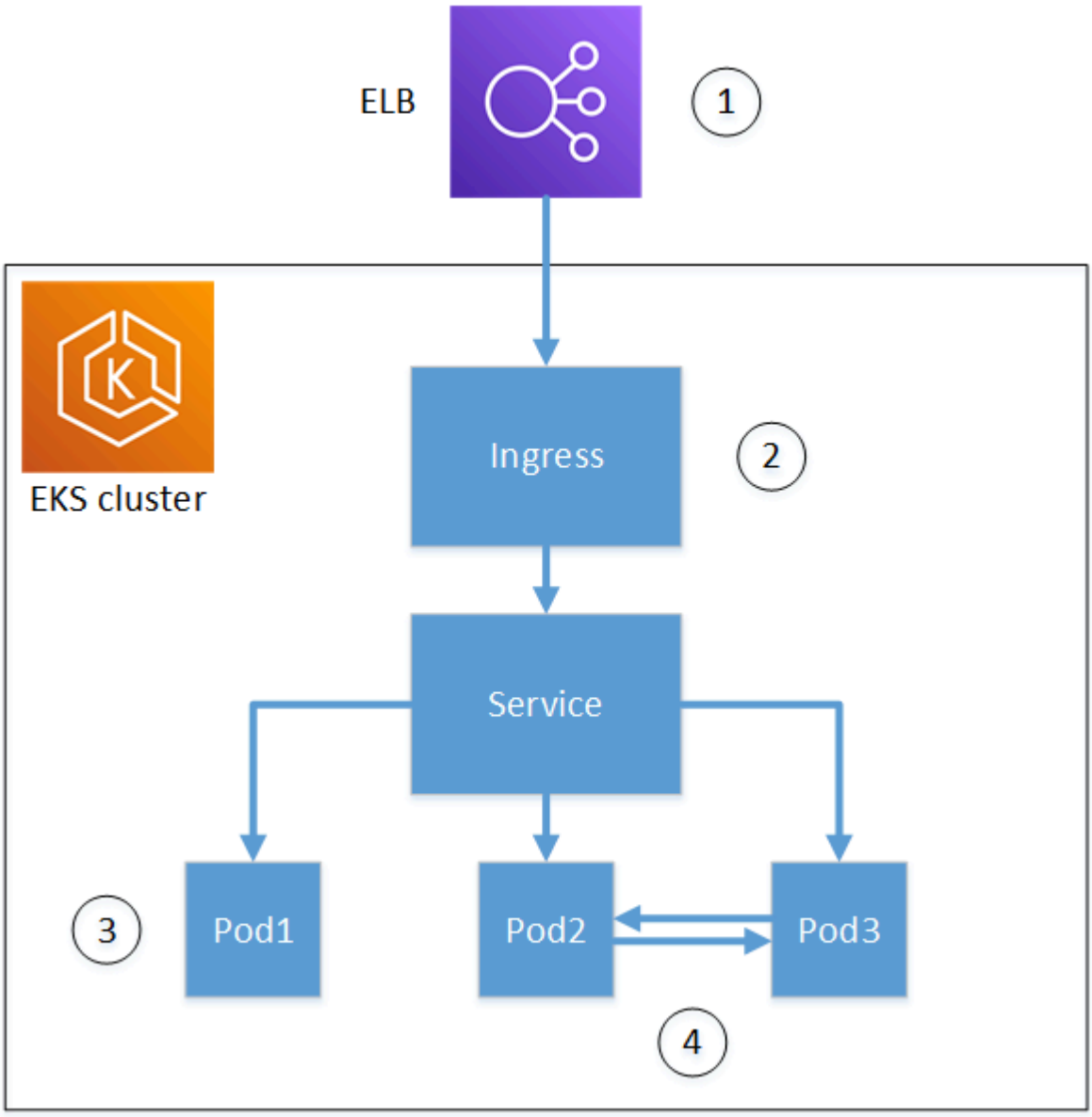
`aws-privateca-issuer`Plugin ini memungkinkan Anda untuk mengeluarkan AWS Private CA sertifikat melalui`cert-manager`. Anda dapat menggunakan plugin dengan Amazon Elastic Kubernetes Service (Amazon EKS), klaster Kubernetes yang dikelola sendiri, atau di klaster Kubernetes AWS di lokasi. Plugin ini bekerja pada arsitektur x86 dan ARM.

AWS Private CA memiliki kunci yang didukung HSM yang tidak dapat diekspor. Jika Anda memiliki persyaratan peraturan untuk mengontrol akses dan mengaudit operasi CA Anda, Anda dapat menggunakannya AWS Private CA untuk meningkatkan auditabilitas dan untuk mendukung kepatuhan.

> **Note**
>
> Jika Anda menjalankan Amazon EKS, kami sarankan Anda menggunakan `cert-manager` dan `aws-privateca-connector-for-kubernetes` add-on untuk pengalaman instalasi terkelola. Untuk informasi lebih lanjut, lihat AWS add-on.

Konsep

Diagram berikut menunjukkan beberapa opsi yang tersedia untuk menggunakan TLS di cluster Amazon EKS. Contoh cluster berada di belakang penyeimbang beban. Angka-angka tersebut mengidentifikasi kemungkinan titik akhir untuk komunikasi yang diamankan TLS.



1. Penghentian di penyeimbang beban

   Elastic Load Balancing (Elastic Load Balancing) terintegrasi dengan layanan. AWS Certificate Manager Anda tidak perlu menginstal `cert-manager` pada penyeimbang beban. Anda dapat menyediakan ACM dengan CA pribadi, menandatangani sertifikat dengan CA pribadi, dan menginstal sertifikat menggunakan konsol Elastic Load Balancing. AWS Private CA sertifikat diperpanjang secara otomatis.

   Sebagai alternatif, Anda dapat memberikan sertifikat pribadi ke penyeimbang AWS non-beban untuk mengakhiri TLS.

Ini menyediakan komunikasi terenkripsi antara klien jarak jauh dan penyeimbang beban. Data setelah penyeimbang beban diteruskan tanpa dienkripsi ke cluster Amazon EKS.

2. Penghentian pada pengontrol ingress Kubernetes

   Pengontrol ingress ada di dalam kluster Amazon EKS dan bertindak sebagai penyeimbang beban dan router. Untuk menggunakan pengontrol ingress sebagai titik akhir cluster untuk komunikasi eksternal, Anda harus:

   - Instal keduanya `cert-manager` dan `aws-privateca-issuer`
   - Menyediakan pengontrol dengan sertifikat pribadi TLS dari file. AWS Private CA

   Komunikasi antara penyeimbang beban dan pengontrol masuk dienkripsi, data melewati tidak terenkripsi ke sumber daya cluster.

3. Pengakhiran di pod

   Setiap pod adalah grup dari satu atau lebih kontainer yang berbagi penyimpanan dan sumber daya jaringan. Jika Anda menginstal keduanya `cert-manager` dan `aws-privateca-issuer` dan menyediakan klaster dengan CA pribadi, Kubernetes dapat menginstal sertifikat pribadi TLS yang ditandatangani pada pod sesuai kebutuhan. Koneksi TLS yang berakhir di sebuah pod tidak tersedia untuk pod lain di cluster secara default.

4. Mengamankan komunikasi antar pod.

   Anda dapat menyediakan beberapa pod dengan sertifikat untuk memungkinkan mereka berkomunikasi satu sama lain. Skenario berikut mungkin terjadi:

   - Penyediaan dengan Kubernetes menghasilkan sertifikat yang ditandatangani sendiri. Ini mengamankan komunikasi antar pod, tetapi sertifikat yang ditandatangani sendiri tidak memenuhi persyaratan HIPAA atau FIPS.
   - Penyediaan dengan sertifikat yang ditandatangani oleh. AWS Private CA Ini membutuhkan pemasangan keduanya `cert-manager` dan`aws-privateca-issuer`. Kubernetes kemudian dapat menginstal sertifikat mTLS yang ditandatangani pada pod sesuai kebutuhan.

Pertimbangan

Saat menggunakan AWS Private Certificate Authority Kubernetes, ingatlah pertimbangan berikut.

Penggunaan cross-account dari cert-manager

Administrator dengan akses lintas akun ke CA dapat menggunakan `cert-manager` add on untuk Kubernetes untuk menyediakan sertifikat untuk klaster menggunakan CA bersama. Untuk informasi lebih lanjut, lihatPraktik terbaik keamanan untuk akses lintas akun ke pribadi CAs.

Anda hanya dapat menggunakan templat AWS Private CA sertifikat tertentu dalam skenario lintas akun.

Tabel berikut mencantumkan AWS Private CA template yang dapat Anda gunakan dengan cert-manager untuk menyediakan klaster Kubernetes.

Template yang didukung untuk Kubernetes	Support untuk penggunaan lintas akun
BlankEndEntityCertificate_ CSRPassthrough / V1 definisi	Tidak
CodeSigningCertificate/V1 definisi	Tidak
EndEntityCertificate/V1 definisi	Ya
EndEntityClientAuthCertificate/V1 definisi	Ya
EndEntityServerAuthCertificate/V1 definisi	Ya
OCSPSigningDefinisi sertifikat/V1	Tidak

Mulai dengan AWS Private CA Connector for Kubernetes.

Topik berikut menunjukkan cara menggunakan AWS Private CA untuk mengamankan komunikasi di klaster Kubernetes. Untuk contoh lain, lihat Enkripsi dalam perjalanan untuk Kubernetes on. GitHub

Anda dapat menggunakan otoritas sertifikat pribadi untuk mengamankan komunikasi dengan kluster Amazon EKS Anda. Sebelum Anda mulai, pastikan Anda memiliki yang berikut:

- AWS Akun dengan izin yang sesuai yang tercakup pada kebijakan keamanan Anda.

Amazon EKS clusters

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAM",
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:AttachRolePolicy",
        "iam:GetRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "EKS",
      "Effect": "Allow",
      "Action": [
        "eks:CreateAddon",
        "eks:DescribeAddon",
        "eks:CreatePodIdentityAssociation",
        "eks:DescribeCluster"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::*:role/CertManagerPrivateCARole"
    }
  ]
}
```

Other clusters

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GetAndIssuePCACertificates",
      "Effect": "Allow",
      "Action": [
        "acm-pca:GetCertificate",
        "acm-pca:IssueCertificate"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RolesAnywhere",
      "Effect": "Allow",
      "Action": [
        "rolesanywhere:CreateProfile"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAM",
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:AttachRolePolicy"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::*:role/CertManagerPrivateCARole"
    }
  ]
}
```

- Sebuah cluster Kubernetes. Untuk membuat cluster Amazon Elastic Kubernetes Service, lihat panduan mulai cepat Amazon EKS. Untuk mempermudah, buat variabel lingkungan untuk menampung nama cluster:

```
export CLUSTER=aws-privateca-demo
```

- Wilayah AWS Tempat klaster CA dan Amazon EKS Anda berada. Untuk mempermudah, buat variabel lingkungan untuk menampung Wilayah:

```
export REGION=aws-region
```

- Nama Sumber Daya Amazon (ARN) dari otoritas sertifikat AWS Private CA swasta. Untuk mempermudah, buat variabel lingkungan untuk menampung CA ARN pribadi:

```
export CA_ARN="arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
```

  Untuk membuat CA pribadi, lihat https://docs.aws.amazon.com/privateca/latest/userguide/create-CA.html Buat CA pribadi di AWS Private CA

- Komputer dengan perangkat lunak berikut diinstal:
  - AWS CLI v2 dikonfigurasi
  - kubectl v1.13+
  - Untuk cluster EKS non-Amazon, Helm v3

Instal cert-manager

Untuk menggunakan CA pribadi, Anda harus menginstal `cert-manager>` add-on yang meminta sertifikat, mendistribusikannya, dan mengotomatiskan perpanjangan sertifikat. Anda juga harus menginstal `aws-private-ca-issuer` plugin yang memungkinkan Anda mengeluarkan sertifikat pribadi dari AWS Private CA. Gunakan langkah-langkah berikut untuk menginstal add-on dan plugin.

Amazon EKS clusters

Instal `cert-manager` sebagai add-on Amazon EKS:

```
aws eks create-addon \
  --cluster-name $CLUSTER \
```

```
  --addon-name cert-manager \
  --region $REGION
```

Other clusters

Instal `cert-manager` menggunakan Helm:

```
helm repo add jetstack https://charts.jetstack.io
helm repo update

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --set crds.enabled=true
```

Mengonfigurasi izin IAM

`aws-privateca-issuer`Plugin memerlukan izin berinteraksi dengan AWS Private CA. Untuk klaster Amazon EKS, Anda menggunakan identitas pod. Untuk cluster lain yang Anda gunakan AWS Identity and Access Management Roles Anywhere.

Tinju, buat kebijakan IAM. Kebijakan menggunakan kebijakan `AWSPrivateCAConnectorForKubernetesPolicy` terkelola. Untuk informasi selengkapnya tentang kebijakan ini, lihat AWSPrivateCAConnectorForKubernetesPolicydi Panduan referensi kebijakan AWS Terkelola.

Amazon EKS clusters

1. Buat file bernama `trust-policy.json` berisi kebijakan kepercayaan berikut:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "TrustPolicyForEKSClusters",
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
```

```
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

2. Jalankan perintah berikut untuk membuat peran IAM:

```
ROLE_ARN=$(aws iam create-role \
  --role-name CertManagerPrivateCARole \
  --assume-role-policy-document file://trust-policy.json \
  --region $REGION \
  --output text \
  --query "Role.Arn")

 aws iam attach-role-policy \
  --role-name CertManagerPrivateCARole \
  --policy-arn arn:aws:iam::aws:policy/AWSPrivateCAConnectorForKubernetesPolicy
```

Other clusters

1. Buat jangkar kepercayaan yang mempercayai CA pribadi yang disimpan. CA_ARN Untuk instruksi, lihat Memulai dengan IAM Roles Anywhere. Buat variabel lingkungan untuk menyimpan jangkar kepercayaan ARN:

```
export TRUST_ANCHOR_ARN=trustAnchorArn
```

2. Buat file bernama `trust-policy.json` berisi kebijakan kepercayaan berikut:

   JSON

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

```
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity",
                "sts:TagSession"
            ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": [
                        "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TRUST_ANCHOR_ARN"
                    ]
                },
                "StringEquals": {
                    "aws:PrincipalTag/x509Subject/CN": "aws-privateca-
issuer"
                }
            }
        }
    ]
}
```

3. Jalankan perintah berikut untuk membuat peran IAM:

```
ROLE_ARN=$(aws iam create-role \
  --role-name CertManagerPrivateCARole \
  --assume-role-policy-document file://trust-policy.json \
  --query "Role.Arn" \
  --region $REGION \
  --output text)

 aws iam attach-role-policy \
  --role-name CertManagerPrivateCARole \
  --region $REGION \
  --policy-arn arn:aws:iam::aws:policy/AWSPrivateCAConnectorForKubernetesPolicy
```

Instal dan konfigurasikan penerbit AWS Private CA cluster

Untuk menginstal `aws-privateca-connector-for-kubernetes` add-on, gunakan perintah berikut:

Amazon EKS clusters

Buat add-on:

```
aws eks create-addon --region $REGION \
  --cluster-name $CLUSTER \
  --addon-name aws-privateca-connector-for-kubernetes \
  --pod-identity-associations "[{
    \"serviceAccount\": \"aws-privateca-issuer\",
    \"roleArn\": \"$ROLE_ARN\"
  }]"
```

Kemudian tunggu hingga add-on aktif:

```
aws eks describe-addon \
  --cluster-name $CLUSTER \
  --addon-name aws-privateca-connector-for-kubernetes \
  --region $REGION \
  --query 'addon.status'
```

Other clusters

1. Buat profil di IAM Roles Anywhere:

   ```
   PROFILE_ARN=$(aws rolesanywhere create-profile \
     --name "privateca-profile" \
     --role-arns "$ROLE_ARN" \
     --region "$REGION" \
     --query 'profile.profileArn' \
     --enabled \
     --output text)
   ```

2. Buat sertifikat klien untuk digunakan dengan Konektor untuk Kubernetes dan untuk mengautentikasi IAM Roles Anywhere dengan: AWS Private CA

   a. Buat kunci pribadi untuk sertifikat klien:

   ```
   openssl genrsa -out client.key 2048
   ```

   b. Buat permintaan penandatanganan sertifikat (CSR) untuk sertifikat klien:

   ```
   openssl req -new \
   ```

```
  -key client.key \
  -out client.csr \
  -subj "/CN=aws-privateca-issuer"
```

c. Keluarkan sertifikat klien dari AWS Private CA:

```
CERT_ARN=$(aws acm-pca issue-certificate \
  --signing-algorithm SHA256WITHRSA \
  --csr fileb://client.csr \
  --validity Value=1,Type=DAYS \
  --certificate-authority-arn "$CA_ARN" \
  --region "$REGION" \
  --query 'CertificateArn' \
  --output text)
```

d. Simpan sertifikat klien secara lokal:

```
aws acm-pca get-certificate \
  --certificate-authority-arn $CA_ARN \
  --certificate-arn $CERT_ARN \
  --region $REGION \
  --query 'Certificate'
  --output text > pca-issuer-client-cert.pem
```

3. Instal AWS Private CA penerbit di cluster dengan sertifikat klien:

a. Tambahkan repositori Helm `awspca`:

```
helm repo add awspca https://cert-manager.github.io/aws-privateca-issuer
helm repo update
```

b. Buat namespace:

```
kubectl create namespace aws-privateca-issuer
```

c. Masukkan sertifikat yang dibuat sebelumnya menjadi rahasia:

```
kubectl create secret tls aws-privateca-credentials \
  -n aws-privateca-issuer \
  --cert=pca-issuer-client-cert.pem \
  --key=client.key
```

4. Instal AWS Private CA penerbit dengan IAM Roles Anywhere:

a. Buat file bernama `values.yaml` untuk mengonfigurasi plugin AWS Private CA penerbit untuk digunakan dengan IAM Roles Anywhere:

```
cat > values.yaml <<EOF
env:
  AWS_EC2_METADATA_SERVICE_ENDPOINT: "http://127.0.0.1:9911"

extraContainers:
  - name: "rolesanywhere-credential-helper"
    image: "public.ecr.aws/rolesanywhere/credential-helper:latest"
    command: ["aws_signing_helper"]
    args:
      - "serve"
      - "--private-key"
      - "/etc/cert/tls.key"
      - "--certificate"
      - "/etc/cert/tls.crt"
      - "--role-arn"
      - "$ROLE_ARN"
      - "--profile-arn"
      - "$PROFILE_ARN"
      - "--trust-anchor-arn"
      - "$TRUST_ANCHOR_ARN"
    volumeMounts:
      - name: cert
        mountPath: /etc/cert/
        readOnly: true

volumes:
  - name: cert
    secret:
      secretName: aws-privateca-credentials
EOF
```

b. Instal AWS Private CA penerbit dengan IAM Roles Anywhere:

```
helm install aws-privateca-issuer awspca/aws-privateca-issuer \
  -n aws-privateca-issuer \
  -f values.yaml
```

Tunggu penerbit siap. Gunakan perintah berikut ini.

```
kubectl wait --for=condition=ready pods --all -n aws-privateca-issuer --timeout=120s
```

Dan kemudian verifikasi instalasi untuk memastikan bahwa semua pod telah mencapai READY status:

```
kubectl -n aws-privateca-issuer get all
```

Untuk mengonfigurasi`aws-private-ca-cluster-issuer`, buat file YAMB bernama `cluster-issuer.yaml` berisi konfigurasi penerbit:

```
cat > cluster-issuer.yaml <<EOF
apiVersion: awspca.cert-manager.io/v1beta1
kind: AWSPCAClusterIssuer
metadata:
  name: aws-privateca-cluster-issuer
spec:
  arn: "$CA_ARN"
  region: "$REGION"
EOF
```

Selanjutnya, terapkan konfigurasi cluster:

```
kubectl apply -f cluster-issuer.yaml
```

Periksa status penerbit:

```
kubectl describe awspcaclusterissuer aws-privateca-cluster-issuer
```

Anda akan melihat respons yang mirip dengan berikut ini:

```
Status:
  Conditions:
    Last Transition Time:  2025-08-13T21:00:00Z
    Message:               AWS PCA Issuer is ready
    Reason:                Verified
    Status:                True
    Type:                  Ready
```

Mengelola sertifikat AWS Private CA klien dengan cert-manager

Jika Anda tidak menggunakan kluster Amazon EKS, setelah Anda mem-bootstrap sertifikat tepercaya secara manual, `aws-privateca-issuer` Anda dapat beralih ke sertifikat otentikasi klien yang dikelola oleh`cert-manager`. Hal ini memungkinkan `cert-manager` untuk secara otomatis memperbarui sertifikat otentikasi klien.

1. Buat file bernama`pca-auth-cert.yaml`:

```
cat > pca-auth-cert.yaml <<EOF
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: aws-privateca-client-cert
  namespace: aws-privateca-issuer
spec:
  secretName: aws-privateca-credentials
  duration: 168h
  renewBefore: 48h
  commonName: aws-privateca-issuer
  privateKey:
    algorithm: ECDSA
    size: 256
    rotationPolicy: Always
  usages:
    - client auth
  issuerRef:
    name: aws-privateca-cluster-issuer
    kind: AWSPCAClusterIssuer
    group: awspca.cert-manager.io
EOF
```

2. Buat sertifikat otentikasi klien terkelola baru:

```
kubectl apply -f pca-auth-cert.yaml
```

3. Validasi bahwa sertifikat telah dibuat:

```
kubectl get certificate aws-privateca-client-cert -n aws-privateca-issuer
```

   Anda akan melihat respons yang mirip dengan berikut ini:

```
NAME                          READY   SECRET                        AGE
aws-privateca-client-cert     True    aws-privateca-credentials     19m
```

Keluarkan sertifikat TLS pertama Anda

Sekarang setelah `cert-manager` dan `aws-privateca-issuer` diinstal, Anda dapat mengeluarkan sertifikat.

Buat file YAMM bernama `certificate.yaml` yang berisi sumber daya sertifikat:

```
cat > certificate.yaml <<EOF
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: example-certificate
  namespace: default
spec:
  secretName: example-certificate-tls
  issuerRef:
    name: aws-privateca-cluster-issuer
    kind: AWSPCAClusterIssuer
    group: awspca.cert-manager.io
  commonName: example.internal
  dnsNames:
    - example.internal
    - api.example.internal
  duration: 2160h # 90 days
  renewBefore: 360h # 15 days
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

Terapkan sertifikat menggunakan perintah berikut:

```
kubectl apply -f certificate.yaml
```

Anda kemudian dapat memeriksa status sertifikat dengan perintah berikut:

```
kubectl get certificate example-certificate
kubectl describe certificate example-certificate
```

Anda akan melihat respons yang mirip dengan ini:

```
NAME                  READY   SECRET                     AGE
example-certificate   True    example-certificate-tls    30s
```

Anda dapat memeriksa sertifikat yang dikeluarkan dengan perintah berikut:

```
kubectl get secret example-certificate-tls -o yaml
```

Anda juga dapat memecahkan kode dan memeriksa sertifikat dengan perintah berikut:

```
kubectl get secret example-certificate-tls -o jsonpath='{.data.tls\.crt}' | base64 -d |
 openssl x509 -text -noout
```

Contoh

Contoh berikut menunjukkan bagaimana AWS Private CA dapat digunakan dengan cluster Kubernetes.

- Contoh: Enkripsi dalam perjalanan untuk Kubernetes
- 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

Pantau Kubernetes dengan AWS Private CA

Untuk memantau CA pribadi klaster Kubernetes, gunakan teknik yang diuraikan dalam. Pantau AWS Private CA sumber daya Anda dapat menggunakan yang berikut ini untuk memantau CA pribadi:

- AWS Private CA CloudWatch metrik
- Monitor AWS Private CA dengan CloudWatch Acara
- Pencatatan panggilan AWS Private Certificate Authority API menggunakan AWS CloudTrail

Memecahkan masalah Kubernetes dengan AWS Private CA

Anda bisa mendapatkan log `aws-private-ca-issuer` dengan prosedur berikut:

1. Dapatkan nama pod:

```
kubectl get pods -A
```

2. Untuk melihat log penerbit, gunakan perintah berikut:

```
kubectl logs -n aws-privateca-issuer <pod-name> aws-privateca-issuer
```

3. Untuk melihat IAM Roles Anywhere log, gunakan perintah berikut:

```
kubectl logs -n aws-privateca-issuer <pod-name> rolesanywhere-credentials-helper
```

Untuk memeriksa status AWS Private CA penerbit Anda, gunakan salah satu dari berikut ini:

Untuk memeriksa apakah penerbit Anda siap, gunakan perintah berikut:

```
kubectl get AWSPCAClusterIssuers -o json | jq '.items[].status
```

Responsnya harus serupa dengan yang berikut:

```
{
  "conditions": [
    {
      "lastTransitionTime": "2024-07-03T13:56:37Z",
      "message": "Issuer verified",
      "reason": "Verified",
      "status": "True",
      "type": "Ready"
    }
  ]
}
```

Jika penerbit tidak berada di `Ready` negara bagian, `message` bidang tersebut memberikan informasi tentang mengapa penerbit tidak dapat mencapai negara bagian`Ready`.

Untuk memeriksa apakah sertifikat Anda sudah siap, gunakan perintah berikut:

```
kubectl get certificates -o json | jq '.items[].status'
```

Responsnya harus serupa dengan yang berikut:

```
{
  "conditions": [
    {
      "lastTransitionTime": "2024-07-03T13:58:13Z",
      "message": "Certificate is up to date and has not expired",
      "observedGeneration": 1,
      "reason": "Ready",
      "status": "True",
      "type": "Ready"
    }
  ],
  "notAfter": "2024-10-01T13:58:12Z",
  "notBefore": "2024-07-03T12:58:12Z",
  "renewalTime": "2024-09-16T13:58:12Z",
  "revision": 1
}
```

Jika sertifikat tidak ada di `Ready` negara bagian, `message` bidang memberikan informasi tentang mengapa sertifikat tidak dapat mencapai `Ready` negara bagian.

AWS Private CA Konektor untuk Active Directory

AWS Private CA dapat menerbitkan dan mengelola sertifikat yang diperlukan oleh AWS Managed Microsoft AD. Menggunakan AWS Private CA Konektor untuk Direktori Aktif (Konektor untuk AD), Anda dapat mengganti perusahaan lokal atau pihak ketiga lainnya CAs 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?

Jika Anda adalah pengguna pertama kali Connector for AD, kami sarankan Anda mulai dengan membaca bagian berikut:

- Apa itu AWS Private CA?
- Apa itu Directory Service?

Konektor Akses untuk AD

Anda dapat mengakses Konektor untuk AD melalui konsol, AWS CLI, dan APIs. Anda bisa mendapatkan akses ke konektor di konsol dari AWS Private CA konsol, dari Directory Service konsol Anda, atau dengan mencari Konektor untuk AD di bilah Konsol Manajemen AWS pencarian.

Harga

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 Authority Harga. Anda juga dapat menggunakan kalkulator AWS harga untuk memperkirakan biaya.

Siapkan Konektor untuk AD

Langkah-langkah di bagian ini adalah prasyarat untuk menggunakan Connector for AD. Ini mengasumsikan bahwa Anda telah membuat AWS akun. Setelah Anda menyelesaikan langkah-langkah di halaman ini, Anda dapat memulai dengan membuat konektor untuk AD.

Langkah 1: Buat CA pribadi menggunakan AWS Private CA

Siapkan otoritas sertifikat pribadi (CA) untuk menerbitkan sertifikat ke objek direktori Anda. Untuk informasi selengkapnya, lihat Otoritas sertifikat di AWS Private CA.

CA pribadi harus 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.

Langkah 2: Siapkan Direktori Aktif

Selain CA pribadi, Anda memerlukan direktori aktif di cloud pribadi virtual (VPC). Konektor untuk AD mendukung jenis direktori berikut yang ditawarkan oleh Directory Service:

- AWS Direktori Aktif Microsoft Terkelola: Dengan Directory Service Anda dapat menjalankan Microsoft Active Directory (AD) sebagai layanan terkelola. AWS Directory Service for Microsoft Active Directory juga disebut sebagai AWS Managed Microsoft AD, didukung oleh Windows Server 2019. Dengan AWS 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

(Hanya Konektor Direktori Aktif) Langkah 3: Delegasikan izin ke akun layanan

> **Note**
>
> Jika Anda menggunakan izin tambahan AWS Managed Microsoft AD akan didelegasikan secara otomatis saat Anda mengotorisasi Connector for AD service dengan direktori Anda. Anda dapat melewati langkah prasyarat ini.

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
```

- Membuat dan memperbarui objek NTAuth Certificates Certification Authority (CA). Catatan: jika objek NTAuth Certificates 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
```

PowerShell Skrip yang tersedia di Connector resmi untuk repositori Active Directory dapat digunakan untuk mendelegasikan izin tambahan yang diperlukan untuk akun layanan Directory Service AD Connector.

Skrip ini membuat objek otoritas sertifikasi NTAuth Sertifikat.

Untuk versi terbaru skrip dan detail penggunaan, lihat README di GitHub repositori.

Langkah 4: Buat 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:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "pca-connector-ad:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListCertificateAuthorities",
                "acm-pca:ListTags",
                "acm-pca:PutPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "acm-pca:IssueCertificate",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APIPassthrough/V*"
                },
                "ForAnyValue:StringEquals": {
```

```
                    "aws:CalledVia": "pca-connector-ad.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ds:AuthorizeApplication",
                "ds:DescribeDirectories",
                "ds:ListTagsForResource",
                "ds:UnauthorizeApplication",
                "ds:UpdateAuthorizedApplication"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ec2:DeleteTags",
                "ec2:CreateTags"
            ],
            "Resource": "arn:*:ec2:*:*:vpc-endpoint/*"
        }
    ]
}
```

Konektor untuk AD memerlukan AWS RAM izin tambahan, untuk penggunaan konsol dan baris perintah.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ram:CreateResourceShare",
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "ram:Principal": "pca-connector-ad.amazonaws.com",
                    "ram:RequestedResourceType": "acm-pca:CertificateAuthority"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:GetResourcePolicies",
                "ram:GetResourceShareAssociations",
                "ram:GetResourceShares",
                "ram:ListPrincipals",
                "ram:ListResources",
                "ram:ListResourceSharePermissions",
                "ram:ListResourceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

Langkah 5: Bagikan CA pribadi Anda dengan Connector for AD

Anda harus membagikan CA pribadi Anda 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 pembagian sumber daya menggunakan AWS 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 menelepon 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 izinnya menggunakan. `CalledVia`

Langkah 6: Buat pendaftaran direktori

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 Kelola pendaftaran direktori

Langkah 7: Konfigurasikan grup keamanan

Komunikasi antara VPC Anda dan konektor Konektor untuk AD sudah selesai AWS PrivateLink, yang memerlukan grup keamanan dengan aturan masuk yang membuka port 443 TCP 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).

Langkah 8: Konfigurasikan akses jaringan untuk objek direktori

Objek direktori memerlukan akses internet publik untuk memvalidasi Online Certificate Status Protocol (OCSP) dan daftar pencabutan sertifikat (CRLs) dari domain berikut:

```
*.windowsupdate.com
*.amazontrust.com
```

Aturan akses minimum yang diperlukan:

- Diperlukan untuk komunikasi OCSP dan CRL:

```
TCP 80: (HTTP) to 0.0.0.0/0
```

- Diperlukan untuk Konektor untuk AD:

```
TCP 443: (HTTPS) to 0.0.0.0/0
```

- Diperlukan untuk Active Directory:

```
TCP 88: (Kerberos) to Domain Controller IP range
TCP/UDP 389/636: (LDAP/LDAPS) to Domain Controller IP range, depending on Domain
 Controller configuration
TCP/UDP 53: (DNS) to 0.0.0.0/0
```

Jika perangkat tidak memiliki akses internet publik, penerbitan sertifikat akan gagal sebentar-sebentar dengan kode kesalahan `WS_E_OPERATION_TIMED_OUT`.

> **Note**
>
> Jika Anda mengonfigurasi grup keamanan untuk EC2 instans Amazon, itu tidak harus sama di Langkah 7.

Memulai dengan AWS Private CA Connector for Active Directory

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

Sebelum Anda mulai

Tutorial berikut memandu Anda melalui proses pembuatan konektor untuk AD dan template konektor. Untuk mengikuti tutorial ini, Anda harus terlebih dahulu memenuhi prasyarat yang tercantum di bagian ini.

Langkah 1: Buat konektor

Untuk membuat konektor, lihatMembuat konektor untuk Active Directory.

Langkah 2: Konfigurasikan kebijakan Microsoft Active Directory

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.

Saat menggunakan AWS Managed Microsoft AD, gunakan Directory Service enable-ca-enrollment-policy perintah untuk mengkonfigurasi GPOs pada pengontrol domain AWS Managed Microsoft AD instance.

Perintah berikut memungkinkan mendaftarkan pengontrol domain:

```
$  aws ds enable-ca-enrollment-policy \
    --pca-connector-arn MyPcaConnectorAdArn \
    --directory-id MyDirectoryId
```

Saat menggunakan AD Connector, gunakan langkah-langkah berikut 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 GPOs.

1. Buka Server Manager di DC
2. Buka Tools dan pilih Group Policy Management 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 dialog 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 GPOs 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.

Langkah 3: Buat template

Untuk membuat template, lihatBuat template konektor.

Langkah 4: Konfigurasikan izin grup Microsoft

Untuk mengonfigurasi izin grup Microsoft, lihatKelola Konektor untuk entri kontrol akses template AD.

AWS Private CA konektor untuk Active Directory

Prosedur di bagian ini menjelaskan cara membuat konektor Active Directory (AD), 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 AD AWS CLI, atau dengan menggunakan AWS Private CA Connector for AD API.

> **Note**
>
> Meskipun AWS Private CA Konektor untuk AD terintegrasi erat dengan AWS Private CA, kedua layanan tersebut terpisah APIs. Untuk informasi selengkapnya, lihat Referensi AWS

Private Certificate Authority API dan Referensi API AWS Private CA Connector for Active Directory.

Membuat konektor untuk Active Directory

Gunakan prosedur berikut untuk membuat konektor menggunakan konsol, baris perintah, atau API untuk AWS Private CA Connector for Active Directory.

Console

Untuk membuat konektor menggunakan 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**.

1. Pada halaman landing layanan pertama kali atau halaman Konektor untuk Direktori Aktif, pilih Buat konektor.
2. 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 oleh 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 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 Directory Service konsol:
     >
     > - AWS Microsoft 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 EC2 konsol Amazon 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.

3. Di bagian Jenis alamat IP, pilih dari opsi berikut:

   - IPv4- Memungkinkan IPv4 konektivitas ke layanan. Pilih opsi ini hanya jika semua subnet yang menghosting direktori Anda memiliki rentang IPv4 alamat.
   - Dualstack - Memungkinkan keduanya IPv4 dan IPv6 konektivitas ke layanan. Pilih opsi ini hanya jika semua subnet yang menghosting direktori Anda memiliki keduanya IPv4 dan rentang IPv6 alamat.

4. 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 CAs. CA baru Anda harus tersedia untuk dipilih.

5. 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.

6. 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.

API

Untuk membuat konektor menggunakan API

Untuk membuat konektor Active Directory dengan API, gunakan CreateConnectortindakan di AWS Private CA Connector for Active Directory API.

CLI

Untuk membuat konektor menggunakan 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 template konektor

Template adalah daftar konfigurasi untuk bagaimana sertifikat harus terlihat setelah dikeluarkan, dan bagaimana klien harus menangani sertifikat. Prosedur berikut menjelaskan cara membuat template.

Console

Untuk membuat template menggunakan konsol

1. 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**.
2. Pilih konektor dari daftar Konektor untuk Direktori Aktif dan kemudian pilih Lihat detail.
3. Pada halaman detail untuk konektor, temukan bagian Template dan kemudian pilih Buat template.
4. Pada halaman Buat template, di bagian Metode pembuatan Template, 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 kustom yang sebelumnya Anda buat.
- Mulai dari template kosong - Pilih opsi ini untuk mulai membuat template yang sama sekali baru.

5. Di bagian Pengaturan sertifikat, tentukan pengaturan berikut untuk sertifikat berdasarkan templat ini.

   - Jenis sertifikat - Tentukan apakah akan membuat sertifikat Pengguna atau Komputer.
   - Auto-enrollment - 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 beberapa 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)

6. Di bagian Penanganan permintaan sertifikat dan opsi pendaftaran, tentukan tujuan sertifikat berdasarkan templat, pilih 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

7. Di bagian Kebijakan aplikasi, pilih semua kebijakan aplikasi yang berlaku. Kebijakan yang tersedia tercantum di beberapa halaman. Beberapa kebijakan mungkin telah dipilih sebelumnya karena pengaturan sebelumnya.
8. Di bagian Kebijakan aplikasi kustom, Anda dapat menambahkan kustom OIDs ke template, dan menentukan apakah ekstensi kebijakan aplikasi penting.
9. Di bagian Pengaturan Kriptografi, pilih kategori pengaturan kriptografi berikut untuk sertifikat berdasarkan templat ini.
10. Di bagian Grup dan izin, Anda dapat melihat templat grup dan izin yang ada untuk pendaftaran, atau Anda dapat memilih tombol Tambahkan grup dan izin baru untuk menambahkan yang baru. Tombol membuka formulir yang membutuhkan informasi berikut:

    - Nama tampilan
    - Pengidentifikasi keamanan (SID)
    - Mendaftar, dengan opsi IZINKAN | DENY | NOT SET
    - Mendaftar otomatis, dengan opsi IZINKAN | DENY | NOT SET

11. Di bagian Supersede templates, Anda dapat memberi tahu Active Directory bahwa template saat ini menggantikan satu atau beberapa templat yang dibuat di AD. Terapkan template pengganti dengan memilih Add template dari Active Directory untuk menggantikan dan menentukan nama umum dari template pengganti.
12. 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.

13. Setelah memberikan informasi yang diperlukan dan meninjau pilihan Anda, pilih Buat template. Ini membuka detail 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.

API

Untuk membuat template konektor menggunakan API

Gunakan CreateTemplatetindakan di AWS Private CA Connector for Active Directory API.

CLI

Untuk membuat template konektor menggunakan AWS CLI

Gunakan perintah create-template di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

Perbarui template untuk Active Directory

Gunakan prosedur berikut untuk memperbarui template menggunakan konsol, baris perintah, atau API untuk AWS Private CA Connector for Active Directory.

Console

Untuk memperbarui template menggunakan 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**.

1. Pada daftar Konektor untuk Direktori Aktif, pilih konektor yang templatnya ingin Anda perbarui. Pilih Edit untuk melihat dan memodifikasi templat konektor.
2. Di halaman detail templat konektor Anda, pilih Edit. Ikuti petunjuk untuk membuat pembaruan Anda. Setelah selesai mengedit suatu area, pilih Simpan untuk menyimpan perubahan.

API

Untuk memperbarui template menggunakan API

Untuk memperbarui template Active Directory dengan API, gunakan UpdateTemplatetindakan di AWS Private CA Connector for Active Directory API.

CLI

Untuk memperbarui template menggunakan AWS CLI

Untuk memperbarui konektor untuk Active Directory dengan CLI, gunakan perintah update-template di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

Bagaimana Connector for Active Directory menyebarkan perubahan template Anda

AWS Private CA menerapkan template ke kebijakan Anda saat klien Anda menyegarkan cache kebijakan, yaitu setiap delapan jam. Ini termasuk perubahan pada entri kontrol akses grup templat. Saat klien Anda menyegarkan cache, ia menanyakan konektor untuk templat yang tersedia. Dalam hal penyegaran pendaftaran otomatis, klien mengeluarkan sertifikat yang cocok dengan salah satu atau kedua kondisi berikut:

- Sertifikat berada dalam periode pembaruan.
- Sertifikat tidak ada di perangkat klien.

Untuk penyegaran manual, klien akan menanyakan konektor, dan Anda harus mengatur template untuk diterbitkan.

Jika Anda melakukan debug, Anda dapat menghapus cache kebijakan secara manual untuk segera melihat perubahan templat. Untuk melakukannya, jalankan perintah Powershell berikut pada klien Anda.

```
certutil -f -user -policyserver * -policycache delete
```

Konektor daftar untuk Active Directory

Anda dapat menggunakan konsol AWS Private CA Connector for Active Directory atau AWS CLI untuk membuat daftar konektor yang Anda miliki.

Console

Untuk membuat daftar konektor Anda menggunakan konsol

1. 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**.
2. Tinjau informasi dalam daftar Konektor untuk Direktori Aktif. Anda dapat menavigasi melalui beberapa halaman konektor menggunakan nomor halaman di kanan atas. Setiap konektor menempati baris yang menampilkan kolom informasi berikut secara default.

- Connector ID — ID unik konektor.
- Nama direktori — Sumber daya Direktori Aktif yang terkait dengan konektor.
- Status konektor - Status konektor. Nilai yang mungkin adalah: Membuat | Aktif | Menghapus | Gagal.
- Status nama utama layanan — Status nama utama layanan (SPN) yang terkait dengan konektor. Nilai yang mungkin adalah: Membuat | Aktif | Menghapus | Gagal.
- Status pendaftaran direktori — Status pendaftaran direktur asosiasi. Nilai yang mungkin adalah: Membuat | Aktif | Menghapus | Gagal.
- Dibuat pada — Cap waktu pada pembuatan konektor.

Dengan memilih ikon roda gigi di sudut kanan atas konsol, Anda dapat menyesuaikan jumlah konektor yang ditampilkan pada halaman menggunakan preferensi ukuran Halaman.

API

Untuk membuat daftar konektor Anda menggunakan API

Gunakan ListConnectorstindakan di AWS Private CA Connector for Active Directory API.

CLI

Untuk membuat daftar konektor Anda menggunakan AWS CLI

Gunakan perintah daftar-konektor untuk membuat daftar konektor Anda.

Daftar templat konektor

Anda dapat menggunakan konsol AWS Private CA Connector for Active Directory atau AWS CLI untuk membuat daftar template untuk konektor yang Anda miliki. Template konektor didasarkan pada template AWS Private CA _BlankEndEntityCertificate_ APIPassthrough /V1.

Console

Untuk membuat daftar template Anda menggunakan konsol

1. 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**.
2. Pilih konektor dari daftar Konektor untuk Direktori Aktif dan kemudian pilih Lihat detail.

3. Pada halaman detail konektor, tinjau informasi di bagian Template. 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 template - Status template. Nilai yang mungkin adalah: Aktif | Menghapus.
- ID Template - Pengidentifikasi unik template.

API

Untuk membuat daftar konektor Anda menggunakan API

Gunakan ListTemplatestindakan di AWS Private CA Connector for Active Directory API untuk mencantumkan templat untuk konektor yang ditentukan.

CLI

Untuk membuat daftar konektor Anda menggunakan AWS CLI

Gunakan perintah daftar-templat untuk membuat daftar templat untuk konektor yang ditentukan.

Lihat detail konektor

Gunakan prosedur berikut untuk melihat detail konfigurasi konektor di konsol, baris perintah, atau API untuk AWS Private CA Connector for Active Directory.

Console

Untuk melihat detail untuk konektor menggunakan konsol

1. 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**.
2. Pilih konektor dari daftar Konektor untuk Direktori Aktif dan kemudian pilih Lihat 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 CA subjek
- AWS Private CA status
- Jenis alamat IP
- Titik akhir VPC dan grup keamanan

4. Di panel Template, Anda dapat membuat atau mengelola template yang terkait dengan konektor.
5. Dari panel Nama utama layanan (SPN), Anda dapat melihat nama prinsip layanan yang terkait dengan konektor.
6. Dari panel Pendaftaran Direktori, Anda dapat melihat atau mengubah pendaftaran direktori yang terkait dengan konektor.
7. Dari tag — panel opsional, Anda dapat membuat atau mengelola tag yang terkait dengan konektor.

API

Untuk membuat daftar konektor Anda menggunakan API

Gunakan GetConnectortindakan di AWS Private CA Connector for Active Directory API.

CLI

Untuk membuat daftar konektor Anda menggunakan AWS CLI

Gunakan perintah get-connector di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

Lihat detail templat konektor

Gunakan prosedur berikut untuk melihat detail konfigurasi template konektor menggunakan konsol, baris perintah, atau API untuk AWS Private CA Connector for Active Directory

Console

Untuk melihat detail untuk template konektor menggunakan konsol

1. 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**.
2. Pilih konektor dari daftar Konektor untuk Direktori Aktif dan kemudian pilih Lihat detail.
3. Pada halaman detail konektor, tinjau informasi di bagian Template, dan pilih templat yang ingin Anda periksa. Kemudian pilih Lihat detail.
4. Pada halaman detail, panel Detail Template 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 menggunakan tombol Edit, Hapus, dan Tindakan.

- Sunting
- Hapus
- Mengelola grup dan izin — Untuk informasi selengkapnya, lihat Mengonfigurasi grup dan izin.
- Mengelola template yang digantikan — Untuk informasi selengkapnya, lihat Meninjau dan membuat.
- Mengelola tag — Untuk informasi selengkapnya, lihatTagging Connector untuk sumber daya AD.
- Daftarkan ulang 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 selengkapnya, 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 templat ini. Untuk informasi selengkapnya, lihat Mengonfigurasi grup dan izin
   - Kebijakan aplikasi - Lihat dan kelola kebijakan aplikasi template. Untuk informasi selengkapnya, lihat Menetapkan kebijakan aplikasi.
   - Template yang digantikan - Lihat dan kelola template yang digantikan. Untuk informasi selengkapnya, lihat Meninjau dan membuat.
   - Tag opsional - Lihat dan kelola penandaan pada template ini. Untuk informasi selengkapnya, lihat Tagging Connector untuk sumber daya AD.

API

Untuk membuat daftar konektor Anda menggunakan API

Gunakan GetTemplatetindakan di AWS Private CA Connector for Active Directory API.

CLI

Untuk membuat daftar konektor Anda menggunakan AWS CLI

Gunakan perintah get-template di bagian AWS Private CA Connector for Active Directory dari. AWS CLI

Kelola pendaftaran direktori

Console

Untuk mengelola pendaftaran direktori menggunakan konsol

Pendaftaran direktori untuk konektor dapat dikelola dari tingkat atas AWS Private CA Konektor untuk konsol Direktori Aktif. Topik ini berjalan melalui opsi manajemen yang tersedia.

1. 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**.
2. Di area navigasi kiri, pilih Pendaftaran direktori.
3. Halaman pendaftaran Direktori menampilkan tabel direktori terdaftar dengan 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 dapat menggunakan direktori Register untuk membuat pendaftaran baru.
4. Anda dapat memilih salah satu pendaftaran yang terdaftar untuk mengelolanya. Ini memungkinkan Lihat rincian pendaftaran dan tombol direktori Deregister. Tombol Lihat detail pendaftaran membuka halaman detail untuk pendaftaran.
5. Panel detail pendaftaran Direktori menampilkan informasi berikut:
   - Nama situs domain direktori
   - ID Direktori — ID unik dari direktori. Memilih tautan akan membawa Anda ke AWS Directory Service konsol.
   - Jenis direktori
   - Status - Status direktori
   - Pendaftaran direktori ARN — Nama sumber daya Amazon dari pendaftaran direktori
   - Informasi status tambahan

6. Di panel Connectors and service principal name (SPNs), Anda dapat mengatur SPNs konektor. Untuk informasi selengkapnya, lihat Lihat detail konektor.
7. 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.

API

Untuk mengelola pendaftaran direktori menggunakan API

Create: CreateDirectoryRegistrationaction di AWS Private CA Connector for Active Directory API.

Ambil: GetDirectoryRegistrationtindakan di AWS Private CA Connector for Active Directory API.

Daftar: ListDirectoryRegistrationstindakan di AWS Private CA Connector for Active Directory API.

Hapus: DeleteDirectoryRegistrationtindakan di AWS Private CA Connector for Active Directory API.

CLI

Untuk mengelola pendaftaran direktori menggunakan CLI

Buat: Gunakan create-directory-registrationperintah di bagian AWS Private CA Connector for Active Directory dari file AWS CLI.

Ambil: get-directory-registrationperintah di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

List: list-directory-registrationsperintah di bagian AWS Private CA Connector for Active Directory dari file AWS CLI.

Hapus: delete-directory-registrationperintah di bagian AWS Private CA Connector for Active Directory dari file AWS CLI.

Kelola Konektor untuk entri kontrol akses template AD

Entri kontrol akses memberikan kontrol grup Active Directory mana yang dapat atau tidak dapat mendaftarkan sertifikat untuk Konektor khusus untuk templat AD. Bila Anda dapat membuat atau mengelola grup dan izin di Connector for AD, Anda harus memberikan pengenal Keamanan (SID) objek grup dari Active Directory. Anda dapat memperoleh SID menggunakan PowerShell perintah berikut. Untuk selengkapnya SIDs, lihat Cara kerja pengidentifikasi keamanan di dokumentasi Layanan Domain Direktori Microsoft.

```
$ Get-ADGroup -Identity "my_active_directory_group_name"
```

Prosedur berikut menggambarkan cara membuat dan mengelola Konektor untuk entri grup akses template AD.

Console

Untuk mengelola izin grup templat menggunakan konsol

Anda dapat mengelola grup dan izin untuk templat yang ada dapat dikelola dari halaman detail templat. 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.”

API

Untuk mengelola izin grup template menggunakan 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.

CLI

Untuk mengelola izin grup template menggunakan 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

Mengkonfigurasi nama utama layanan

Pelajari cara mengonfigurasi nama utama layanan untuk konektor.

Console

Untuk mengelola mengelola nama utama layanan menggunakan konsol

Nama utama layanan (SPN) dari konektor AD yang ada dapat dikelola dari halaman detail konektor. Untuk informasi selengkapnya, lihat Mengelola pendaftaran direktori Lihat detail konektor

API

Untuk mengelola nama utama layanan menggunakan API

Create: CreateServicePrincipalNameaction di AWS Private CA Connector for Active Directory API.

Ambil: GetServicePrincipalNametindakan di AWS Private CA Connector for Active Directory API.

Daftar: ListServicePrincipalNamestindakan di AWS Private CA Connector for Active Directory API.

Hapus: DeleteServicePrincipalNametindakan di AWS Private CA Connector for Active Directory API.

CLI

Untuk mengelola nama utama layanan menggunakan CLI

Buat: create-service-principal-nameperintah di bagian AWS Private CA Connector for Active Directory dari file AWS CLI.

Ambil: get-service-principal-nameperintah di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

List: list-service-principal-namesperintah di bagian AWS Private CA Connector for Active Directory dari file AWS CLI.

Hapus: delete-service-principal-nameperintah di bagian AWS Private CA Connector for Active Directory dari file AWS CLI.

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.

Console

Untuk mengelola penandaan sumber daya menggunakan 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

API

Untuk mengelola penandaan sumber daya menggunakan API

Tag: TagResourcetindakan di AWS Private CA Connector for Active Directory API.

List tags: ListTagsForResourceaction di AWS Private CA Connector for Active Directory API.

Untag: UntagResourcetindakan di AWS Private CA Connector for 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.

CLI

Untuk mengelola penandaan sumber daya menggunakan CLI

Tag: perintah tag-resource di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

List tags: list-tags-for-resourceperintah di bagian AWS Private CA Connector for Active Directory dari file AWS CLI.

Untag: perintah untag-resource di bagian AWS Private CA Connector for Active Directory dari file. AWS CLI

Mengintegrasikan Konektor untuk AD ke dalam aplikasi berbasis peristiwa menggunakan Amazon EventBridge

Anda dapat menggabungkan Connector for AD ke dalam aplikasi berbasis peristiwa (EDAs) yang menggunakan peristiwa yang terjadi di Connector for AD untuk berkomunikasi antar komponen aplikasi dan memulai proses hilir.

Misalnya, Anda dapat memanggil AWS layanan lain atau komponen khusus ketika peristiwa Konektor untuk AD berikut terjadi di akun Anda:

- Sertifikat dibuat atau saat pembuatan gagal.
- Sertifikat terdaftar, atau pendaftaran gagal.

Anda melakukan ini dengan menggunakan Amazon EventBridge untuk merutekan peristiwa dari Connector for AD ke komponen perangkat lunak lainnya. Amazon EventBridge adalah layanan tanpa server yang menggunakan peristiwa untuk menghubungkan komponen aplikasi bersama-sama,

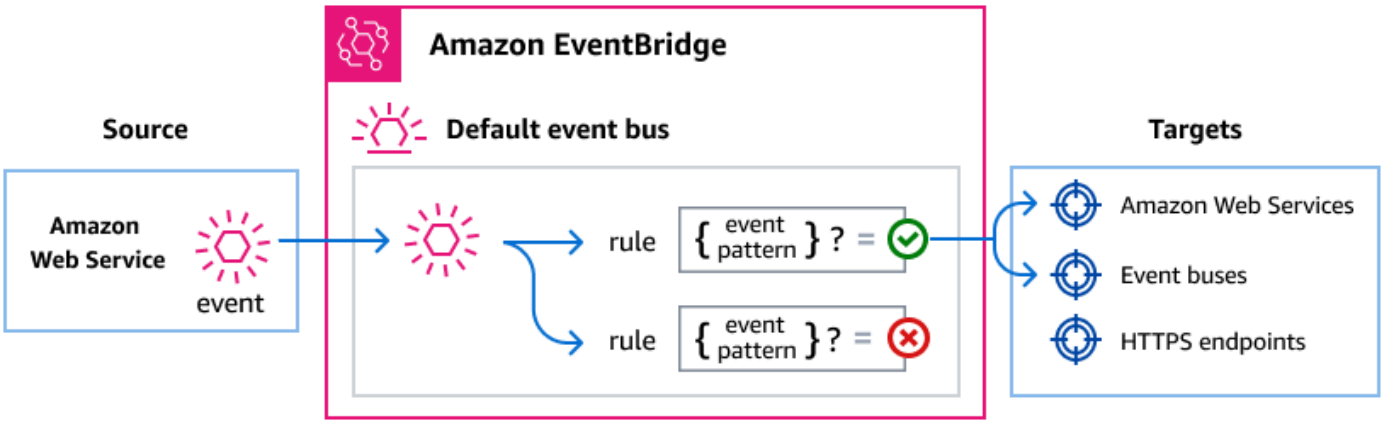
sehingga memudahkan Anda untuk mengintegrasikan AWS layanan seperti Connector for AD ke dalam arsitektur berbasis peristiwa tanpa kode dan operasi tambahan.

Cara EventBridge merutekan Konektor untuk acara AD

Berikut cara EventBridge kerja dengan Connector untuk acara AD:

Seperti banyak AWS layanan, Connector for AD menghasilkan dan mengirim acara ke bus acara EventBridge default. Bus acara adalah router yang menerima acara dan merutekan mereka ke tujuan, atau target, yang Anda tentukan. Target dapat mencakup AWS layanan lain, aplikasi khusus, dan aplikasi mitra SaaS.

EventBridge rute acara sesuai dengan aturan yang Anda buat di bus acara. Untuk setiap aturan, Anda menentukan filter, atau pola acara, untuk memilih hanya peristiwa yang Anda inginkan. Setiap kali acara dikirim ke bus acara, EventBridge bandingkan dengan setiap aturan. Jika acara cocok dengan aturan, EventBridge rutekan acara ke target yang ditentukan.



Konektor untuk acara AD

Untuk daftar Konektor untuk peristiwa AD yang dikirim ke EventBridge, lihat topik Konektor untuk AD di Referensi EventBridge Acara.

Struktur peristiwa

Semua peristiwa dari AWS layanan berisi dua jenis data:

- Seperangkat bidang umum yang berisi metadata tentang acara, seperti AWS layanan yang merupakan sumber acara, waktu acara dibuat, akun dan wilayah tempat acara berlangsung,

dan lain-lain. Untuk definisi bidang umum ini, lihat Struktur acara di Referensi EventBridge Acara Amazon.

- `detail`Bidang yang berisi data khusus untuk acara layanan tertentu.

Membuat pola acara yang cocok dengan Connector untuk acara AD

Pola acara adalah filter tempat menentukan data apa yang harus dikandung oleh peristiwa yang ingin Anda pilih.

Setiap pola acara adalah objek JSON yang berisi:

- `source`Atribut yang mengidentifikasi layanan yang mengirim acara. Untuk Konektor untuk acara AD, sumbernya adalah`aws.pca-connector-ad`.
- (Opsional): `detail-type` Atribut yang berisi array nama acara untuk dicocokkan.
- (Opsional): `detail` Atribut yang berisi data acara lain yang cocok.

Misalnya, pola peristiwa berikut akan memilih semua acara Berhasil Pendaftaran Kebijakan Sertifikat dari Konektor untuk AD:

```
{
  "source": ["aws.pca-connector-ad"],
  "detail-type": ["Certificate Policy Enrollment Succeeded"]
}
```

Untuk informasi selengkapnya tentang penulisan pola acara, lihat Pola acara di Panduan EventBridge Pengguna.

Menerima acara dari EventBridge

Anda dapat menentukan Konektor untuk sertifikat AD sebagai target aturan. Ini memungkinkan Konektor untuk AD menerima acara dari berbagai sumber, termasuk AWS layanan lain, aplikasi khusus, dan mitra SaaS. Untuk informasi selengkapnya, lihat Membuat aturan yang bereaksi terhadap peristiwa di Panduan EventBridge Pengguna.

Untuk daftar lengkap AWS layanan yang dapat Anda tentukan sebagai target, lihat Jenis target di Referensi EventBridge Acara.

Memecahkan masalah dengan AWS Private CA Konektor untuk Direktori Aktif

Gunakan informasi di sini untuk membantu Anda mendiagnosis dan memperbaiki AWS Private Certificate Authority Konektor untuk masalah AD.

Topik

Memecahkan Masalah Konektor untuk kode kesalahan AD

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	Penyebab galat	Remediasi
0x8FFFA000	Otentikasi Kerberos gagal.	Pastikan direktori Anda dapat dijangkau dan klien adalah pengguna atau komputer. Jika Anda menggunakan pendaftaran otomatis, perbaiki prinsip layanan AWS sumber daya Anda. Jika Anda menggunakan Active Directory UI untuk mendapatk an sertifikat, jalankan`gpupdate /force` .
0x8FFFA001	Pesan SOAP harus berisi header tindakan.	Tambahkan header tindakan.

Kode kesalahan	Penyebab galat	Remediasi
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.
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 maka 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.

Kode kesalahan	Penyebab galat	Remediasi
0x8FFFA007	Template menetapkan atribut email yang akan disertakan dalam subjek sertifikat atau nama alternatif subjek, tetapi atribut tidak ditemukan di objek AD untuk pemohon.	Tambahkan atribut email ke objek AD Anda
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.

Kode kesalahan	Penyebab galat	Remediasi
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.
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 mengeluarkan 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://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .	Tetapkan jenis permintaan ke`http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .

Kode kesalahan	Penyebab galat	Remediasi
0x8FFFA010	Pesan SOAP harus memiliki header to dari bidang konektor atau `CertificateEnrollmentPolicyServerEndpoint` bidang URI dalam respons XCEP.	Tetapkan header token keamanan permintaan ke `CertificateEnrollmentPolicyServerEndpoint` 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.
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.

Kode kesalahan	Penyebab galat	Remediasi
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.
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.

Memecahkan Masalah Konektor untuk 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.

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 ke direktori Anda. Tinjau Langkah 4: Buat 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. Jika menggunakan Connector for AD dengan AWS Directory Service AD Connector, pastikan kata sandi akun

Status kegagalan	Deskripsi	Remediasi
		layanan AD Connector tidak kedaluwarsa dan valid. Untuk informasi tentang akun layanan AD Connector , lihat Memulai AD Connector di Panduan Administrasi Konektor AD.
`INTERNAL_FAILURE`	Konektor untuk AD mengalami kegagalan internal.	Coba lagi nanti. Hapus konektor yang gagal dan buat yang baru.
`INSUFFICIENT_FREE_ADDRESSES`	Subnet VPC harus memiliki setidaknya satu alamat IP pribadi yang tersedia.	Pastikan bahwa ada alamat IP pribadi yang tersedia di subnet. Hapus konektor yang gagal dan buat yang baru.
`INVALID_SUBNET_IP_PROTOCOL`	Konektor untuk AD tidak dapat membuat titik akhir pada VPC Anda karena subnet yang terkait dengan direktori Anda tidak mendukung jenis alamat IP yang ditentukan.	Pastikan VPC dan subnet yang menghosting direktori Anda mendukung jenis alamat IP yang Anda pilih. Untuk informasi selengkapnya, lihat Jenis alamat IP. Hapus konektor yang gagal dan buat yang baru dengan jenis alamat IP yang didukung.

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 Langkah 4: Buat 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 Langkah 7: Konfigurasikan 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 Langkah 4: Buat 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.

Status kegagalan	Deskripsi	Remediasi
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.

Memecahkan Masalah Konektor untuk kegagalan pembuatan AD 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. Langkah 4: Buat 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

Status kegagalan	Deskripsi	Remediasi
		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.
`SPN_LIMIT_EXCEEDED`	Konektor untuk AD tidak dapat membuat SPN karena Anda telah mencapai batas SPNs per direktori. Jumlah maksimum SPNs per direktori adalah 10.	Hapus satu atau lebih SPNs dari akun Anda, dan coba buat SPN lagi.

Memecahkan Masalah Konektor untuk masalah pembaruan template AD

Jika Anda membuat perubahan pada templat atau entri kontrol akses grup, tetapi Anda tidak melihat perubahannya, ini mungkin disebabkan oleh cache kebijakan. AWS Private CA menerapkan template ke kebijakan Anda saat klien Anda menyegarkan cache kebijakan, yaitu setiap delapan jam. Saat klien Anda menyegarkan cache, ia menanyakan konektor untuk templat yang tersedia. Dalam hal penyegaran pendaftaran otomatis, klien mengeluarkan sertifikat yang cocok dengan salah satu atau kedua kondisi berikut:

- Sertifikat berada dalam periode pembaruan.
- Sertifikat tidak ada di perangkat klien.

Untuk penyegaran manual, klien akan menanyakan konektor, dan Anda harus mengatur template untuk diterbitkan.

Jika Anda melakukan debug, Anda dapat menghapus cache kebijakan secara manual untuk segera melihat perubahan templat. Untuk melakukannya, jalankan perintah Powershell berikut pada klien Anda.

```
certutil -f -user -policyserver * -policycache delete
```

AWS Private CA 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) yang populer dan dirancang untuk bekerja dengan klien atau titik akhir yang mendukung SCEP.

Topik

Fitur

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 Connector for 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.

Konektor Akses untuk SCEP

Anda dapat membuat, mengakses, dan mengelola Konektor untuk konektor SCEP menggunakan salah satu antarmuka berikut:

- Konsol Manajemen AWS- 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 SDKs- Menyediakan khusus bahasa APIs 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

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 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 Konfigurasikan sistem MDM Anda untuk Konektor untuk SCEP.

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.

Memahami Konektor untuk pertimbangan dan batasan SCEP

Ingatlah pertimbangan dan batasan berikut saat menggunakan Konektor untuk SCEP.

Pertimbangan-pertimbangan

Mode operasi CA

Anda hanya dapat menggunakan Konektor untuk SCEP dengan pribadi CAs 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, lihatMemahami mode AWS Private CA CA.

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 apa pun dan SANs, 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.

CSRs

- Jika permintaan penandatanganan sertifikat (CSR) yang dikirim ke Connector for SCEP tidak menyertakan ekstensi Extended Key Usage (EKU), kami akan menetapkan nilai EKU ke. `clientAuthentication` Untuk informasi, lihat 4.2.1.12. Penggunaan Kunci yang Diperpanjang di RFC 5280.
- Kami mendukung `ValidityPeriod` dan atribut `ValidityPeriodUnits` khusus di CSRs. 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.

Pribadi bersama CAs

Anda hanya dapat menggunakan Konektor untuk SCEP dengan pribadi CAs di mana Anda adalah pemiliknya.

Siapkan Konektor untuk SCEP

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.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "pca-connector-scep:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListCertificateAuthorities",
                "acm-pca:ListTags",
                "acm-pca:PutPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "acm-pca:IssueCertificate",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APICSRPassthrough/V*"
                },
                "ForAnyValue:StringEquals": {
```

```
                    "aws:CalledVia": "pca-connector-scep.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:CreateResourceShare",
                "ram:GetResourcePolicies",
                "ram:GetResourceShareAssociations",
                "ram:GetResourceShares",
                "ram:ListPrincipals",
                "ram:ListResources",
                "ram:ListResourceSharePermissions",
                "ram:ListResourceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

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 CRLs CA pribadi Anda untuk memungkinkan pencabutan. Untuk informasi selengkapnya, lihat Rencanakan metode pencabutan AWS Private CA sertifikat Anda.
- 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, lihatBuat CA pribadi di AWS Private CA.

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 nilainya. *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 Connector for SCEP

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

Tutorial berikut memandu Anda melalui proses pembuatan 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. Siapkan 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 tag 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 pribadi CAs. 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 tag 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. Konfigurasikan sistem MDM Anda untuk Konektor untuk SCEP
   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 pribadi CAs. 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 apa pun dan SANs, 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 Konfigurasikan sistem MDM Anda untuk Konektor untuk SCEP.

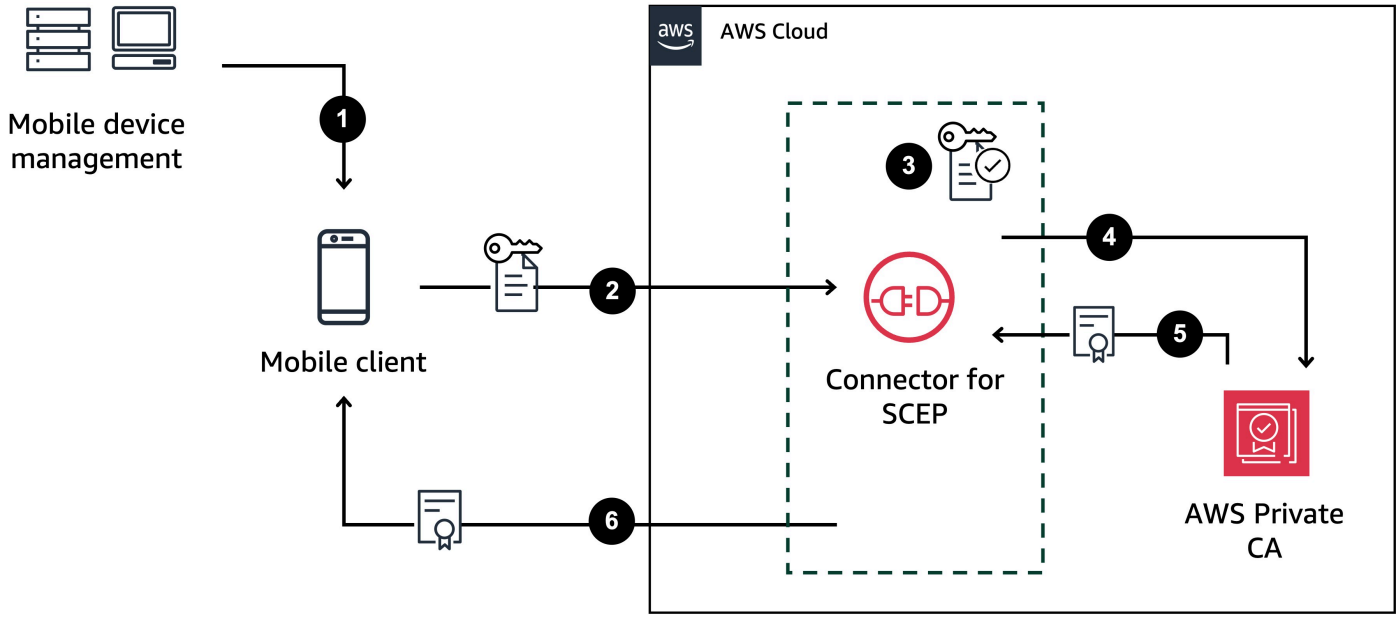
Konfigurasikan sistem MDM Anda untuk Konektor untuk SCEP

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 sistem manajemen perangkat seluler (MDM) yang populer, termasuk Microsoft Intune, Omnissa Workspace ONE dan Jamf Pro, untuk mendaftarkan perangkat seluler. Ini dirancang untuk 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, dan cara mengonfigurasi sistem MDM Anda untuk menggunakannya.

Konektor tujuan umum

Konektor tujuan umum dirancang untuk bekerja dengan titik akhir perangkat seluler yang mendukung SCEP, kecuali untuk Microsoft Intune, yang memiliki konektor khusus. Dengan konektor tujuan umum, seperti Jamf Pro atau Omnissa Workspace ONE, Anda mengelola kata sandi tantangan SCEP. Diagram berikut menggunakan sistem manajemen perangkat seluler (MDM) sebagai contoh, tetapi fungsionalitas yang sama berlaku untuk sistem atau perangkat berkemampuan SCEP lainnya.

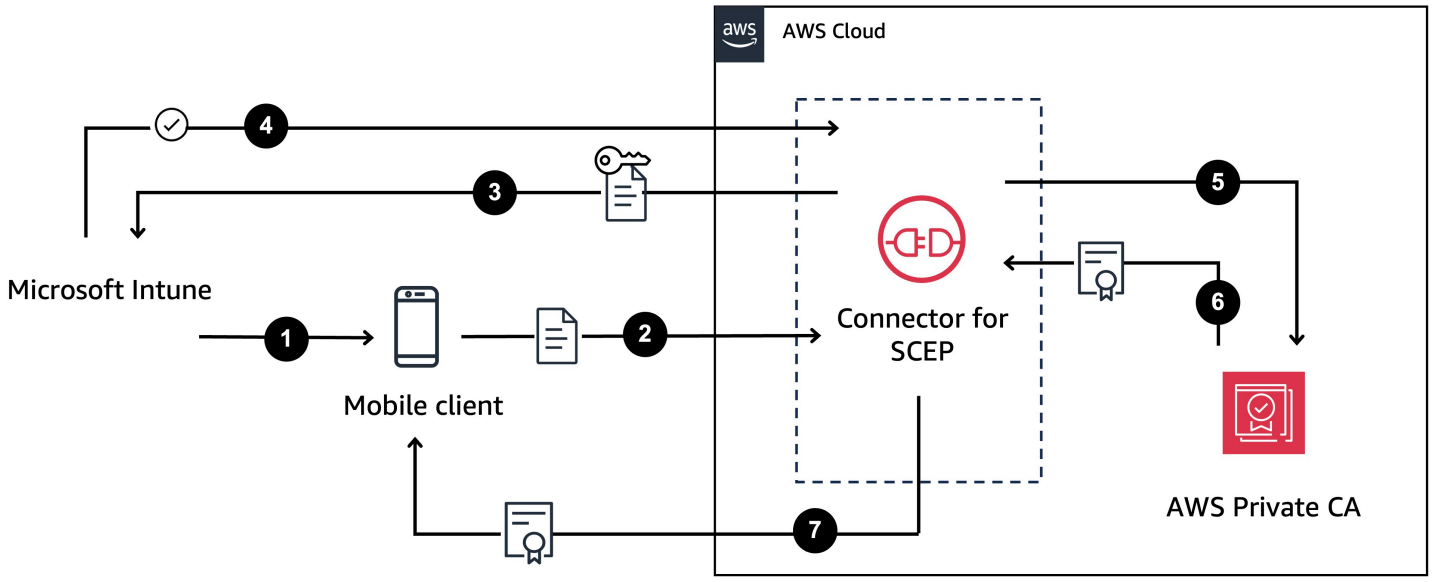


1. Sistem MDM (atau perangkat atau sistem lain) mengirimkan profil SCEP ke klien seluler. Profil SCEP berisi parameter konfigurasi yang menentukan profil sertifikat, seperti periode validitas sertifikat, kata sandi tantangan, dan informasi lain yang relevan dengan penerbitan sertifikat.
2. Klien seluler 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 CA 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 Konektor untuk SCEP dengan Microsoft Intune, lihat. Konfigurasikan Microsoft Intune untuk Konektor untuk SCEP

Untuk menggunakan Connector for SCEP dengan Microsoft Intune, Anda harus mengaktifkan fungsionalitas tertentu menggunakan Microsoft Intune API, dan memiliki lisensi Microsoft Intune yang valid. 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.

Topik

- Konfigurasikan Jamf Pro untuk Konektor untuk SCEP
- Konfigurasikan Microsoft Intune untuk Konektor untuk SCEP
- Konfigurasikan Omnissa Workspace ONE untuk Konektor untuk SCEP

Konfigurasikan Jamf Pro untuk Konektor untuk SCEP

Anda dapat menggunakan AWS Private CA sebagai otoritas sertifikat eksternal (CA) dengan sistem manajemen perangkat seluler Jamf Pro (MDM). Panduan ini memberikan petunjuk tentang cara mengkonfigurasi Jamf Pro setelah Anda membuat konektor tujuan umum.

Konfigurasikan Jamf Pro untuk Konektor untuk SCEP

Panduan ini memberikan petunjuk tentang cara mengkonfigurasi Jamf Pro untuk digunakan dengan Konektor untuk SCEP. Setelah berhasil mengonfigurasi Jamf Pro dan Connector untuk SCEP, Anda akan dapat mengeluarkan AWS Private CA sertifikat ke perangkat yang dikelola.

Persyaratan Jamf Pro

Implementasi Jamf Pro Anda harus memenuhi persyaratan berikut.

- 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.

Langkah 1: (Opsional - disarankan) Dapatkan sidik jari CA pribadi Anda

Sidik jari adalah pengenal unik untuk CA pribadi Anda yang dapat digunakan untuk memverifikasi identitas CA Anda saat membangun kepercayaan dengan sistem atau aplikasi lain. Memasukkan sidik jari otoritas sertifikat (CA) memungkinkan perangkat yang dikelola untuk mengotentikasi CA yang mereka sambungkan dan meminta sertifikat semata-mata dari CA yang diantisipasi. Sebaiknya gunakan sidik jari CA dengan Jamf Pro.

Untuk menghasilkan sidik jari untuk CA pribadi Anda

1. Dapatkan sertifikat CA pribadi baik dari AWS Private CA konsol atau dengan menggunakan GetCertificateAuthorityCertificate. Simpan sebagai `ca.pem` file.
2. Instal Utilitas Baris Perintah OpenSSL.
3. Di OpenSSL, jalankan perintah berikut untuk menghasilkan sidik jari:

```
openssl x509 -in ca.pem -sha256 -fingerprint
```

Langkah 2: Konfigurasikan AWS Private CA sebagai CA eksternal di Jamf Pro

Setelah Anda membuat konektor untuk SCEP, Anda harus menetapkan AWS Private CA sebagai otoritas sertifikat eksternal (CA) 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 otoritas sertifikat eksternal (CA) 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 dari konektor Anda, dan tempelkan ke bidang Tantangan. Konektor dapat memiliki beberapa kata sandi 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 GetChallengePassworddan menyalin `Password` nilai dari respons. Untuk informasi tentang menggunakan kata sandi tantangan, lihatMemahami Konektor untuk pertimbangan dan batasan SCEP.
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 - disarankan) Masukkan string hex di bidang Sidik Jari. Kami menyarankan Anda menambahkan sidik jari CA untuk memungkinkan perangkat terkelola memverifikasi CA, dan hanya meminta sertifikat dari CA. Untuk petunjuk tentang cara membuat sidik jari untuk CA pribadi Anda, lihatLangkah 1: (Opsional - disarankan) Dapatkan sidik jari CA pribadi Anda.
17. Pilih Simpan.

Langkah 3: Siapkan sertifikat penandatanganan profil konfigurasi

Untuk menggunakan Jamf Pro dengan Konektor untuk SCEP, 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.

Berikut adalah langkah-langkah untuk membuat keystore sertifikat dan mengunggahnya ke Jamf Pro:

- Buat permintaan penandatanganan sertifikat (CSR) menggunakan proses internal Anda.
- Dapatkan CSR yang ditandatangani oleh CA pribadi yang terkait dengan konektor Anda.
- Buat keystore sertifikat penandatanganan profil yang berisi penandatanganan profil dan sertifikat CA.
- Unggah keystore sertifikat ke Jamf Pro.

Dengan mengikuti langkah-langkah ini, Anda dapat memastikan bahwa perangkat Anda dapat memvalidasi dan mengautentikasi profil konfigurasi yang ditandatangani oleh CA pribadi Anda, memungkinkan penggunaan Konektor untuk SCEP dengan Jamf Pro.

1. Contoh berikut menggunakan OpenSSL AWS Certificate Manager dan, tetapi Anda dapat membuat permintaan penandatanganan sertifikat menggunakan metode pilihan Anda.

   AWS Certificate Manager console

   Untuk membuat sertifikat penandatanganan profil menggunakan konsol ACM

   1. Gunakan ACM untuk meminta sertifikat PKI pribadi. Sertakan yang berikut ini:

- Jenis - Gunakan jenis CA pribadi yang sama yang berfungsi sebagai otoritas sertifikat SCEP untuk sistem MDM Anda.
- Di bagian Detail otoritas sertifikat, pilih menu Otoritas sertifikat dan pilih CA pribadi yang berfungsi sebagai CA untuk Jamf Pro.
- Nama domain - Berikan nama domain untuk disematkan ke dalam sertifikat. Anda dapat menggunakan nama domain yang sepenuhnya memenuhi syarat (FQDN), sepertiwww.example.com, atau nama domain telanjang atau puncak seperti example.com (yang tidak termasuk). www.

2. Gunakan ACM untuk mengekspor sertifikat pribadi yang Anda buat pada langkah sebelumnya. Pilih Ekspor file untuk sertifikat, rantai sertifikat, dan kunci terenkripsi. Simpan Passphrase berguna karena Anda akan membutuhkannya di langkah berikutnya.
3. Di terminal, jalankan perintah berikut di folder yang berisi file yang diekspor untuk menulis bundel PKCS #12 ke dalam output.p12 file yang dikodekan oleh frasa sandi yang Anda buat pada langkah sebelumnya.

```
openssl pkcs12 -export \
  -in "Exported Certificate.txt" \
  -certfile "Certificate Chain.txt" \
  -inkey "Exported Certificate Private Key.txt" \
  -name example \
  -out output.p12 \
  -passin pass:your-passphrase \
  -passout pass:your-passphrase
```

AWS Certificate Manager CLI

Untuk membuat sertifikat penandatanganan profil menggunakan ACM CLI

- Perintah berikut menunjukkan cara membuat sertifikat di ACM, dan kemudian mengekspor file sebagai bundel PKCS #12.

```
PCA=<Enter your Private CA ARN>

CERTIFICATE=$(aws acm request-certificate \
    --certificate-authority-arn $PCA \
    --domain-name <any valid domain name, such as test.name> \
```

```
    | jq -r '.CertificateArn')

while [[ $(aws acm describe-certificate \
  --certificate-arn $CERTIFICATE \
  | jq -r '.Certificate.Status') != "ISSUED" ]] do sleep 1; done

aws acm export-certificate \
  --certificate-arn $CERTIFICATE \
  --passphrase password | jq -r '.Certificate' > Certificate.pem
aws acm export-certificate \
  --certificate-arn $CERTIFICATE \
  --passphrase password | jq -r '.CertificateChain' > CertificateChain.pem
aws acm export-certificate \
  --certificate-arn $CERTIFICATE \
  --passphrase password | jq -r '.PrivateKey' > PrivateKey.pem

openssl pkcs12 -export \
  -in "Certificate.pem" \
  -certfile "CertificateChain.pem" \
  -inkey "PrivateKey.pem" \
  -name example \
  -out output.p12 \
  -passin pass:passphrase \
  -passout pass:passphrase
```

OpenSSL CLI

Untuk membuat sertifikat penandatanganan profil menggunakan OpenSSL CLI

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 sebelumnya. Jalankan perintah berikut, dan perhatikan sertifikat ARN dalam respons.

```
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. Tentukan sertifikat ARN dari langkah sebelumnya.

```
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. Gunakan file CRT yang Anda buat dalam langkah empat dan lima.

```
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 untuk diberikan kepada Jamf Pro.

2. Di Jamf Pro, arahkan ke Template Sertifikat Manajemen dan buka panel CA Eksternal.
3. Di bagian bawah panel CA Eksternal, pilih Ubah Penandatanganan dan Sertifikat CA.
4. Ikuti petunjuk di layar untuk mengunggah sertifikat penandatanganan dan CA untuk CA eksternal.

Langkah 4: (Opsional) Instal sertifikat selama pendaftaran yang dimulai pengguna

Untuk membangun kepercayaan antara perangkat klien Anda dan CA pribadi Anda, Anda harus memastikan perangkat Anda mempercayai sertifikat yang dikeluarkan oleh Jamf Pro. Anda dapat menggunakan Pengaturan Pendaftaran yang Dimulai Pengguna Jamf Pro untuk menginstal sertifikat CA Anda AWS Private CA secara otomatis di perangkat klien saat mereka meminta sertifikat selama proses pendaftaran.

Memecahkan masalah kegagalan instalasi profil

Jika Anda mengalami kegagalan penginstalan profil setelah mengaktifkan Gunakan Jamf Pro sebagai Proxy SCEP untuk pendaftaran komputer dan perangkat seluler, lihat log perangkat Anda dan coba yang berikut ini.

Pesan kesalahan log perangkat	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 SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:14006>`	Kata sandi tantangan Anda mungkin salah dikonfigurasi. Verifikasi bahwa kata sandi tantangan di Jamf Pro cocok dengan kata sandi tantangan konektor Anda.

Konfigurasikan Microsoft Intune untuk Konektor untuk SCEP

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.

Langkah 1: Berikan AWS Private CA izin untuk menggunakan Aplikasi Microsoft Entra ID Anda

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. Untuk Jenis, pilih Pengidentifikasi subjek eksplisit.
9. Salin dan tempel nilai subjek OpenID dari konektor Anda ke bidang Nilai. 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.
10. (Opsional) Masukkan nama instance di bidang Nama. Misalnya, Anda bisa menamainya AWS Private CA.
11. (Opsional) Masukkan deskripsi ke dalam bidang Deskripsi.
12. 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.
13. Pilih Tambahkan.

Langkah 2: Siapkan 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 Server SCEP URLs, salin dan tempel URL SCEP Publik dari detail konektor Anda ke bidang Server SCEP. URLs Untuk melihat detail konektor, pilih konektor dari daftar Konektor untuk SCEP. Atau, Anda bisa mendapatkan URL dengan menelepon ListConnectors, 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.

Langkah 3: 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.

Konfigurasikan Omnissa Workspace ONE untuk Konektor untuk SCEP

Anda dapat menggunakan AWS Private CA sebagai otoritas sertifikat eksternal (CA) dengan sistem Omnissa Workspace ONE UEM (Unified Endpoint Management). Panduan ini memberikan petunjuk tentang cara mengkonfigurasi Omnissa Workspace ONE setelah Anda membuat konektor SCEP. AWS

Prasyarat

Sebelum Anda membuat konektor SCEP untuk Omnissa Workspace ONE, Anda harus menyelesaikan prasyarat berikut:

- Buat CA pribadi di AWS konsol. Untuk informasi selengkapnya, lihat Buat CA pribadi di AWS Private CA.
- Buat konektor SCEP tujuan umum. Untuk informasi selengkapnya, lihat Membuat konektor.
- Memiliki akun admin lingkungan Omnissa Workspace ONE yang aktif dengan ID Grup Organisasi.
- Jika Anda mendaftarkan perangkat Apple, konfigurasikan Layanan Pemberitahuan Push Apple (APNs) untuk MDM. Untuk informasi selengkapnya, lihat APNs Sertifikat di dokumentasi Omnissa.

Langkah 1: Tentukan otoritas sertifikat dan template di Omnissa Workspace ONE

Setelah membuat konektor CA dan SCEP pribadi di AWS konsol, tentukan otoritas sertifikat dan templat di Omnissa Workspace ONE.

Tambahkan AWS Private CA sebagai otoritas sertifikat

1. Dari menu Sistem, pilih Integrasi Perusahaan dan kemudian pilih Otoritas Sertifikat.
2. Pilih + ADD dan berikan informasi berikut:
   - Nama: AWS-Private-CA.
   - Deskripsi: AWS Private CA untuk penerbitan sertifikat perangkat.
   - Jenis Otoritas: Pilih SCEP Generik.
   - URL SCEP: Masukkan URL SCEP dari. AWS Private CA
   - Jenis Tantangan: Pilih STATIC.
   - Tantangan Statis: Masukkan kata sandi tantangan statis SCEP dari Konektor untuk konfigurasi SCEP di AWS konsol.
   - Masukkan nilai Try Timeout dan Max Retries.
3. Simpan konfigurasi.

Buat templat sertifikat

1. Dari menu Sistem, pilih Integrasi Perusahaan, pilih Otoritas Sertifikat, lalu pilih Template.
2. Pilih Tambahkan Template dan berikan informasi berikut:
   - Nama Templat: Device-Cert-Template.
   - Otoritas Sertifikat: Pilih AWS-Private-CA.

- Nama Subjek: Ini adalah bidang yang dapat disesuaikan. Anda dapat memilih nilai variabel dari daftar atribut. Misalnya, CN= {DeviceReportedName}, O = {}, DevicePlatform OU = {1} CustomAttribute
- Panjang Kunci Pribadi: 2048 bit.
- Jenis Kunci Pribadi: Pilih Penandatanganan dan Enkripsi sesuai kebutuhan
- Pembaruan Otomatis: Enabled/Disabled (Berdasarkan kebutuhan Anda).

3. Simpan template.

Langkah 2: Siapkan konfigurasi profil Omnissa Workspace ONE UEM

Buat profil di Omnissa Workspace ONE UEM yang mengarahkan perangkat ke Connector agar SCEP mengeluarkan sertifikat.

Membuat profil perangkat SCEP untuk distribusi sertifikat

1. Dari menu Sumber Daya, pilih Profil & Garis Dasar, lalu pilih Profil.
2. Pilih Tambah lalu Tambahkan Profil
3. Pilih platform perangkat (Android, iOS, macOS, Windows).
4. Tetapkan tipe Manajemen dan Konteks yang sesuai.
5. Tetapkan Nama: Device-Cert-Profile.
6. Gulir ke SCEP Payload.
7. Pilih SCEP dan kemudian pilih +Add.
8. Gunakan konfigurasi berikut:
   - SCEP:
     - Untuk Sumber Kredenal pilih Otoritas Sertifikat yang Ditetapkan (Default).
     - Untuk Otoritas Sertifikat pilih AWS-Private-CA
     - Untuk Templat Sertifikat pilih Device-Cert-Template yang ditentukan dalam Langkah 1.
9. Pilih Berikutnya dan di bagian Penugasan pilih grup pintar yang tepat dari daftar (grup penugasan untuk perangkat).
10. Pilih Jenis Penugasan sebagai Otomatis untuk mengaktifkan perpanjangan otomatis.
11. Simpan dan publikasikan profil.

> **Note**
>
> Untuk informasi selengkapnya, lihat SCEP di dokumentasi Omnissa.

Langkah 3: Daftarkan perangkat di Omnissa Workspace ONE

Membuat atau memverifikasi grup pintar

1. Dari Grup & Pengaturan pilih Grup dan kemudian pilih Grup Penugasan.
2. Membuat atau mengedit grup pintar perangkat POC:
   - Nama: Perangkat POC.
   - Jenis Perangkat: Pilih Semua atau platform tertentu (Android atau iOS, misalnya).
   - Kriteria: Gunakan UserGroup, Platform dan OS, OEM dan Model untuk menentukan kriteria untuk mengelompokkan perangkat target.
   - Kepemilikan: Pilih Apa saja untuk perangkat pribadi atau perusahaan.
3. Simpan dan verifikasi perangkat target muncul di tab Pratinjau.

Pendaftaran perangkat manual

Android

- Unduh aplikasi Workspace ONE Intelligent Hub dari Google Play.
- Buka aplikasi dan masukkan URL pendaftaran atau pindai kode QR.
- Masuk dan ikuti petunjuk untuk mendaftar sebagai perangkat yang dikelola MDM.

iOS/macOS

- Pada perangkat, buka Safari dan arahkan ke URL pendaftaran (https://<Workspace ONEUEMHostname >/daftar, misalnya).
- Masuk dengan kredensi pengguna.
- Unduh dan instal aplikasi Workspace ONE Intelligent Hub dari App Store.
- Ikuti petunjuk untuk menginstal profil MDM di Settings > General > VPN & Device Management > Profile > Install.

Windows

- Unduh Workspace ONE Intelligent Hub dari server Workspace ONE atau Microsoft Store.

- Mendaftar melalui Hub menggunakan URL pendaftaran dan kredensialnya.

Tetapkan perangkat yang terdaftar ke Grup Cerdas Perangkat POC di Perangkat > Tampilan Daftar > Tindakan Lainnya > Tetapkan ke Grup Cerdas.

Untuk informasi selengkapnya, lihat Pendaftaran Perangkat Otomatis di dokumentasi Omnissa.

Verifikasi pendaftaran

1. Di Omnissa Workspace ONE UEM Console, buka Devices dan kemudian List View.
2. Konfirmasikan bahwa perangkat terdaftar Anda muncul dengan status yang disetel ke Terdaftar.
3. Verifikasi perangkat berada dalam grup pintar perangkat POC di tab Grup pada Detail Perangkat.

Langkah 4: Menerbitkan sertifikat

Pemicu mengeluarkan sertifikat

1. Di Tampilan Daftar Perangkat, pilih perangkat yang terdaftar.
2. Pilih pada tombol Query untuk meminta check-in.
3. Device-Cert-ProfileHarus mengeluarkan sertifikat melalui. AWS Private CA

Verifikasi instalasi sertifikat

Android

Pilih Setelan, lalu Keamanan, lalu Kredensial Tepercaya, lalu Pengguna untuk memverifikasi sertifikat.

iOS

Buka Pengaturan, lalu pilih Umum, lalu VPN & Manajemen Perangkat, dan kemudian Profil Konfigurasi. Verifikasi bahwa sertifikat dari AWS-Private-CA ada.

macOS

Buka Keychain Access dan kemudian System Keychain dan verifikasi sertifikat.

Windows

Buka certmgr.msc, lalu Pribadi, dan kemudian Sertifikat untuk memverifikasi sertifikat.

Pemecahan masalah

Kesalahan SCEP (“22013 - Server SCEP mengembalikan respons yang tidak valid” misalnya)

- Verifikasi URL SCEP dan kata sandi tantangan statis di Workspace ONE match. AWS Private CA
- <SCEP_URL>Uji konektivitas titik akhir SCEP: curl.
- Periksa AWS CloudTrail log untuk AWS Private CA kesalahan (`IssueCertificate`kegagalan, misalnya).

APNs masalah (iOS/macOS)

- Pastikan APNs sertifikat valid dan ditetapkan ke Grup Organisasi yang benar.
- Uji APNs konektivitas: telnet gateway.push.apple.com 2195.

Kegagalan instalasi profil

- Konfirmasikan perangkat berada di Grup Cerdas yang benar (Perangkat, lalu Tampilan Daftar, dan kemudian Grup).
- Memaksa sinkronisasi profil: Tindakan Lainnya, lalu Kirim, dan kemudian Daftar Profil.

Beberapa catatan

- Android: Gunakan log Logcat atau Workspace ONE.
- iOS/macOS: log show --predicate 'process == "mdmclient"' --last 1h (via Xcode/ AppleKonfigurator).
- Windows: Event Viewer, lalu Aplikasi dan Layanan Log dan kemudian Microsoft-Windows -. DeviceManagement
- Workspace ONE UEM: Pantau, lalu Laporan & Analisis, lalu Acara, dan kemudian Acara Perangkat.

Untuk detail Konektor untuk pemantauan SCEP AWS, lihat Konektor Monitor untuk SCEP.

Pertimbangan keamanan

- Simpan SCEP URLs dan rahasia dengan aman. Untuk informasi selengkapnya, lihat AWS Secrets Manager layanannya.
- Batasi kriteria grup pintar hanya untuk menargetkan perangkat.
- Perbarui sertifikat Pemberitahuan Push Apple (APNs) secara teratur (berlaku selama 1 tahun).
- Tetapkan periode validitas sertifikat singkat untuk bukti proyek konsep untuk meminimalkan risiko.

- Untuk perangkat pribadi, pastikan pembersihan menghapus semua profil dan sertifikat.

Untuk informasi tentang cara mengkonfigurasi integrasi Omnissa Workspace ONE UEM dan CA menggunakan konektor SCEP, lihat dokumentasi SCEP di Omnissa Workspace ONE.

Konektor Monitor untuk SCEP

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Konektor untuk SCEP dan AWS solusi Anda yang lain. AWS menyediakan alat pemantauan berikut untuk menonton Konektor untuk SCEP, melaporkan ketika ada sesuatu yang salah, dan mengambil tindakan otomatis bila perlu:

- AWS CloudTrail menangkap panggilan API dan peristiwa terkait yang dibuat oleh atau atas nama Akun AWS Anda dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang dipanggil AWS APIs, alamat IP sumber dari mana panggilan dilakukan, dan kapan panggilan terjadi.

  Jika Anda memantau peristiwa CloudTrail data, log berisi daftar semua permintaan terbaru dari perangkat klien. Peristiwa data datang dengan mengidentifikasi informasi perangkat klien seperti alamat IP, jenis operasi yang dilakukan, dan kode kesalahan dan pesan terperinci jika operasi menghasilkan `failed` status. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CloudTrail.
- Amazon EventBridge adalah layanan bus acara tanpa server yang memudahkan untuk menghubungkan aplikasi Anda dengan data dari berbagai sumber. EventBridge mengirimkan aliran data real-time dari aplikasi Anda sendiri, aplikasi Software-as-a-Service (SaaS), AWS dan layanan serta rute data tersebut ke target seperti Lambda dan Log. CloudWatch Hal ini memungkinkan Anda memantau kejadian yang terjadi dalam layanan, dan membangun arsitektur yang didorong kejadian. Untuk informasi selengkapnya, lihat Panduan EventBridge Pengguna Amazon.

Topik

- Konektor Otomatis untuk menggunakan SCEP EventBridge
- Konektor Log untuk panggilan API SCEP menggunakan AWS CloudTrail

Konektor Otomatis untuk menggunakan SCEP EventBridge

Anda dapat menggunakan Amazon EventBridge 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 EventBridge 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. EventBridge diterbitkan setidaknya sekali. Untuk informasi selengkapnya, lihat Membuat aturan yang bereaksi terhadap peristiwa di EventBridge.

CloudWatch Acara diubah menjadi tindakan menggunakan EventBridge. Dengan EventBridge, Anda dapat menggunakan peristiwa untuk memicu target. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge?

Konektor untuk jenis acara SCEP

Penerbitan Sertifikat Berhasil

Konektor untuk SCEP mengirimkan `Certificate Issuance Succeeded` acara ke EventBridge saat kami mengeluarkan sertifikat sebagai tanggapan `PkiOperationPost` atas permintaan.

Berikut ini adalah contoh data untuk acara tersebut.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Issuance Succeeded",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "success",
       "requestType": "PkiOperationPost",
       "certificateArn": "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/
certificate/certificate_ID"
```

```
    }
}
```

Penerbitan Sertifikat Gagal

Konektor untuk SCEP mengirimkan `Certificate Issuance Failed` acara EventBridge ketika kami tidak dapat mengeluarkan sertifikat sebagai tanggapan `PkiOperationPost` atas permintaan.

Berikut ini adalah contoh data untuk acara tersebut.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Issuance Failed",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "failure",
       "requestType": "PkiOperationPost",
       "reason": "The certificate authority is not active."
   }
}
```

Pengambilan Sertifikat Otoritas Sertifikat Berhasil

Konektor untuk SCEP mengirimkan `Certificate Authority Certificate Retrieval Succeeded` acara ke EventBridge saat kami menerima `GetCACert` permintaan dan berhasil mengambil sertifikat CA pribadi konektor.

Berikut ini adalah contoh data untuk acara tersebut.

```
{
   "version": "0",
```

```
   "id": "event_ID",
   "detail-type": "Certificate Authority Certificate Retrieval Succeeded",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "success",
       "requestType": "GetCACert"
   }
}
```

Pengambilan Sertifikat Otoritas Sertifikat Gagal

Konektor untuk SCEP mengirimkan `Certificate Authority Certificate Retrieval Failed` acara ke EventBridge saat kami menerima `GetCACert` permintaan dan tidak dapat mengambil sertifikat CA pribadi konektor. Acara tersebut mencakup alasan kegagalan.

Berikut ini adalah contoh data untuk acara tersebut.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Authority Certificate Retrieval Failed",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "failure",
       "requestType": "GetCACert",
```

```
       "reason": "The certificate authority certificate validity must be at least one
 year from today."
    }
}
```

Pengambilan Sertifikat Otoritas Sertifikat Berhasil

Konektor untuk SCEP mengirimkan `Certificate Authority Certificate Retrieval Succeeded` acara ke EventBridge saat kami menerima `GetCACert` permintaan dan berhasil mengambil sertifikat CA pribadi konektor.

Berikut ini adalah contoh data untuk acara tersebut.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Certificate Authority Certificate Retrieval Succeeded",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
       "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
   ],
   "detail": {
       "result": "success",
       "requestType": "GetCACert"
   }
}
```

Pengambilan Kemampuan Otoritas Sertifikat Berhasil

Konektor untuk SCEP mengirimkan `Certificate Authority Capabilities Retrieval Succeeded` acara ke EventBridge saat kami menerima `GetCACaps` permintaan SCEP dan berhasil mengambil kemampuan CA.

Berikut ini adalah contoh data untuk acara tersebut.

Pengambilan Kemampuan Otoritas Sertifikat Gagal

Konektor untuk SCEP mengirimkan `Certificate Authority Capabilities Retrieval Failed` acara ke EventBridge saat kami menerima `GetCACaps` permintaan SCEP dan tidak dapat mengambil kemampuan CA. Kami memasukkan alasan kegagalan dalam acara tersebut.

Berikut ini adalah contoh data untuk acara tersebut.

```
{
 "resources":
     [
     "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
     "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector11223344-1234-1122-2233-112233445566"
     ],
 "detailType":"Certificate Authority Capabilities Retrieval Failed",
 "detail": {
     "result":"failure",
     "requestType":"GetCACaps",
     "reason":"The request was denied due to request throttling."
 },
 "source":"aws.pca-connector-scep","accountId":"111122223333"
 }
```

Operasi Tidak Didukung Dipanggil

Operasi Tidak Didukung Dipanggil

Konektor untuk SCEP mengirimkan `Unsupported Operation Invoked` peristiwa ke EventBridge jika operasi yang dikirim ke titik akhir konektor tidak didukung atau tidak diketahui.

```
{
   "version": "0",
   "id": "event_ID",
   "detail-type": "Unsupported Operation Invoked",
   "source": "aws.pca-connector-scep",
   "account": "account",
   "time": "2024-09-12T19:14:56Z",
   "region": "region",
   "resources":[
       "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
```

```
        "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/11223344-1234-1122-2233-112233445566"
    ],
    "detail": {}
}
```

Buat EventBridge aturan

Di EventBridge, Anda dapat membuat aturan yang merespons peristiwa yang direkam oleh CloudTrail. Untuk membuat aturan yang mencakup semua peristiwa yang dicatat oleh Connector untuk SCEP, setel sumbernya ke`aws.pca-connector-scep`. Untuk informasi selengkapnya tentang aturan, lihat Membuat aturan di Amazon EventBridge.

Konektor Log untuk panggilan API SCEP menggunakan AWS CloudTrail

Connector for Simple Certificate Enrollment Protocol (SCEP) terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, klien, atau layanan. AWS CloudTrail menangkap semua panggilan API untuk Connector untuk SCEP sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari Konektor untuk konsol SCEP dan panggilan kode ke Konektor untuk operasi API SCEP. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk peristiwa untuk Connector for SCEP. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Konektor untuk SCEP, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Konektor untuk informasi SCEP di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas terjadi di Konektor untuk SCEP, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di situs Anda Akun AWS. Untuk informasi selengkapnya, lihat Melihat peristiwa dengan Riwayat CloudTrail acara.

Untuk catatan acara yang sedang berlangsung di Anda Akun AWS, termasuk acara untuk Connector for SCEP, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua Wilayah

AWS. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran umum untuk membuat jejak
- CloudTrail layanan dan integrasi yang didukung
- Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail
- Menerima file CloudTrail log dari beberapa wilayah dan Menerima file CloudTrail log dari beberapa akun

Semua Konektor untuk tindakan SCEP dicatat oleh CloudTrail dan didokumentasikan dalam referensi Connector for SCEP API. Misalnya, panggilan ke`CreateConnector`, `GetConnector` dan `CreateChallenge` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan itu dibuat dengan kredenal pengguna root atau AWS Identity and Access Management (IAM).
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan itu dibuat oleh AWS layanan lain.
- Apakah permintaan dibuat oleh perangkat klien SCEP.

Untuk informasi selengkapnya, lihat Elemen userIdentity CloudTrail .

Konektor untuk acara manajemen SCEP

Konektor untuk SCEP terintegrasi dengan CloudTrail merekam tindakan API yang dibuat oleh pengguna, peran, atau AWS layanan di Connector for SCEP. Anda dapat menggunakan CloudTrail untuk memantau Connector untuk permintaan SCEP API secara real time dan menyimpan log di Amazon Simple Storage Service, Amazon CloudWatch Logs, dan Amazon CloudWatch Events. Konektor untuk SCEP mendukung pencatatan tindakan berikut sebagai peristiwa dalam file CloudTrail log:

- CreateChallenge

- CreateConnector
- GetConnector
- GetChallengeMetadata
- GetChallengePassword
- DeleteConnector
- DeleteChallenge

Konektor untuk peristiwa data SCEP di CloudTrail

Peristiwa data memberikan informasi tentang operasi sumber daya yang dilakukan pada atau di sumber daya misalnya, ketika klien Anda mengirim `GetCACaps` pesan SCEP ke titik akhir konektor. Ini juga dikenal sebagai operasi bidang data. Peristiwa data sering kali merupakan aktivitas bervolume tinggi. Secara default, CloudTrail tidak mencatat peristiwa data apa pun, dan riwayat CloudTrail Peristiwa tidak merekamnya.

Biaya tambahan berlaku untuk peristiwa data. Untuk informasi selengkapnya tentang CloudTrail harga, lihat AWS CloudTrail Harga.

Anda dapat mencatat peristiwa data untuk jenis `AWS::PCAConnectorSCEP::Connector` sumber daya menggunakan CloudTrail konsol AWS CLI, atau operasi CloudTrail API. Untuk informasi selengkapnya tentang cara mencatat peristiwa data, lihat Mencatat peristiwa data dengan Konsol Manajemen AWS dan Logging peristiwa data dengan AWS Command Line Interface di Panduan AWS CloudTrail Pengguna.

Tabel berikut mencantumkan Konektor untuk jenis sumber daya SCEP yang dapat Anda log peristiwa data. Kolom tipe peristiwa data (konsol) menunjukkan nilai yang akan dipilih dari daftar tipe peristiwa Data di CloudTrail konsol. Kolom nilai resources.type menunjukkan **`resources.type`** nilai, yang akan Anda tentukan saat mengonfigurasi penyeleksi acara lanjutan menggunakan or. AWS CLI CloudTrail APIs CloudTrailKolom Data yang APIs dicatat ke menampilkan panggilan API yang dicatat CloudTrail untuk jenis sumber daya.

Jenis peristiwa data (konsol)	nilai resources.type	Data APIs masuk CloudTrail
Konektor	`AWS::PCAConnectorSCEP::Connector`	• `PKIOperationGet` - Dihasilkan jika permintaan HTTP GET SCEP

Jenis peristiwa data (konsol)	nilai resources.type	Data APIs masuk CloudTrail
		yang berisi PKCSReq pesan dibuat ke titik akhir dataplane konektor, dan pengoperasian pesan tersebut diatur ke. PKIOperation • PKIOperationPost - Dihasilkan jika permintaan HTTP POST SCEP yang berisi PKCSReq pesan dibuat ke titik akhir dataplane konektor, dan pengoperasian pesan tersebut diatur ke. PKIOperation • GetCACaps - Dihasilkan jika permintaan SCEP yang berisi GetCACaps pesan dibuat ke titik akhir dataplane konektor. • GetCACert - Dihasilkan jika permintaan SCEP yang berisi GetCACert pesan dibuat ke titik akhir dataplane konektor.

Anda dapat mengonfigurasi pemilih acara lanjutan untuk memfilter pada `eventName`,`readOnly`, dan `resources.ARN` bidang untuk mencatat hanya peristiwa yang penting bagi Anda. Contoh berikut adalah tampilan JSON dari konfigurasi peristiwa data yang mencatat peristiwa untuk fungsi tertentu saja. Untuk informasi selengkapnya tentang bidang ini, lihat AdvancedFieldSelectordi Referensi AWS CloudTrail API.

```
[
  {
```

```
    "name": "connector-scep-events",
    "fieldSelectors": [
      {
        "field": "eventCategory",
        "equals": [
          "Data"
        ]
      },
      {
        "field": "resources.type",
        "equals": [
          "AWS::PCAConnectorSCEP::Connector"
        ]
      },
      {
        "field": "resources.ARN",
        "equals": [
          "arn:aws:pca-connector-scep:US West (N.
 California):111122223333:connector/11223344-1122-2233-3344-cae95a00d2a7"
        ]
      }
    ]
  }
]
```

Contoh entri

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh 1: Acara manajemen, **`CreateConnector`**

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateConnector` tindakan.

```
{
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
          "principalId": "AABB1122CCDD4455HHJJ1:11cc33nn2a97724dc48a89071111111111",
```

```
          "arn": "arn:aws:sts::111122223333:assumed-role/Admin",
          "accountId": "111122223333",
          "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "AABB1122CCDD4455HHJJ1",
                  "arn": "arn:aws:iam::111122223333:role/Admin",
                  "accountId": "111122223333",
                  "userName": "my-user-name"
              },
              "attributes": {
                  "creationDate": "2024-08-16T17:46:41Z",
                  "mfaAuthenticated": "false"
              }
          }
        },
        "eventTime": "2024-08-16T17:48:07Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "CreateConnector",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "requestParameters": {
          "ClientToken": "11223344-2222-3333-4444-666555444555",
          "CertificateAuthorityArn": "arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
        },
        "responseElements": {
          "ConnectorArn": "arn:aws:pca-connector-scep:us-east-1:111122223333:connector/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
        },
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
        }
```


Contoh 2: Acara manajemen, **`CreateChallenge`**

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateChallenge` tindakan.

```
{
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
          "principalId": "AABB1122CCDD4455HHJJ1:11cc33nn2a97724dc48a89071111111111",
          "arn": "arn:aws:sts::111122223333:assumed-role/Admin",
          "accountId": "111122223333",
          "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "AABB1122CCDD4455HHJJ1",
                  "arn": "arn:aws:iam::111122223333:role/Admin",
                  "accountId": "111122223333",
                  "userName": "user-name"
              },
              "attributes": {
                  "creationDate": "2024-08-16T17:46:41Z",
                  "mfaAuthenticated": "false"
              }
          }
        },
        "eventTime": "2024-08-16T17:47:52Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "CreateChallenge",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "requestParameters": {
          "ConnectorArn": "arn:aws:pca-connector-scep:us-east-1:111122223333:connector/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
          "ClientToken": "11223344-2222-3333-4444-666555444555"
        },
        "responseElements": {
          "Challenge": {
              "Arn": "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/9cac40bc-acba-412e-9a24-f255ef2fe79a/a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222",
              "ConnectorArn": "arn:aws:pca-connector-scep:us-
east-1:111122223333:connector/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
              "CreatedAt": 1723830472.942,
              "Password": "***",
              "UpdatedAt": 1723830472.942
```

```
          }
        },
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
        }
```

Contoh 3: Acara manajemen, **`GetChallengePassword`**

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `GetChallengePassword` tindakan.

```
{
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
          "principalId": "AABB1122CCDD4455HHJJ1:11cc33nn2a97724dc48a89071111111111",
          "arn": "arn:aws:sts::111122223333:assumed-role/Admin",
          "accountId": "111122223333",
          "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "AABB1122CCDD4455HHJJ1",
                  "arn": "arn:aws:iam::111122223333:role/Admin",
                  "accountId": "905418114790",
                  "userName": "111122223333"
              },
              "attributes": {
                  "creationDate": "2024-08-16T17:55:01Z",
                  "mfaAuthenticated": "false"
              }
          },
          "invokedBy": "signin.amazonaws.com"
        },
        "eventTime": "2024-08-16T17:55:54Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "GetChallengePassword",
        "awsRegion": "us-east-1",
```

```
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "requestParameters": {
          "ChallengeArn": "arn:aws:pca-connector-scep:us-east-1:111122223333:challenge/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
        },
        "responseElements": null,
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
        }
```

Contoh 4: Peristiwa data, **`PkiOperationPost`**

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `PkiOperationPost` panggilan gagal. Log menyertakan kode kesalahan dan pesan kesalahan dengan penjelasan kegagalan.

```
{
        "eventVersion": "1.10",
        "userIdentity": {
          "type": "FederatedUser",
          "principalId": "111122223333",
          "accountId": "111122223333"
        },
        "eventTime": "2024-08-16T17:40:09Z",
        "eventSource": "pca-connector-scep.amazonaws.com",
        "eventName": "PkiOperationPost",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.0.0.0",
        "userAgent": "Python/3.11.8 Darwin/22.6.0 exe/x86_64",
        "errorCode": "BadRequestException",
        "errorMessage": "The certificate authority is not in a valid state for issuing
 certificates (Service: AcmPca, Status Code: 400, Request ID: a1b2c3d4-5678-90ab-cdef-
EXAMPLE55555)",
        "requestParameters": null,
        "responseElements": null,
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
```

```
        "resources": [
          {
              "accountId": "111122223333",
              "type": "AWS::PCAConnectorSCEP::Connector",
              "ARN": "arn:aws:pca-connector-scep:us-east-1:111122223333:connector/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
          }
        ],
        "eventType": "AwsApiCall",
        "managementEvent": false,
        "recipientAccountId": "905418114790",
        "eventCategory": "Data",
        "tlsDetails": {
          "clientProvidedHostHeader": "111122223333-a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333.enroll.pca-connector-scep.us-east-1.api.aws"
        }
        }
```

Memecahkan Masalah AWS Private Certificate Authority Konektor untuk masalah SCEP

Anda mungkin perlu memecahkan masalah yang terkait dengan Konektor Anda untuk implementasi SCEP. Bab ini memberikan informasi rinci tentang HTTP dan kesalahan klien yang dikirim oleh layanan.

Topik

- Memecahkan masalah kesalahan HTTP dari Konektor untuk SCEP
- Memecahkan masalah Konektor untuk kesalahan klien SCEP

Memecahkan masalah kesalahan HTTP dari Konektor untuk SCEP

Saat klien Anda memicu aksi API Connector for SCEP dataplane dan menghasilkan kesalahan, Connector for SCEP mengirimkan kode respons HTTP ke klien yang meminta dengan informasi tentang kesalahan tersebut.

Selain tanggapan layanan yang diberikan langsung ke klien Anda, Anda dapat menggunakan alat pemantauan yang dijelaskan di Konektor Monitor untuk SCEP bagian untuk melihat dan men-debug kesalahan yang mengakibatkan kesalahan HTTP.

Berikut ini adalah pesan kesalahan yang dikembalikan oleh layanan ke klien SCEP, penyebab potensial, dan langkah-langkah yang dapat Anda ambil untuk menyelesaikan masalah.

HTTP 400 Permintaan Buruk

Kode respons HTTP 400 berarti bahwa Konektor untuk SCEP tidak dapat memproses permintaan karena kesalahan klien yang jelas, seperti data yang hilang atau tidak valid dalam permintaan. Jika kesalahan dihasilkan dari kesalahan spesifik protokol SCEP, Konektor untuk SCEP menyertakan respons SCEP sebagai biner dalam pesan. Konektor untuk SCEP APIs dapat mengembalikan 400 tanggapan untuk salah satu alasan berikut.

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
LimitExceededException	Batas penerbitan otoritas sertifikat terlampaui.	Private Certificate Authority (CA) yang terkait dengan konektor telah melampaui kuota untuk jumlah sertifikat yang dapat diterbitkan.	Konektor SCEP hanya dapat dihubungkan ke satu CA pribadi selama masa pakainya. Jika Anda telah kehabisan batas CA pribadi Anda, buat konektor baru atau minta peningkatan kuota. Untuk informasi lebih lanjut tentang kuota CA pribadi, lihat AWS Private Certificate Authority kuota.	Tidak
ValidationException	Permintaan harus berisi base64.	Konektor untuk SCEP tidak dapat memproses permintaan HTTP GET karena	Jika memungkinkan, konfigurasikan klien Anda untuk menggunakan pesan	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
		badannya tidak valid Base64.	HTTP POST alih-alih pesan HTTP GET. Jika Anda harus menggunakan HTTP GET, pesan harus menggunakan format Base64. Jika klien Anda tidak sesuai dengan persyaratan ini, hubungi AWS Dukunganuntuk bantuan.	
ValidationException	Otoritas sertifikat tidak aktif.	CA pribadi yang terkait dengan konektor tidak aktif.	Aktifkan kembali CA pribadi. Untuk informasi, lihat Perbarui CA pribadi di AWS Private Certificate Authority.	Tidak
ValidationException	Validitas sertifikat otoritas sertifikat harus setidaknya satu tahun dari hari ini.	CA pribadi yang terkait dengan konektor tujuan umum harus memiliki masa berlaku satu tahun mulai hari ini.	Menerbitkan kembali sertifikat dengan masa berlaku lebih dari satu tahun mulai hari ini. Untuk informasi tentang mengelola sertifikat, lihatMengelola siklus hidup CA pribadi.	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
ValidationException	Sertifikat yang termasuk dalam permintaan sudah kedaluwarsa.	Sertifikat transien yang dihasilkan oleh perangkat klien pada setiap transaksi telah kedaluwarsa pada penerimaan oleh layanan.	Kemungkinan besar perangkat klien Anda tidak memiliki pengaturan waktu yang dikonfigurasi dengan benar, dan mereka membuat sertifikat dengan tanggal di belakang waktu nyata. Jika Anda tidak dapat mengatasi masalah ini, hubungi AWS Dukunganuntuk bantuan.	Tidak
ValidationException	Permintaan berisi Sintaks Pesan Kriptografi yang tidak valid.	Layanan tidak dapat memecahkan kode pesan permintaan SCEP.	Periksa apakah pesan SCEP Anda sesuai dengan Sintaks Pesan Kriptografi yang ditentukan dalam SCEP RFC 8894. Jika Anda tidak dapat mengatasi masalah ini, hubungi AWS Dukunganuntuk bantuan.	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
ValidationException	Konektor tidak aktif.	Status konektor tidak aktif.	Anda dapat menemukan status konektor di konsol atau di bidang Status di API. Status konektor dapat dibuat, aktif, dihapus, atau gagal. Jika status dibuat, coba permintaan Anda nanti. Jika status gagal, lihat alasan status untuk memecahkan masalah, lalu buat konektor baru.	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
ValidationException	Harus ada sertifikat yang valid yang termasuk dalam permintaan.	Sertifikat transien yang disertakan dalam pesan permintaan dari klien hilang atau tidak valid.	Klien yang kompatibel dengan SCEP harus memberikan sertifikat yang ditandatangani sendiri untuk mengotentikasi diri mereka sendiri. Jika klien Anda tidak dapat memberikan sertifikat yang ditandatangani sendiri yang diperlukan, hubungi AWS Dukunganuntuk bantuan.	Tidak
ValidationException	URI permintaan tidak valid.	Konektor untuk SCEP tidak dapat mengurai permintaan karena jalur URI atau kueri permintaan tidak valid.	Administrator harus memverifikasi pengaturan konfigurasi perangkat klien, yang biasanya dikelola melalui sistem Mobile Device Management (MDM). Untuk informasi selengkapnya, lihat Langkah 2: Salin detail konektor ke sistem MDM Anda.	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
ValidationException	Tepat satu header host diperlukan dalam permintaan.	Klien tidak memberikan header Host HTTP yang valid dalam permintaan, yang diperlukan agar permintaan diproses.	Header host HTTP diperlukan untuk membedakan permintaan yang datang ke konektor yang berbeda. Jika klien Anda tidak dapat memberikan header host HTTP yang diperlukan, hubungi AWS Dukunganuntuk bantuan.	Tidak
ValidationException	Permintaan tidak dapat diterjemahkan. Silakan kirim permintaan SCEP yang valid.	Layanan tidak dapat memecahkan kode dan memproses permintaan Cryptographic Message Syntax (CMS) yang dikirim klien Anda.	Jika klien Anda mengalami masalah dengan penerapan SCEP kami, perhatikan ID permintaan (`x-amzn-requestid`) dari respons dan kontak AWS Dukungan.	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
ValidationException	Respons tidak dapat dikodekan dengan nilai yang berasal dari permintaan. Silakan kirim permintaan SCEP yang valid.	Layanan tidak dapat menyandikan respons SCEP.	Masalah ini biasanya terjadi ketika layanan tidak dapat menggunakan sertifikat pemohon yang disediakan untuk menyandikan pesan respons SCEP dengan benar. Ini dapat terjadi, misalnya, jika sertifikat pemohon memiliki kunci Elliptic Curve Digital Signature Algorithm (ECDSA), yang tidak didukung oleh Connector untuk SCEP. Jika Anda mengalami masalah ini, pertama-tama konfigurasikan klien MDM atau SCEP Anda untuk menggunakan RSA. Jika Anda masih tidak dapat menyelesaikan masalah, catat ID permintaan (`x-amzn-requestid`) dari respons dan hubungi	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
			AWS Dukunganuntuk bantuan.	
ValidationException	Algoritma yang tidak didukung: <OID>	Permintaan tersebut ditandatangani atau dienkripsi oleh algoritma kriptografi yang tidak didukung.	Layanan kami tidak mendukung algoritma kriptografi yang sudah ketinggalan zaman dan lemah. Informasi ini dikomunikasikan kepada klien melalui `GetCACaps` permintaan. Namun, beberapa klien mungkin tidak menggunakan metode ini untuk memeriksa algoritma yang didukung. Jika klien Anda tampaknya tidak kompatibel dengan algoritma kriptografi yang didukung oleh layanan kami, hubungi AWS Dukunganuntuk bantuan.	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
ValidationException	PkiOperation MessageType yang tidak didukung.	Pesan permintaan berisi jenis `PkiOperation` pesan yang tidak valid dan tidak dapat diproses oleh layanan.	Layanan kami hanya mendukung sebagian dari jenis pesan protokol SCEP yang ditentukan dalam RFC 8894. Secara khusus, kami mengenali dan memproses jenis pesan berikut: CertRep,, PKCSReq GetCert, GetCrl, dan. CertPoll Kami mengkomunikasikan jenis pesan yang didukung kepada klien melalui CACaps metode Get. Sayangnya, beberapa klien mungkin tidak menggunakan metode ini dan mungkin tidak sesuai dengan kemampuan layanan kami. Jika klien Anda tampaknya tidak kompatibel dengan jenis pesan SCEP yang didukung	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
			oleh layanan kami, hubungi. AWS Dukungan	
BadRequestException	Kata sandi tantangan tidak valid.	Kata sandi tantangan yang diberikan oleh klien tidak valid untuk titik akhir layanan yang dihubungi dan konektor terkaitnya. Kata sandi tantangan adalah ukuran keamanan yang diperlukan yang ditentukan dalam protokol SCEP untuk memastikan hanya klien yang berwenang yang dapat mengakses layanan.	Pastikan klien Anda memberikan kata sandi tantangan yang benar dalam permintaannya. Anda dapat menemukan detail konektor di konsol atau melalui GetChallengePasswordAPI. Untuk informasi selengkapnya, lihat Langkah 2: Salin detail konektor ke sistem MDM Anda.	Ya

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
BadRequestException	Tepat satu kata sandi tantangan diperlukan dalam permintaan penandatanganan sertifikat.	Klien memberikan nol atau beberapa kata sandi tantangan dalam permintaannya.	Pastikan klien Anda memberikan satu kata sandi tantangan dalam permintaannya. Anda dapat menemukan kata sandi tantangan di detail konektor di konsol atau melalui GetChallengePasswordAPI. Untuk informasi selengkapnya, lihat Langkah 2: Salin detail konektor ke sistem MDM Anda.	Ya
BadRequestException	Konektor tidak memiliki akses ke Azure.	Konektor untuk Microsoft Intune mengotorisasi permintaan klien melalui Microsoft Intune. Ini mengharuskan Anda memberikan izin kepada Connector for SCEP untuk mengakses sumber daya Azure Anda.	Konfigurasikan izin yang dirinci diLangkah 1: Berikan AWS Private CA izin untuk menggunakan Aplikasi Microsoft Entra ID Anda.	Ya

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
BadRequestException	<action>Aplikasi Azure tidak memiliki akses untuk melakukan.	Konektor untuk Microsoft Intune mengotorisasi permintaan klien melalui Microsoft Intune. Ini mengharuskan Anda memberikan izin kepada Connector for SCEP untuk mengakses sumber daya Azure Anda.	Konfigurasikan izin yang dirinci di Langkah 1: Berikan AWS Private CA izin untuk menggunakan Aplikasi Microsoft Entra ID Anda.	Ya
BadRequestException	Aplikasi Azure tidak ditemukan.	Konektor untuk Microsoft Intune mengotorisasi permintaan klien melalui Microsoft Intune. Kesalahan ini menunjukkan bahwa Anda tidak memiliki Pendaftaran Aplikasi di ID Microsoft Entra, atau detail Intune konektor Anda salah dikonfigurasi.	Ikuti panduan dalam Konfigurasikan Microsoft Intune untuk Konektor untuk SCEP topik.	Ya

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
BadRequestException	Validasi permintaan penandatanganan sertifikat intune gagal. Alasan: <reason>	Konektor untuk Microsoft Intune mengotorisasi permintaan klien melalui Microsoft Intune. Pesan kesalahan ini menunjukkan bahwa proses validasi Intune telah gagal, dan kode kesalahan Intune yang sesuai disediakan.	Ikuti panduan dalam Konfigurasikan Microsoft Intune untuk Konektor untuk SCEP topik. Jika masalah Anda berlanjut, hubungi Microsoft Support.	Ya

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
BadRequestException	PkiOperation <message type>MessageType yang tidak didukung:.	Pesan permintaan berisi jenis pesan yang tidak valid dan tidak dapat diproses oleh layanan.	Layanan kami hanya mendukung sebagian dari jenis pesan protokol SCEP yang ditentukan dalam RFC 8894. Secara khusus, kami mengenali dan memproses jenis pesan berikut: CertRep,, PKCSReq GetCert, GetCrl, dan. CertPoll Kami mengkomunikasikan jenis pesan yang didukung kepada klien melalui CACaps metode Get. Sayangnya, beberapa klien mungkin tidak menggunakan metode ini dan mungkin tidak sesuai dengan kemampuan layanan kami. Jika klien Anda tampaknya tidak kompatibel dengan jenis pesan SCEP yang didukung oleh layanan kami,	Ya

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
			hubungi. AWS Dukungan	
BadRequestException	Algoritma kunci atau panjang tidak didukung.	Layanan ini tidak mendukung kunci publik yang disediakan termasuk dalam permintaan penandatanganan sertifikat.	Layanan kami hanya mendukung kunci RSA standar hingga 16.384 bit, dan kunci ECDSA hingga 521 bit. Jika klien Anda memerlukan penggunaan algoritma yang saat ini tidak didukung, silakan hubungi AWS Dukunganuntuk bantuan.	Ya

HTTP 401 Tidak Sah

Kode status respons 401 Tidak Sah menunjukkan bahwa permintaan klien belum selesai karena tidak memiliki kredensi otentikasi yang valid untuk sumber daya yang diminta.

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
AccessDeniedException	Konektor tidak memiliki akses ke otoritas sertifikat.	Konektor untuk SCEP tidak memiliki akses ke CA pribadi terkait konektor.	Bagikan CA pribadi Anda dengan Konektor untuk digunakan AWS	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
			Resource Access Manager SCEP.	
AccountDoesNotExistException	AWS Akun tidak ada.	Konektor untuk sumber daya SCEP tidak ada lagi.	Akun yang memiliki sumber daya target telah dihapus. Jika ini dilakukan secara tidak sengaja, hubungi AWS Dukungandalam periode 90 hari pasca-penutupan.	Tidak

HTTP 404 Tidak Ditemukan

Kode respons HTTP 404 biasanya berarti bahwa sumber daya yang Anda cari tidak dapat ditemukan.

Header respons (x-amzn-ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
ResourceNotFoundException	Otoritas sertifikat tidak ada.	CA pribadi terkait konektor telah dihapus.	Ada masa tenggang di mana Private Certificate Authority (CA) dapat dipulihkan jika telah dihapus secara tidak sengaja. Untuk informasi selengkapnya, lihat	Tidak

Header respons (x-amzn-ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
			Kembalikan CA pribadi.	
ResourceNotFoundException	Konektor dengan titik akhir <URL>tidak ada.	Perangkat klien telah mencoba untuk terhubung ke URL yang bukan milik konektor yang ada.	Pastikan klien Anda menyediakan titik akhir yang benar untuk konektor. Untuk melihat konektor`Endpoint`, panggil GetConnectorAPI atau lihat di halaman detail konektor di konsol.	Tidak

HTTP 409 Konflik

Respons Konflik HTTP 409 memberi sinyal bahwa CA pribadi yang terkait dengan konektor telah berubah sejak permintaan dimulai.

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
ConflictException	Konektor telah berubah sejak permintaan dimulai.	CA pribadi yang terkait dengan konektor telah diperbarui, memicu rotasi sertifikat internal konektor yang digunakan	Coba permintaan Anda lagi dalam beberapa menit. Jika masalah tidak teratasi, hubungi AWS Dukunganuntuk bantuan.	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
		untuk komunikasi dengan perangkat klien melalui SCEP. Rotasi sertifikat ini dapat mengakibatkan masalah sementara selama periode pembaruan, karena sertifikat baru sedang digunakan. Namun, kesalahan ini harus diselesaikan secara otomatis tepat waktu.		

HTTP 429 Terlalu Banyak Permintaan

Konektor untuk SCEP memiliki kuota tingkat akun, per Wilayah. Jika Anda melebihi batas permintaan ke konektor, permintaan Anda akan ditolak dengan kesalahan HTTP 429. Jika Anda perlu menambah kuota, lihat AWS Private Certificate Authority titik akhir dan kuota.

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
ThrottlingException	Permintaan ditolak karena throttling permintaan.	Terlalu banyak permintaan telah dikeluarkan untuk Konektor ini, memicu	Jika Anda melebihi batas permintaan ke konektor, permintaan Anda akan ditolak. Jika Anda perlu	Tidak

Header respons (x-amzn-) ErrorType	Pesan kesalahan (x-amzn-) ErrorMessage	Penyebab galat	Remediasi	Termasuk respon SCEP?
		beberapa permintaan untuk ditolak. Rotasi sertifikat ini dapat mengakibatkan masalah sementara selama periode pembaruan, karena sertifikat baru sedang digunakan. Namun, kesalahan ini harus diselesaikan secara otomatis tepat waktu.	menambah kuota, lihat Konektor untuk titik akhir dan kuota SCEP.	

Memecahkan masalah Konektor untuk kesalahan klien SCEP

Gunakan panduan berikut untuk memecahkan masalah kesalahan klien yang terkait dengan Konektor untuk SCEP.

Contoh pesan	Penyebab galat	Solusi
Kunci ECDSA tidak didukung	Konektor terhubung ke CA pribadi yang menggunakan kunci ECDSA, bukan RSA. Meskipun layanan ini mendukung kunci ECDSA, tidak semua perangkat klien mungkin kompatibel dengan algoritma ini.	Pertimbangkan untuk menggunakan CA pribadi yang dienkripsi RSA alih-alih ECDSA. Jika Anda membuat CA pribadi yang menggunakan RSA, Anda juga harus membuat konektor baru. Konektor hanya dapat diikat ke satu CA pribadi selama masa pakainya.

Contoh pesan	Penyebab galat	Solusi
Enkripsi atau sertifikat penandata nganan tidak ada	Menurut RFC 8894, layanan SCEP mengembalikan sertifikat CA menengah ke klien. Sertifika t ini digunakan oleh klien untuk melakukan operasi enkripsi dan validasi tanda tangan sebagai bagian dari protokol SCEP. Konektor untuk SCEP menggunak an sertifikat yang sama untuk tujuan enkripsi dan validasi tanda tangan, yang merupakan pendekatan umum. Namun, beberapa klien mungkin berharap memiliki dua sertifikat terpisah sebagai gantinya.	Jika Anda tidak dapat menggunakan klien yang kompatibel, hubungi AWS Dukunganuntuk bantuan.

AWS Private CA kuota layanan

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 APIs.

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 Service Quotas 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.

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
Pembaruan dokumentasi	Memperbarui Secure Kubernetes AWS Private Certificate Authority dengan prosedur baru memulai, contoh, dan topik pemantauan dan pemecahan masalah.	Oktober 1, 2025
Dukungan dual-stack	AWS Private Certificate Authority mendukung dual-stack.	Juni 23, 2025
Dukungan domain anak untuk Connector for AD sekarang tersedia secara umum	Anda sekarang dapat mengatur Konektor untuk AD dengan domain anak Anda.	Juni 2, 2025
Kebijakan terkelola baru: `AWSPrivateCAConnectorForKubernetesPolicy`	Kebijakan terkelola baru diperkenalkan untuk digunakan dengan AWS Private CA Connector untuk Kubernetes.	19 Mei 2025
Kebijakan yang diperbarui `AWSPrivateCAPrivilegedUser` dan `AWSPrivateCAUser` dikelola	Diganti `StringLike` dengan `ArnLike` in `AWSPrivateCAUser` dan`AWSPrivateCAPrivilegedUser` . Template ARN yang diperbarui untuk memasukkan	Januari 22, 2025

	kartu `arn:aws:acm-pca:::template` liar ke. `arn:aws:acm-pca:*:*:template`	
Konektor untuk SCEP sekarang tersedia secara umum	Konektor untuk SCEP sekarang tersedia secara umum.	September 16, 2024
Topik pemecahan masalah baru	Menambahkan topik baru yang membantu Anda memecahkan masalah yang terkait dengan memperbarui Connector for Active Directory template.	Juli 31, 2024
Ditambahkan cara memperbarui Konektor untuk template AD	Menambahkan prosedur yang menjelaskan cara memperbarui Konektor untuk template AD, dan cara AWS Private CA menyebarkan pembaruan tersebut.	Juli 31, 2024
Konektor untuk SCEP untuk Omnissa Workspace ONE sekarang tersedia secara umum	Konektor untuk SCEP untuk Omnissa Workspace ONE sekarang tersedia secara umum.	Juli 23, 2024
Menambahkan kendala untuk laporan audit	AWS Private CA tidak mendukung penggunaan Amazon S3 Object Lock dengan bucket yang digunakan untuk laporan audit.	Juli 3, 2024
Sekarang mendukung SM2 untuk Wilayah Tiongkok	AWS Private CA sekarang mendukung algoritma SM2 penandatanganan, hanya untuk Wilayah Tiongkok.	27 Juni 2024

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 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 masalah CAs itu 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 Certifica te 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 CustomAttributeobjek untuk mengonfigurasi kustomisasi CAs dan sertifikat, 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 CRLs	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 privat CAs yang tidak mereka miliki. Untuk informasi selengkapnya, lihat Akses Lintas Akun ke Pribadi CAs.	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 CloudFormation. Untuk informasi selengkapnya, lihat Referensi Jenis Sumber Daya ACMPCA dalam Panduan Pengguna 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 yang baru APIs `TagResource` ,`UntagResource` , dan. `ListTagsForResource` 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	Templat sertifikat baru ditambahkan, termasuk templat untuk penandata nganan 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 selengkap nya, 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 Sertifika t AWS Certificate Manager Pribadi.	24 Juli 2019
Menambahkan dukungan hierarki CA pribadi lengkap	Support untuk membuat dan hosting root CAs menghilan gkan kebutuhan untuk 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 (Californ ia 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 Certi ficate Manager Private Certificate Authority Endpoints and Quotas.	1 Agustus 2018
Kembalikan dihapus CAs	Pemulihan CA pribadi memungkinkan pelanggan untuk memulihkan otoritas sertifikat (CAs) 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.

Ubah	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.