aws

Guida per l'utente

AWS Private Certificate Authority

Version latest

AWS Private Certificate Authority: Guida per l'utente

Table of Contents

Cos'è CA privata AWS?

CA privata AWS consente la creazione di gerarchie CA (Private Certificate Authority), incluse CA root e subordinate, senza i costi di investimento e manutenzione di una CA locale. Le CA private possono emettere certificati X.509 dell'entità finale utili in scenari quali:

- Creazione di canali di comunicazione TLS crittografati
- Autenticazione di utenti, computer, endpoint API e dispositivi IoT
- Codice di firma crittografica
- Implementazione del protocollo OCSP (Online Certificate Status Protocol) per ottenere lo stato di revoca del certificato

Puoi accedere alle operazioni CA privata AWS dalla AWS Management Console, utilizzando l'API CA privata AWS o l'AWS CLI.

Argomenti

Qual è il miglior servizio di certificazione per le mie esigenze?

Esistono due servizi AWS per l'emissione e la distribuzione di certificati X.509. Scegli quello che meglio si adatta alle tue esigenze. Le considerazioni includono se sono necessari certificati pubblici o privati, certificati personalizzati, certificati che si desidera distribuire in altri servizi AWS o gestione e rinnovo automatizzati dei certificati.

1. CA privata AWS: questo servizio è rivolto ai clienti aziendali che costruiscono un'infrastruttura a chiave pubblica (PKI) all'interno del cloud AWS ed è destinato all'uso privato all'interno di un'organizzazione. Con CA privata AWS, puoi creare la tua gerarchia di CA ed emettere certificati

per l'autenticazione di utenti, computer, applicazioni, servizi, server e altri dispositivi, nonché per la firma di codice dei computer. I certificati emessi da una CA privata sono attendibili solo all'interno dell'organizzazione, non su Internet.

Dopo aver creato una CA privata, puoi emettere certificati direttamente (ovvero senza ottenere la convalida da una CA di terze parti) e personalizzarli in base alle esigenze interne dell'organizzazione. Ad esempio, potresti voler:

- Creare certificati con qualsiasi nome di oggetto.
- Creare certificati con qualsiasi data di scadenza.
- Utilizzare qualsiasi algoritmo di chiave privata e lunghezza di chiave supportati.
- Utilizzare qualsiasi algoritmo di firma supportato.
- Controllare l'emissione di certificati utilizzando i modelli.

Sei nel posto giusto per questo servizio. Per iniziare, accedi alla console https://console.aws.amazon.com/acm-pca/.

2. AWS Certificate Manager(ACM): questo servizio gestisce i certificati per i clienti aziendali che necessitano di una presenza web sicura e pubblicamente affidabile tramite TLS. Puoi distribuire certificati ACM in AWS Elastic Load Balancing, CloudFront Amazon, Amazon API Gateway e altri servizi integrati. L'applicazione più comune di questo tipo è un sito pubblico sicuro con requisiti di traffico significativi.

   Puoi utilizzare certificati pubblici forniti da ACM o certificati importati su ACM.. Se utilizzi CA privata AWS per creare una CA, ACM può gestire l'emissione di certificati da quella CA privata e automatizzare i rinnovi dei certificati.

   Per ulteriori informazioni, consulta la AWS Certificate ManagerGuida per l'utente.

Regioni

Come la maggior parte delle risorse AWS, le autorità di certificazione (CA) sono risorse regionali. Per utilizzare CA private in più di una regione, è necessario creare le CA in quelle regioni. Non è possibile copiare CA private tra regioni. Visita Regioni ed endpoint AWS in Riferimenti generali di AWS o la Tabella delle regioni AWS per vedere la disponibilità regionale per CA privata AWS.

> **Note**
>
> ACM è attualmente disponibile in alcune regioni che non lo sono. CA privata AWS

Servizi integrati con AWS Private Certificate Authority

Se si utilizza l'opzione AWS Certificate Manager per richiedere un certificato privato, è possibile associare tale certificato a qualsiasi servizio integrato con ACM. Ciò vale sia per i certificati concatenati a una root CA privata AWS sia per i certificati concatenati a una root esterna. Per ulteriori informazioni, consulta Integrated Services nella Guida per l'AWS Certificate Managerutente.

Puoi anche integrare CA private in Amazon Elastic Kubernetes Service per fornire l'emissione di certificati all'interno di un cluster Kubernetes. Per ulteriori informazioni, consulta Proteggere Kubernetes con CA privata AWS.

> **Note**
>
> Amazon Elastic Kubernetes Service non è un servizio integrato ACM.

Se utilizzi l'CA privata AWSAPI o AWS CLI per emettere un certificato o esportare un certificato privato da ACM, puoi installare il certificato ovunque desideri.

Algoritmi crittografici supportati

CA privata AWSsupporta i seguenti algoritmi crittografici per la generazione di chiavi private e la firma dei certificati.

Algoritmo supportato

Algoritmi a chiave privata	Algoritmi di firma
RSA_2048	SHA256 CON ECDSA
RSA_4096	SHA384 CON ECDSA
EC_Prime256v1	SHA512 CON ECDSA
EC_SECP384R1	SHA256WITHRSA

Algoritmi a chiave privata	Algoritmi di firma
	SHA384WITHRSA SHA512WITHRSA

Questo elenco si applica solo ai certificati emessi direttamente CA privata AWS tramite la console, l'API o la riga di comando. Quando AWS Certificate Manager emette certificati utilizzando un CA daCA privata AWS, supporta alcuni ma non tutti questi algoritmi. Per ulteriori informazioni, consulta Richiedere un certificato privato nella Guida per l'AWS Certificate Managerutente.

> **Note**
>
> In tutti i casi, la famiglia di algoritmi di firma specificata (RSA o ECDSA) deve corrispondere alla famiglia di algoritmi della chiave privata della CA.

Quote

CA privata AWSassegna quote al numero consentito di certificati e alle autorità di certificazione. Anche le tariffe di richiesta per le azioni API sono soggette a quote. CA privata AWSle quote sono specifiche per un AWS account e una regione.

CA privata AWSlimita le richieste API a velocità diverse a seconda del funzionamento dell'API. Limitare significa che CA privata AWS rifiuta una richiesta valida perché la richiesta supera il limite dell'operazione stabilito di richieste al secondo. Quando una richiesta viene limitata, CA privata AWS restituisce un errore. ThrottlingException CA privata AWSnon garantisce una frequenza minima di richieste per le API.

Per vedere quali quote possono essere modificate, consulta la tabella delle CA privata AWS quote in. Riferimenti generali di AWS

Puoi visualizzare le quote attuali e richiedere aumenti delle quote utilizzando AWS Service Quotas.

Per visualizzare un up-to-date elenco delle tue quote CA privata AWS

1. Accedi al tuo AWS account.
2. Apri la console Service Quotas all'indirizzo https://console.aws.amazon.com/servicequotas/

3. Nell'elenco Servizi, selezionare AWSCertificate Manager Private Certificate Authority (ACM PCA). Ogni quota nell'elenco delle quote di servizio mostra il valore della quota attualmente applicata, il valore di quota predefinito e se la quota è regolabile o meno. Scegli il nome di una quota per ulteriori informazioni al riguardo.

Richiesta di un aumento delle quote

1. Nell'elenco delle quote di servizio, scegli il pulsante di opzione per una quota regolabile.
2. Scegli il pulsante Richiedi aumento della quota.
3. Compila e invia il modulo di richiesta di aumento della quota.

CA privata AWS è integrato con AWS Certificate Manager. È possibile utilizzare la console ACM o AWS CLI l'API ACM per richiedere certificati privati da una CA privata esistente. Questi certificati PKI privati, gestiti da ACM, sono soggetti sia alle quote PCA sia alle quote che ACM impone ai certificati pubblici e importati. Per ulteriori informazioni sui requisiti ACM, consulta Request a Private Certificate and Quotas nella Guida per l'utente. AWS Certificate Manager

Conformità RFC

CA privata AWS non applica determinati vincoli definiti nella RFC 5280. Anche la situazione inversa è vera: vengono applicati alcuni vincoli aggiuntivi appropriati per una CA privata.

Applicato

- Non dopo la data. Conformemente alla RFC 5280, CA privata AWS impedisce il rilascio di certificati recanti una data `Not After` successiva alla data `Not After` del certificato della CA emittente.
- Vincoli di base. CA privata AWS applica i vincoli di base e la lunghezza del percorso nei certificati emessi da una CA importati.

  I vincoli di base indicano se la risorsa identificata dal certificato è una CA e può emettere certificati. I certificati emessi da una CA importati in CA privata AWS devono includere l'estensione dei vincoli di base e l'estensione deve essere contrassegnata con `critical`. Oltre al flag `critical`, `CA=true` deve essere impostato. CA privata AWS applica i vincoli di base fallendo con un'eccezione di convalida per i seguenti motivi:

  - L'estensione non è inclusa nel certificato emesso da una CA.
  - L'estensione non è contrassegnata `critical`.

La lunghezza del percorso (pathLenConstraint) determina quante CA subordinate possono esistere a valle del certificato CA importato. CA privata AWSimpone la lunghezza del percorso fallendo con un'eccezione di convalida per i seguenti motivi:

- L'importazione di un certificato emesso da una CA violerebbe il vincolo di lunghezza del percorso nel certificato emesso da una CA o in qualsiasi certificato emesso da una CA nella catena.
- L'emissione di un certificato violerebbe un vincolo di lunghezza del percorso.

Non applicato

- Vincoli delle policy. Questi vincoli limitano la capacità di una CA di emettere certificati emessi da una CA subordinata.
- Identificatore chiave soggetto (SKI) e identificatore chiave dell'autorità (AKI). La RFC richiede un certificato emesso da una CA per contenere l'estensione SKI. I certificati rilasciati dalla CA devono contenere un'estensione AKI corrispondente allo SKI del certificato emesso da una CA. AWS non applica questi requisiti. Se il certificato emesso da una CA non contiene uno SKI, l'AKI del certificato emesso da una CA dell'entità finale o subordinata emesso sarà invece l'hash SHA-1 della chiave pubblica emittente.
- SubjectPublicKeyInfoe Subject Alternative Name (SAN). Quando si emette un certificato, CA privata AWS copia le estensioni SubjectPublicKeyInfo e le estensioni SAN dal CSR fornito senza eseguire la convalida.

Prezzi

Al tuo account viene addebitato un prezzo mensile per ogni CA privata a partire dal momento in cui la crei. Verranno addebitati anche i costi per ogni certificato rilasciato. Questo addebito include i certificati esportati da ACM e i certificati creati dall'CA privata AWSAPI o dalla CA privata AWS CLI. Non è previsto alcun addebito per una CA privata dopo che è stata eliminata. Tuttavia, se ripristini una CA privata, ti verrà addebitata per l'intervallo di tempo compreso tra l'eliminazione e il ripristino. I certificati privati per i quali non hai accesso alla chiave privata sono gratuiti. Questi includono certificati utilizzati con servizi integrati come Elastic Load Balancing e API CloudFront Gateway.

Per le informazioni più recenti CA privata AWS sui prezzi, consulta la sezione AWS Private Certificate AuthorityPrezzi. Puoi anche utilizzare il calcolatore dei AWS prezzi per stimare i costi.

Sicurezza in AWS Private Certificate Authority

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di data center e architetture di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra te e te. AWS Il modello di responsabilità condivisa descrive questo aspetto come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. I revisori esterni testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei AWS Programmi di AWS conformità dei Programmi di conformità dei di . Per maggiori informazioni sui programmi di conformità applicabili AWS Private Certificate Authority, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS .
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. Sei anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della tua azienda e le leggi e normative vigenti.

Questa documentazione ti aiuta a capire come applicare il modello di responsabilità condivisa durante l'utilizzo CA privata AWS. I seguenti argomenti mostrano come configurare per CA privata AWS soddisfare gli obiettivi di sicurezza e conformità. Imparerai anche a utilizzarne altri Servizi AWS che ti aiutano a monitorare e proteggere CA privata AWS le tue risorse.

Argomenti

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

L'accesso a CA privata AWS richiede credenziali che AWS possono essere utilizzate per autenticare le richieste. Negli argomenti seguenti vengono fornite informazioni dettagliate su come utilizzare AWS Identity and Access Management (IAM) per proteggere le autorità di certificazione (CA) private mediante il controllo degli accessi.

In CA privata AWS, la risorsa principale con cui lavori è un'autorità di certificazione (CA). Ogni CA privata di proprietà o controllata dall'utente è identificata da un Amazon Resource Name (ARN), con il seguente formato.

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

Il proprietario di una risorsa è l'entità principale dell' AWS account in cui viene creata una AWS risorsa. Negli esempi seguenti viene illustrato il funzionamento.

- Se utilizzi le tue credenziali Utente root dell'account AWS per creare una CA privata, il tuo AWS account possiede la CA.

  > **Important**
  >
  > - Non è consigliabile utilizzarne una Utente root dell'account AWS per creare CA.
  > - Consigliamo vivamente di utilizzare l'autenticazione a più fattori (MFA) ogni volta che si accede. CA privata AWS

- Se crei un utente IAM nel tuo AWS account, puoi concedere a quell'utente l'autorizzazione a creare una CA privata. Tuttavia l'account a cui appartiene l'utente è il proprietario della CA.
- Se crei un ruolo IAM nel tuo AWS account e gli concedi l'autorizzazione a creare una CA privata, chiunque possa assumere il ruolo può creare la CA. Tuttavia, l'account a cui appartiene il ruolo è il proprietario della CA privata.

La policy delle autorizzazioni descrive chi ha accesso a cosa. Nella sezione seguente vengono descritte le opzioni disponibili per la creazione di policy relative alle autorizzazioni.

> **Note**
>
> Questa documentazione descrive l'utilizzo di IAM nel contesto di CA privata AWS. Non vengono fornite informazioni dettagliate sul servizio IAM. Per la documentazione IAM completa, consulta la Guida per l'utente IAM. Per informazioni sulla sintassi e le descrizioni delle policy IAM, consulta AWS Referenza sulla Policy IAM.

CA privata AWS Operazioni e autorizzazioni delle API

Quando configuri le politiche di controllo degli accessi e di autorizzazione che intendi allegare a un'identità IAM (politiche basate sull'identità), utilizza la tabella seguente come riferimento. La prima colonna della tabella elenca ogni operazione API. CA privata AWS È possibile specificare le operazioni nell'elemento `Action` di una policy. Le restanti colonne forniscono ulteriori informazioni.

CA privata AWS Operazioni API	Autorizzazioni richieste	Risorse
CreateCertificateAuthority	`acm-pca:CreateCertificateAuthority` `acm-pca:TagCertificateAuthority` (Richiesto solo quando si crea una CA con 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-`

CA privata AWS Operazioni API	Autorizzazioni richieste	Risorse
		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*

CA privata AWS Operazioni API	Autorizzazioni richieste	Risorse
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*

CA privata AWS Operazioni API	Autorizzazioni richieste	Risorse
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/D
ListPermissions	acm-pca:ListPermissions	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListTags	acm-pca:ListTags	N/D
PutPolicy	acm-pca:PutPolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

CA privata AWS Operazioni API	Autorizzazioni richieste	Risorse
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*

Per fornire l'accesso, aggiungi autorizzazioni ai tuoi utenti, gruppi o ruoli:

- Utenti e gruppi in AWS IAM Identity Center:

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center .

- Utenti gestiti in IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Creating a role for a third-party identity provider (federation) (Creazione di un ruolo per un provider di identità di terze parti [federazione]) nella Guida per l'utente di IAM.

- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Per istruzioni, consulta la pagina Creating a role for an IAM user (Creazione di un ruolo per un utente IAM) nella Guida per l'utente di IAM.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente di IAM.

AWS politiche gestite

CA privata AWS include una serie di politiche AWS gestite predefinite per CA privata AWS amministratori, utenti e revisori. La comprensione di questi criteri può aiutarti a implementare Policy gestite dal cliente.

Scegli una delle politiche elencate di seguito per visualizzare i dettagli e un esempio di codice delle politiche.

AWSPrivateCAFullAccess

Garantisce un controllo amministrativo illimitato.

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

AWSPrivateCAReadOnly

Concede l'accesso limitato alle operazioni API di sola lettura.

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

AWSPrivateCAPrivilegedUser

Garantisce la possibilità di emettere e revocare certificati CA. Questa policy non ha altre funzionalità amministrative e non è in grado di emettere certificati di entità finale. Le autorizzazioni si escludono a vicenda con la policy User.

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

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

AWSPrivateCAUser

Concede la possibilità di emettere e revocare certificati di entità finale. Questa policy non dispone di funzionalità amministrative e non è in grado di emettere certificati emessi da una CA. Le autorizzazioni si escludono a vicenda con la policy. PrivilegedUser

```
{
```

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

```
         "Resource":"*"
      }
   ]
}
```

AWSPrivateCAAuditor

Concedi l'accesso alle operazioni API di sola lettura e l'autorizzazione a generare un rapporto di audit CA.

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

Aggiornamenti alle politiche AWS gestite per CA privata AWS

Nella tabella seguente, visualizza i dettagli sugli aggiornamenti alle politiche AWS gestite CA privata AWS da quando il servizio ha iniziato a tenere traccia di queste modifiche. Per ricevere avvisi

automatici su tutte le modifiche apportate CA privata AWS, iscriviti al feed RSS presente nella Cronologia dei documenti pagina.

Modifiche gestite alle politiche

Modifica	Descrizione	Data
Nuovi nomi delle politiche: • `AWSPrivateCAFullAccess` • `AWSPrivateCAReadOnly` • `AWSPrivateCAPrivilegedUser` • `AWSPrivateCAAuditor` • `AWSPrivateCAUser`	I prefissi dei nomi delle politiche sono stati modificati da a`AWSCertificateManagerPrivateCA` . `AWSPrivateCA` La funzionalità rimane invariata.	13 febbraio 2023

Policy gestite dal cliente

Come best practice, non utilizzare il tuo Utente root dell'account AWS per interagire con AWS, tra cui CA privata AWS. Utilizza invece AWS Identity and Access Management (IAM) per creare un utente IAM, un ruolo IAM o un utente federato. Creare un gruppo di amministratori e aggiungersi a tale gruppo, quindi accedere come amministratore. Aggiungi altri utenti al gruppo, se necessario.

Un'altra best practice consiste nel creare una policy IAM gestita dal cliente che puoi assegnare agli utenti. Le policy gestite dal cliente sono policy standalone basate sulle identità create dagli utenti e che possono essere collegate a più utenti, gruppi o ruoli nell'account AWS . Tale politica limita gli utenti a eseguire solo le CA privata AWS azioni specificate.

La seguente policy gestita dal cliente di esempio consente a un utente di creare un report di audit per la CA. Questo è solo un esempio. Puoi scegliere tutte CA privata AWS le operazioni che desideri. Per ulteriori esempi, consulta Policy inline.

Per creare una policy gestita dal cliente

1. Accedi alla console IAM utilizzando le credenziali di un AWS amministratore.
2. Nel riquadro di navigazione della console, scegliere Policies (Policy).

3. Scegli Crea policy.
4. Scegliere la scheda JSON.
5. Copia il criterio seguente e incollalo nell'editor.

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

6. Scegli Esamina la policy.
7. In Name (Nome) digitare `PcaListPolicy`.
8. (Opzionale) Digita una descrizione.
9. Scegli Crea policy.

Un amministratore può allegare la policy a qualsiasi utente IAM per limitare CA privata AWS le azioni che l'utente può eseguire. Per informazioni su come applicare una politica di autorizzazioni, consulta Modifica delle autorizzazioni per un utente IAM nella Guida per l'utente IAM.

Policy inline

Le policy inline sono policy create, gestite e incorporate direttamente in un utente, gruppo o ruolo. I seguenti esempi di policy mostrano come assegnare le autorizzazioni per eseguire azioni. CA privata AWS Per informazioni generali sulle politiche in linea, consulta Working with Inline Policies nella IAM User Guide. Puoi utilizzare AWS Management Console, the AWS Command Line Interface (AWS CLI) o l'API IAM per creare e incorporare politiche in linea.

> **Important**
>
> Consigliamo vivamente di utilizzare l'autenticazione a più fattori (MFA) ogni volta che si accede. CA privata AWS

Argomenti

Elencare CA private

La policy seguente permette a un utente di elencare tutte le CA private in un account.

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

Recupero di un certificato CA privato

La policy seguente permette a un utente di recuperare un certificato CA privato specifico.

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

```
   }
}
```

Importazione di un certificato CA privato

La policy seguente permette a un utente di importare un certificato CA privato.

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

Eliminazione di una CA privata

La policy seguente permette a un utente di eliminare un certificato CA privato specifico.

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

Tag-on-create: Allegare tag a una CA al momento della creazione

La seguente politica consente a un utente di applicare i tag durante la creazione di una CA.

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

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

Tag-on-create: Etichettatura limitata

La seguente tag-on-create politica impedisce l'uso della coppia chiave-valore Environment=Prod durante la creazione della CA. È consentita l'etichettatura con altre coppie chiave-valore.

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

Controllo dell'accesso alla CA privata tramite tag

La seguente politica consente l'accesso solo alle CA con la coppia chiave-valore Environment=. PreProd Richiede inoltre che le nuove CA includano questo tag.

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

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

Accesso in sola lettura a CA privata AWS

La policy seguente permette a un utente di descrivere ed elencare le autorità di certificazione private e di recuperare il certificato emesso da una CA privata e la catena di certificati.

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

Accesso completo a CA privata AWS

La seguente politica consente a un utente di eseguire qualsiasi CA privata AWS azione.

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

Accesso amministratore a tutte le risorse AWS

La seguente politica consente a un utente di eseguire qualsiasi azione su qualsiasi AWS risorsa.

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

Procedure consigliate di sicurezza per l'accesso tra più account alle CA private

Un CA privata AWS amministratore può condividere una CA con i principali (utenti, ruoli, ecc.) di un altro AWS account. Una volta ricevuta e accettata una condivisione, il mandante può utilizzare la CA per emettere certificati di entità finale utilizzando CA privata AWS le nostre risorse. AWS Certificate Manager Il principale può utilizzare la CA per emettere certificati CA subordinati utilizzando. CA privata AWS

> **Important**
>
> Gli addebiti associati a un certificato emesso in uno scenario che coinvolge più account vengono fatturati all' AWS account che emette il certificato.

Per condividere l'accesso a una CA, CA privata AWS gli amministratori possono scegliere uno dei seguenti metodi:

- Utilizza AWS Resource Access Manager (RAM) per condividere la CA come risorsa con un responsabile di un altro account o con AWS Organizations. La RAM è un metodo standard per la condivisione di AWS risorse tra account. Per ulteriori informazioni sulla RAM, consulta la Guida AWS RAM per l'utente. Per ulteriori informazioni in merito AWS Organizations, consulta la Guida AWS Organizations per l'utente.
- Utilizza l' CA privata AWS API o la CLI per allegare una policy basata sulle risorse a una CA, garantendo in tal modo l'accesso a un principale in un altro account. Per ulteriori informazioni, consulta Policy basate su risorse.

La Controllo dell'accesso a una CA privata sezione di questa guida fornisce i flussi di lavoro per concedere l'accesso alle CA in scenari con account singolo e con più account.

Policy basate su risorse

Le politiche basate sulle risorse sono politiche di autorizzazione create e allegate manualmente a una risorsa (in questo caso, una CA privata) anziché all'identità o al ruolo di un utente. Oppure, invece di creare politiche personalizzate, puoi utilizzare AWS politiche gestite per. AWS Private CA Applicando AWS RAM una policy basata sulle risorse, un CA privata AWS amministratore può condividere l'accesso a una CA con un utente di un altro AWS account direttamente o tramite. AWS OrganizationsIn alternativa, un CA privata AWS amministratore può utilizzare le API PCA e PutPolicy, GetPolicyo AWS CLI i comandi corrispondenti put-policy DeletePolicy, get-policy e delete-policy, per applicare e gestire politiche basate sulle risorse.

Per informazioni generali sulle politiche basate sulle risorse, vedere Politiche basate sull'identità e Politiche basate sulle risorse e Controllo dell'accesso tramite le politiche.

Per visualizzare l'elenco delle politiche basate sulle risorse AWS gestite per AWS Private CA, accedi alla libreria delle autorizzazioni gestite nella console e cerca. AWS Resource Access

ManagerCertificateAuthority Come per qualsiasi politica, prima di applicarla, ti consigliamo di applicarla in un ambiente di test per assicurarti che soddisfi i tuoi requisiti.

AWS Certificate Manager (ACM) gli utenti con accesso condiviso tra più account a una CA privata possono emettere certificati gestiti firmati dalla CA. Gli emittenti con più account sono vincolati da una politica basata sulle risorse e hanno accesso solo ai seguenti modelli di certificato per entità finali:

- EndEntityCertificate/V1
- EndEntityClientAuthCertificate/V1
- EndEntityServerAuthCertificate/V1
- BlankEndEntityCertificate_API passa/v1
- BlankEndEntityCertificate_APIC SRPassthrough/v1
- Certificato CA subordinato_ 0/V1 PathLen

Esempi di policy

Questa sezione fornisce esempi di politiche tra account per varie esigenze. In tutti i casi, per applicare una politica viene utilizzato il seguente schema di comandi:

```
$ 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
```

Oltre a specificare l'ARN di una CA, l'amministratore fornisce un ID account o AWS AWS Organizations un ID a cui verrà concesso l'accesso alla CA. Il JSON di ciascuna delle seguenti policy è formattato come file per motivi di leggibilità, ma può anche essere fornito come argomenti CLI in linea.

> **Note**
>
> La struttura delle policy basate sulle risorse JSON mostrate di seguito deve essere seguita con precisione. Solo i campi ID per i principali (il numero di AWS account o l'ID AWS Organizations) e i CA ARN possono essere configurati dai clienti.

1. File: policy1.json — Condivisione dell'accesso a una CA con un utente in un account diverso

Sostituisci *5555* con l'ID dell' AWS account che condivide la CA.

Per la risorsa ARN, sostituisci quanto segue con i tuoi valori:

- *aws*- La AWS partizione. Ad esempio`aws`,`aws-us-gov`,`aws-cn`, ecc.
- *us-east-1*- La AWS regione in cui è disponibile la risorsa, ad esempio`us-west-1`.
- *111122223333*- L'ID dell' AWS account del proprietario della risorsa.
- *11223344-1234-1122-2233-112233445566*- L'ID della risorsa dell'autorità di certificazione.

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

           "AWS":"555555555555"

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

        ],

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

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

2. File: policy2.json — Condivisione dell'accesso a una CA tramite AWS Organizations

   Sostituisci *o-a1b2c3d4z5 con* l'ID. AWS Organizations

   Per la risorsa ARN, sostituisci quanto segue con i tuoi valori:

   - *aws*- La AWS partizione. Ad esempio`aws`,`aws-us-gov`,`aws-cn`, ecc.
   - *us-east-1*- La AWS regione in cui è disponibile la risorsa, ad esempio`us-west-1`.
   - *111122223333*- L'ID dell' AWS account del proprietario della risorsa.
   - *11223344-1234-1122-2233-112233445566*- L'ID della risorsa dell'autorità di certificazione.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ExampleStatementID3",
         "Effect":"Allow",
         "Principal":"*",
         "Action":"acm-pca:IssueCertificate",
         "Resource":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
         "Condition":{
            "StringEquals":{
               "acm-pca:TemplateArn":"arn:aws:acm-pca:::template/
EndEntityCertificate/V1",
               "aws:PrincipalOrgID":"o-a1b2c3d4z5"
            },
            "StringNotEquals":{
               "aws:PrincipalAccount":"111122223333"
            }
```

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

Protezione dei dati in AWS Private Certificate Authority

Il modello di responsabilità AWS condivisa modello di di si applica alla protezione dei dati in AWS Private Certificate Authority. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. Inoltre, sei responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS che utilizzi. Per ulteriori informazioni sulla privacy dei dati, vedi Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog AWS Shared Responsibility Model and GDPR nel Blog sulla sicurezza AWS .

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal

modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-2 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Ti consigliamo vivamente di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori o Servizi AWS utilizzi la console, l'API CA privata AWS o gli SDK. AWS CLI AWS I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Conformità all'archiviazione e alla sicurezza delle chiavi CA privata AWS private

Le chiavi private per le CA private sono archiviate in moduli di sicurezza hardware AWS gestiti (HSM). Gli HSM sono conformi ai requisiti di sicurezza FIPS PUB 140-2 di livello 3 per i moduli crittografici.

Crittografia dei dati in AWS Private CA Connector for Active Directory

AWS Private CA Connector for AD memorizza i dati di configurazione dei clienti relativi a connettori, modelli, registrazioni di directory, nomi principali dei servizi e voci di controllo degli accessi ai gruppi di modelli. Questi dati vengono crittografati in transito e a riposo. Le informazioni sui certificati emessi tramite Connector for AD possono essere scoperte utilizzando l'GetCertificateazione nell' AWS

Private CA API. Nessuna informazione relativa ai certificati emessi o al client o alla macchina che richiede un certificato viene archiviata da AWS.

Convalida della conformità per AWS Private Certificate Authority

I revisori esterni valutano la sicurezza e la conformità nell' AWS Private Certificate Authority ambito di più programmi di AWS conformità. Questi includono SOC, PCI, FedRAMP, HIPAA e altri.

Per un elenco dei AWS servizi che rientrano nell'ambito di specifici programmi di conformità, vedere AWS Servizi rientranti nell'ambito del programma di conformitàServizi AWS in Ambito . Per informazioni generali, vedere Programmi di AWS conformità Programmi di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo CA privata AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Per le organizzazioni che devono crittografare i propri bucket Amazon S3, i seguenti argomenti descrivono come configurare la crittografia per ospitare gli asset: CA privata AWS
  - Crittografia dei report di audit
  - Crittografia dei CRL
- Guide rapide su sicurezza e conformità Guide introduttive implementazione illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla sicurezza e la conformità. AWS
- Whitepaper sull'architettura per la sicurezza e la conformità HIPAA: questo white paper descrive in che modo le aziende possono utilizzare per creare applicazioni conformi all'HIPAA. AWS
- AWS Risorse per la conformità Risorse per la conformità: questa raccolta di potrebbe riguardare il settore e la località in cui operate.
- Valutazione delle risorse con le regole nella Guida per gli AWS Config sviluppatori: il AWS Config servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida del settore e alle normative.
- AWS Security Hub— Questo AWS servizio offre una visione completa dello stato di sicurezza dell'utente, AWS che consente di verificare la conformità agli standard e alle best practice del settore della sicurezza.

Utilizzo dei report di controllo con la tua CA privata

È possibile creare un report di audit per elencare i certificati emessi o revocati dalla CA privata. Il report viene salvato in un bucket S3 nuovo o esistente specificato nell'input.

Per informazioni sull'aggiunta della protezione di crittografia ai report di audit, consulta Crittografia dei report di controllo .

Il file del rapporto di controllo ha il percorso e il nome di file seguenti. L'ARN per un bucket Amazon S3 è il valore per. `bucket-name` `CA_ID`è l'identificatore univoco di una CA emittente. `UUID`è l'identificatore univoco di un rapporto di audit.

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

È possibile generare un nuovo report ogni 30 minuti e scaricarlo dal bucket. Nell'esempio seguente viene illustrato un report 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
```

Il seguente esempio mostra un report formattato 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**
>
> Quando AWS Certificate Manager rinnova un certificato, il rapporto di audit CA privato compila il campo con. `requestedByServicePrincipal` `acm.amazonaws.com` Ciò indica che il AWS Certificate Manager servizio ha richiamato l'`IssueCertificate`azione dell' CA privata AWS API per conto di un cliente per rinnovare il certificato.

Preparazione di un bucket Amazon S3 per i report di controllo

Per archiviare i report di controllo, devi preparare un bucket Amazon S3. Per ulteriori informazioni, consulta Come si crea un bucket S3?

Il tuo bucket S3 deve essere protetto da una politica di autorizzazioni allegata. Gli utenti e i responsabili del servizio autorizzati richiedono `Put` l'autorizzazione per consentire di CA privata AWS inserire oggetti nel bucket e l'autorizzazione per recuperarli. `Get` Ti consigliamo di applicare la politica mostrata di seguito, che limita l'accesso sia a un AWS account che all'ARN di una CA privata. Per ulteriori informazioni, consulta Aggiungere una policy sui bucket utilizzando la console Amazon S3.

> **Note**
>
> Durante la procedura della console per la creazione di un rapporto di controllo, puoi scegliere di consentire la CA privata AWS creazione di un nuovo bucket e applicare una politica di autorizzazioni predefinita. La politica predefinita non applica alcuna `SourceArn` restrizione alla CA ed è quindi più permissiva della politica consigliata. Se scegli l'impostazione predefinita, puoi sempre modificarla in un secondo momento.

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

Creazione di un rapporto di controllo

È possibile creare un rapporto di controllo dalla console o dal AWS CLI.

Per creare un report di audit (console)

1. Accedi al tuo AWS account e apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home.
2. Nella pagina Autorizzazioni di certificazione private, scegli la tua CA privata dall'elenco.
3. Dal menu Operazioni scegliere Genera report di audit.
4. In Audit report destination, per Creare un nuovo bucket S3? , scegli Sì e digita un nome di bucket univoco oppure scegli No e scegli un bucket esistente dall'elenco.

   Se scegli Sì, CA privata AWS crea e allega la politica predefinita al tuo bucket. Se scegli No, devi allegare una policy al tuo bucket prima di poter generare un rapporto di controllo. Utilizza il modello di policy descritto inPreparazione di un bucket Amazon S3 per i report di controllo. Per informazioni su come allegare una policy, consulta Aggiungere una bucket policy usando la console Amazon S3
5. In Formato di output, scegli JSON per JavaScript Object Notation o CSV per i valori separati da virgole.
6. Scegliere Generate audit report (Genera report di audit).

Per creare un report di audit (AWS CLI)

1. Se non hai già un bucket S3 da usare, creane uno.
2. Allega una policy al tuo bucket. Utilizza il modello di policy descritto inPreparazione di un bucket Amazon S3 per i report di controllo. Per informazioni su come allegare una policy, consulta Aggiungere una bucket policy usando la console Amazon S3
3. Usa il comando create-certificate-authority-audit-report per creare il report di audit e inserirlo nel bucket S3 preparato.

```
$ aws acm-pca create-certificate-authority-audit-report \
--certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
--s3-bucket-name bucket_name \
--audit-report-response-format JSON
```

Recupero di un rapporto di audit

Per recuperare un rapporto di controllo per l'ispezione, utilizza la console Amazon S3, l'API, la CLI o l'SDK. Per ulteriori informazioni, consulta Downloading an object nella Amazon Simple Storage Service User Guide.

Crittografia dei report di controllo

Facoltativamente, puoi configurare la crittografia sul bucket Amazon S3 contenente i report di controllo. CA privata AWS supporta due modalità di crittografia per gli asset in S3:

- Crittografia automatica lato server con chiavi AES-256 gestite da Amazon S3.
- Crittografia gestita dal cliente utilizzando AWS Key Management Service e configurata secondo le tue specifiche. AWS KMS key

> **Note**
>
> CA privata AWS non supporta l'utilizzo di chiavi KMS predefinite generate automaticamente da S3.

Nelle procedure seguenti viene descritto come impostare ciascuna delle opzioni di crittografia.

Per configurare la crittografia automatica

Completare la procedura seguente per abilitare la crittografia lato server S3.

1. Apri la console Amazon S3 all'indirizzo https://console.aws.amazon.com/s3/.
2. Nella tabella Bucket, scegli il bucket che conterrà i tuoi asset. CA privata AWS
3. Nella pagina relativa al bucket scegliere la scheda Proprietà .
4. Scegliere la scheda di crittografia predefinita.
5. Scegli Abilita .
6. Scegli la chiave Amazon S3 (SSE-S3).
7. Seleziona Salva modifiche.

Per configurare la crittografia personalizzata

Completa i seguenti passaggi per abilitare la crittografia utilizzando una chiave personalizzata.

1. Apri la console Amazon S3 all'indirizzo https://console.aws.amazon.com/s3/.
2. Nella tabella Bucket, scegli il bucket che conterrà i tuoi CA privata AWS asset.
3. Nella pagina relativa al bucket scegliere la scheda Proprietà .
4. Scegliere la scheda di crittografia predefinita.
5. Scegli Abilita .
6. Scegli la AWS Key Management Service chiave (SSE-KMS).
7. Scegli tra AWS KMS le tue chiavi o Inserisci AWS KMS key ARN.
8. Seleziona Salva modifiche.
9. (Facoltativo) Se non disponi già di una chiave KMS, creane una utilizzando il seguente comando AWS CLI create-key:

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

   L'output contiene l'ID della chiave e l'Amazon Resource Name (ARN) della chiave KMS. Di seguito è riportato un esempio di 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. Utilizzando i passaggi seguenti, si concede al CA privata AWS servizio l'autorizzazione principale a utilizzare la chiave KMS. Per impostazione predefinita, tutte le chiavi KMS sono private; solo il proprietario della risorsa può utilizzare una chiave KMS per crittografare e decrittografare i dati. Tuttavia, il proprietario della risorsa può concedere ad altri utenti e risorse le autorizzazioni per accedere alla chiave KMS. Il responsabile del servizio deve trovarsi nella stessa regione in cui è archiviata la chiave KMS.

a. Innanzitutto, salva la politica predefinita per la tua chiave KMS `policy.json` utilizzando il seguente comando: get-key-policy

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

b. Apri il file `policy.json` in un editor di testo. Seleziona una delle seguenti dichiarazioni politiche e aggiungila alla politica esistente.

Se la tua chiave bucket Amazon S3 è abilitata, usa la seguente dichiarazione:

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

Se la tua chiave bucket Amazon S3 è disabilitata, usa la seguente dichiarazione:

```
{
   "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. Infine, applica la policy aggiornata utilizzando il seguente comando: put-key-policy

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

Sicurezza dell'infrastruttura in AWS Private Certificate Authority

In quanto servizio gestito, AWS Private Certificate Authority è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi AWS di sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzate chiamate API AWS pubblicate per accedere AWS Private CA attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. O puoi utilizzare AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

CA privata AWS Endpoint VPC ()AWS PrivateLink

Puoi creare una connessione privata tra il tuo VPC e CA privata AWS configurando un endpoint VPC di interfaccia. Gli endpoint di interfaccia sono basati su una tecnologia per AWS PrivateLinkl'accesso privato alle operazioni API. CA privata AWS AWS PrivateLink indirizza tutto il traffico di rete tra il tuo VPC e CA privata AWS attraverso la rete Amazon, evitando l'esposizione su Internet aperto. Ogni endpoint VPC è rappresentato da una o più interfacce di rete elastiche con indirizzi IP privati nelle sottoreti del VPC.

L'interfaccia VPC endpoint collega il tuo VPC direttamente CA privata AWS senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione. AWS Direct Connect Le istanze del tuo VPC non necessitano di indirizzi IP pubblici per comunicare con CA privata AWS l'API.

Per utilizzarlo CA privata AWS tramite il tuo VPC, devi connetterti da un'istanza che si trova all'interno del VPC. In alternativa, puoi connettere la tua rete privata al tuo VPC utilizzando un AWS Virtual Private Network (AWS VPN) o. AWS Direct Connect Per informazioni in merito AWS VPN, consulta Connessioni VPN nella Guida per l'utente di Amazon VPC. Per informazioni su AWS Direct Connect, consulta Creazione di una connessione nella Guida per l'AWS Direct Connect utente.

CA privata AWS non richiede l'uso di AWS PrivateLink, ma lo consigliamo come ulteriore livello di sicurezza. Per ulteriori informazioni sugli AWS PrivateLink endpoint VPC, consulta Accesso ai servizi tramite. AWS PrivateLink

Considerazioni sugli endpoint CA privata AWS VPC

Prima di configurare gli endpoint VPC dell'interfaccia per CA privata AWS, tieni presente le seguenti considerazioni:

- CA privata AWS potrebbe non supportare gli endpoint VPC in alcune zone di disponibilità. Quando crei un endpoint VPC, verifica innanzitutto il supporto nella console di gestione. Le zone di disponibilità non supportate sono contrassegnate come «Servizio non supportato in questa zona di disponibilità».
- Gli endpoint VPC non supportano le richieste tra regioni. Assicurati di creare l'endpoint nella stessa regione in cui prevedi di inviare le chiamate API a CA privata AWS.
- Gli endpoint VPC supportano solo il DNS fornito da Amazon tramite Amazon Route 53. Se si desidera utilizzare il proprio DNS, è possibile usare l'inoltro condizionale sul DNS. Per ulteriori informazioni, consulta Set opzioni DHCP nella Guida per l'utente di Amazon VPC.

- Il gruppo di sicurezza collegato all'endpoint VPC deve consentire le connessioni in entrata sulla porta 443 dalla sottorete privata del VPC.
- AWS Certificate Manager non supporta gli endpoint VPC.
- Gli endpoint FIPS (e le relative aree) non supportano gli endpoint VPC.

CA privata AWS L'API attualmente supporta gli endpoint VPC nei seguenti casi: Regioni AWS

- Stati Uniti orientali (Ohio)
- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti occidentali (California settentrionale)
- Stati Uniti occidentali (Oregon)
- Africa (Città del Capo)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Europa (Francoforte)
- Europa (Irlanda)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- Europa (Milano)
- Israele (Tel Aviv)
- Medio Oriente (Bahrein)
- Sud America (San Paolo)

Creazione degli endpoint VPC per CA privata AWS

Puoi creare un endpoint VPC per il CA privata AWS servizio utilizzando la console VPC all'indirizzo https://console.aws.amazon.com/vpc/ oppure il. AWS Command Line Interface Per ulteriori informazioni, consulta la procedura Creating an Interface Endpoint nella Amazon VPC User Guide. CA privata AWS supporta l'effettuazione di chiamate a tutte le sue operazioni API all'interno del tuo VPC.

Se hai abilitato i nomi host DNS privati per l'endpoint, l'endpoint predefinito ora si risolve nel tuo CA privata AWS endpoint VPC. Per un elenco completo degli endpoint di servizio predefiniti, consulta Endpoint e quote del servizio.

Se non hai abilitato i nomi host DNS privati, Amazon VPC fornisce un nome di endpoint DNS che puoi utilizzare nel seguente formato:

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

> **Note**
>
> La *regione* di valore rappresenta l'identificatore di regione per una regione supportata da CA privata AWS, ad esempio `us-east-2` per la AWS regione Stati Uniti orientali (Ohio). Per un elenco di CA privata AWS, consulta AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.

Per ulteriori informazioni, consulta CA privata AWS VPC endpoints (AWS PrivateLink) nella Amazon VPC User Guide.

Creazione di una policy di endpoint VPC per CA privata AWS.

Puoi creare una policy per gli endpoint Amazon VPC CA privata AWS per specificare quanto segue:

- Il principale che può eseguire operazioni.
- Le azioni che possono essere eseguite
- Le risorse sui cui si possono eseguire le azioni

Per ulteriori informazioni, consultare Controllo degli accessi ai servizi con endpoint VPC nella Guida per l'utente di Amazon VPC.

Esempio: policy degli endpoint VPC per le azioni CA privata AWS

Se collegata a un endpoint, la seguente politica consente a tutti i principali di accedere alle CA privata AWS azioni`IssueCertificate`,,,`DescribeCertificateAuthority`, `GetCertificate` e. `GetCertificateAuthorityCertificate` `ListPermissions` `ListTags` La risorsa in ogni stanza è una CA privata. La prima stanza autorizza la creazione di certificati di entità finale utilizzando la CA privata e il modello di certificato specificati. Se non si desidera controllare il modello utilizzato, la sezione `Condition` non è necessaria. Tuttavia, la rimozione di questo consente a tutte le entità di creare certificati emessi da una CA e certificati di entità finale.

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

```
        }
      ]
   }
```

Logging e monitoraggio per AWS Private Certificate Authority

Il monitoraggio è un elemento importante per mantenere l'affidabilità, la disponibilità e le prestazioni delle AWS Private Certificate Authority vostre AWS soluzioni. È necessario raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica.

I seguenti argomenti descrivono gli strumenti di AWS monitoraggio del cloud disponibili per l'uso con. CA privata AWS

Argomenti

- Metriche supportate CloudWatch
- Utilizzo CloudWatch degli eventi
- Usando CloudTrail

Metriche supportate CloudWatch

Amazon CloudWatch è un servizio di monitoraggio delle AWS risorse. Puoi utilizzarlo CloudWatch per raccogliere e tracciare metriche, impostare allarmi e reagire automaticamente ai cambiamenti nelle tue AWS risorse. CloudWatch le metriche vengono pubblicate almeno una volta.

CA privata AWS supporta le seguenti CloudWatch metriche.

Parametro	Descrizione
`CRLGenerated`	È stato generato un elenco di revoche di certificati (CRL). Questo parametro si applica solo a una CA privata.
`MisconfiguredCRLBucket`	Il bucket S3 specificato per il CRL non è configurato correttamente. Controllare la policy del bucket. Questo parametro si applica solo a una CA privata.

Parametro	Descrizione
`Time`	Il tempo in millisecondi che intercorre tra una richiesta di emissione e il completamento (o il fallimento) dell'emissione. Questa metrica si applica solo all'operazione. IssueCertificate
`Success`	Un certificato è stato rilasciato correttamente. Questa metrica si applica solo all'IssueCert ificateoperazione.
`Failure`	Operazione non riuscita. Questa metrica si applica solo all'IssueCertificateoperazione.

Per ulteriori informazioni sulle CloudWatch metriche, consulta i seguenti argomenti:

- Utilizzo di Amazon CloudWatch Metrics
- Creazione di CloudWatch allarmi Amazon

Utilizzo CloudWatch degli eventi

Puoi utilizzare Amazon CloudWatch Events per automatizzare AWS i tuoi servizi e rispondere automaticamente a eventi di sistema come problemi di disponibilità delle applicazioni o modifiche delle risorse. Gli eventi derivanti dai AWS servizi vengono trasmessi a CloudWatch Events quasi in tempo reale. Puoi scrivere regole semplici per indicare quali eventi ti interessano e le azioni automatiche da intraprendere quando un evento corrisponde a una regola. CloudWatch Gli eventi vengono pubblicati almeno una volta. Per ulteriori informazioni, vedere Creazione di una regola per CloudWatch gli eventi che si attiva su un evento.

CloudWatch Gli eventi vengono trasformati in azioni utilizzando Amazon EventBridge. Con EventBridge, puoi utilizzare gli eventi per attivare obiettivi tra cui AWS Lambda funzioni, AWS Batch job, argomenti di Amazon SNS e molti altri. Per ulteriori informazioni, consulta What Is Amazon EventBridge?

Successo o fallimento durante la creazione di una CA privata

Questi eventi vengono attivati dall'CreateCertificateAuthorityoperazione.

Riuscito

In caso di esito positivo, l'operazione restituisce l'ARN della nuova 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":"success"
   }
}
```

Errore

In caso di errore, l'operazione restituisce un ARN per la CA. Utilizzando l'ARN, è possibile chiamare DescribeCertificateAuthorityper determinare lo stato della 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"
   }
}
```

Successo o fallimento nell'emissione di un certificato

Questi eventi vengono attivati dall'IssueCertificateoperazione.

Riuscito

In caso di esito positivo, l'operazione restituisce gli ARN della CA e del nuovo certificato.

```
{
   "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"
   }
}
```

Errore

In caso di errore, l'operazione restituisce un certificato ARN e l'ARN della CA. Con il certificato ARN, puoi chiamare GetCertificateper visualizzare il motivo dell'errore.

```
{
   "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"
   }
}
```

Riuscita della revoca di un certificato

Questo evento viene attivato dall'operazione. RevokeCertificate

Nessun evento viene inviato se la revoca ha esito negativo o se il certificato è già stato revocato.

Riuscito

In caso di esito positivo, l'operazione restituisce gli ARN della CA e del certificato revocato.

```
{
   "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"
   }
}
```

Successo o fallimento durante la generazione di un CRL

Questi eventi vengono attivati dall'RevokeCertificateoperazione, che dovrebbe comportare la creazione di un elenco di revoca dei certificati (CRL).

Riuscito

In caso di esito positivo, l'operazione restituisce l'ARN della CA associata al 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"
   }
}
```

Errore 1: impossibile salvare il CRL su Amazon S3 a causa di un errore di autorizzazione

Verifica le autorizzazioni del tuo bucket Amazon S3 se si verifica questo errore.

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

Errore 2: impossibile salvare il CRL su Amazon S3 a causa di un errore interno

Riprovare l'operazione se si verifica questo errore.

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

Errore 3: CA privata AWS impossibile creare un CRL

Per risolvere questo errore, verifica i parametri CloudWatch .

```
{
   "version":"0",
   "id":"event_ID",
   "detail-type":"ACM Private CA CRL Generation",
   "source":"aws.acm-pca",
   "account":"account",
   "time":"2019-11-07T23:01:25Z",
   "region":"region",
   "resources":[
      "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
   ],
   "detail":{
      "result":"failure",
      "reason":"Failed to generate CRL. Internal failure."
   }
}
```

Successo o fallimento durante la creazione di un rapporto di audit CA

Questi eventi vengono attivati dall'CreateCertificateAuthorityAuditReportoperazione.

Riuscito

In caso di esito positivo, l'operazione restituisce l'ARN della CA e l'ID del report di 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"
   }
}
```

Errore

Un rapporto di controllo può avere esito negativo CA privata AWS quando non sono PUT disponibili le autorizzazioni sul bucket Amazon S3, quando la crittografia è abilitata nel bucket o per altri motivi.

```
{
   "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"
   }
}
```

Usando CloudTrail

Puoi AWS CloudTrailutilizzarlo per registrare le chiamate API effettuate da AWS Private Certificate Authority. Per ulteriori informazioni, consulta i seguenti argomenti.

Argomenti

- Creazione di una policy
- Recupero di una politica
- Eliminazione di una policy
- Creazione di un'autorità di certificazione
- Genera CRL
- Genera una risposta OCS
- Creazione di un rapporto di controllo
- Eliminazione di un'autorità di certificazione
- Ripristino di un'autorità di certificazione
- Descrizione di un'autorità di certificazione
- Recupero di un certificato di autorità di certificazione
- Recupero della richiesta di firma dell'autorità di certificazione
- Recupero di un certificato
- Importazione di un certificato dell'autorità di certificazione
- Emissione di un certificato
- Elenco delle autorità di certificazione
- Come elencare i tag
- Revoca di un certificato
- Etichettatura delle autorità di certificazione private
- Rimozione di tag da un'autorità di certificazione privata
- Aggiornamento di un'autorità di certificazione

Creazione di una policy

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'PutPolicyoperazione.


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

Recupero di una politica

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'GetPolicyoperazione.


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

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

```
    "eventID":"event_ID",
    "readOnly":true,
    "eventType":"AwsApiCall",
    "managementEvent":true,
    "eventCategory":"Management",
    "recipientAccountId":"account"
}
```

Eliminazione di una policy

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'DeletePolicyoperazione.

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

            },
            "attributes":{
                "mfaAuthenticated":"false",
                "creationDate":"2021-02-26T21:23:17Z"
            }
        }
    },
    "eventTime":"2021-02-26T21:23:31Z",
    "eventSource":"acm-pca.amazonaws.com",
    "eventName":"DeletePolicy",
    "awsRegion":"region",
    "sourceIPAddress":"IP_address",
    "userAgent":"agent",
```

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

Creazione di un'autorità di certificazione

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'CreateCertificateAuthorityoperazione.

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

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

Genera CRL

L' CloudTrail esempio seguente mostra il record di un evento GenerateCRL.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "account",
        "invokedBy": "acm-pca.amazonaws.com"
    },
    "eventTime": "2021-02-09T17:37:45Z",
    "eventSource": "acm-pca.amazonaws.com",
    "eventName": "GenerateCRL",
    "awsRegion": "region",
    "sourceIPAddress": "acm-pca.amazonaws.com",
    "userAgent": "acm-pca.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
```

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

Genera una risposta OCS

L' CloudTrail esempio seguente mostra il record di un evento GenerateOCSResponse.

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

Creazione di un rapporto di controllo

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'CreateCertificateAuthorityAuditReportoperazione.


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


Eliminazione di un'autorità di certificazione

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'DeleteCertificateAuthorityoperazione. In questo esempio, l'autorità di certificazione non può essere eliminata perché è nello stato `ACTIVE`.

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

Ripristino di un'autorità di certificazione

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'RestoreCertificateAuthorityoperazione. In questo esempio, l'autorità di certificazione non può essere ripristinata perché non è nello stato `DELETED`.

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

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

Descrizione di un'autorità di certificazione

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'DescribeCertificateAuthorityoperazione.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T21:58:18Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"DescribeCertificateAuthority",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566"
```

```
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Recupero di un certificato di autorità di certificazione

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'GetCertificateAuthorityCertificateoperazione.

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

Recupero della richiesta di firma dell'autorità di certificazione

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'GetCertificateAuthorityCsroperazione.

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


Recupero di un certificato

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'GetCertificateoperazione.


```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-01-26T22:22:54Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"GetCertificate",
   "awsRegion":"region",
```

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

Importazione di un certificato dell'autorità di certificazione

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'ImportCertificateAuthorityCertificateoperazione.

```
{
   "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"
}
```

Emissione di un certificato

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'IssueCertificateoperazione.

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

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

```
   "recipientAccountId":"account"
}
```

Elenco delle autorità di certificazione

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'ListCertificateAuthoritiesoperazione.

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

Come elencare i tag

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'ListTagsoperazione.

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

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

Revoca di un certificato

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'RevokeCertificateoperazione.

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

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

Etichettatura delle autorità di certificazione private

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'TagCertificateAuthorityoperazione.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-02-02T00:18:48Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"TagCertificateAuthority",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
```

```
      "tags":[
         {
            "key":"Admin",
            "value":"Alice"
         }
      ]
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Rimozione di tag da un'autorità di certificazione privata

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'UntagCertificateAuthorityoperazione.

```
{
   "eventVersion":"1.05",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"account",
      "arn":"arn:aws:iam::account:user/name",
      "accountId":"account",
      "accessKeyId":"key_ID"
   },
   "eventTime":"2018-02-02T00:21:50Z",
   "eventSource":"acm-pca.amazonaws.com",
   "eventName":"UntagCertificateAuthority",
   "awsRegion":"region",
   "sourceIPAddress":"IP_address",
   "userAgent":"agent",
   "requestParameters":{
      "certificateAuthorityArn":"arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
      "tags":[
         {
            "key":"Admin",
            "value":"Alice"
         }
      ]
```

```
   },
   "responseElements":null,
   "requestID":"request_ID",
   "eventID":"event_ID",
   "eventType":"AwsApiCall",
   "recipientAccountId":"account"
}
```

Aggiornamento di un'autorità di certificazione

L' CloudTrail esempio seguente mostra i risultati di una chiamata all'UpdateCertificateAuthorityoperazione.

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

      "revocationConfiguration":{
         "crlConfiguration":{
            "enabled":true,
            "expirationInDays":3650,
            "customCname":"your-custom-name",
            "s3BucketName":"your-bucket-name"
         }
      },
      "status":"DISABLED"
   },
   "responseElements":null,
```

```
    "requestID":"request_ID",
    "eventID":"event_ID",
    "eventType":"AwsApiCall",
    "recipientAccountId":"account"
}
```

Pianificazione dell' CA privata AWS implementazione

CA privata AWS offre un controllo completo e basato sul cloud sulla PKI (infrastruttura a chiave pubblica) privata dell'organizzazione, che si estende da un'autorità di certificazione (CA) principale, alle CA subordinate, fino ai certificati di entità finale. Una pianificazione approfondita è essenziale per una PKI sicura, gestibile, estensibile e adatta alle esigenze dell'organizzazione. In questa sezione vengono fornite indicazioni sulla progettazione di una gerarchia CA, sulla gestione dei cicli di vita dei certificati delle autorità di certificazione private e delle entità finali private e sull'applicazione delle procedure consigliate per la sicurezza.

Questa sezione descrive come prepararsi all'uso prima di creare un' CA privata AWS autorità di certificazione (CA) privata. Spiega inoltre la possibilità di aggiungere il supporto per la revoca tramite l'Online Certificate Status Protocol (OCSP) o un elenco di revoca dei certificati (CRL).

Inoltre, è necessario stabilire se l'organizzazione preferisce ospitare le proprie credenziali CA root private in locale anziché presso. AWS In tal caso, è necessario configurare e proteggere una PKI privata autogestita prima di utilizzarla. CA privata AWS In questo scenario, si crea quindi una CA subordinata CA privata AWS supportata da una CA principale esterna a. CA privata AWS Per ulteriori informazioni, vedere Installazione di un certificato CA subordinato firmato da una CA principale esterna.

Argomenti

Configurazione AWS dell'account e del AWS CLI

Se non sei già un cliente Amazon Web Services (AWS), devi registrarti per poter utilizzare CA privata AWS. Il tuo account ha automaticamente accesso a tutti i servizi disponibili, ma ti vengono addebitati solo i servizi che utilizzi.

> **Note**
>
> CA privata AWS non è disponibile nel piano AWS gratuito.

Argomenti

Iscriviti per un Account AWS

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a un Account AWS

1. Apri la pagina all'indirizzo https://portal.aws.amazon.com/billing/signup.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, Utente root dell'account AWSviene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente amministrativo e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

AWS ti invia un'email di conferma dopo il completamento della procedura di registrazione. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo https://aws.amazon.com/ e selezionando Il mio account.

Creazione di un utente amministratore

Dopo la registrazione Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi AWS Management Consolecome proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Signing in as the root user della Guida per l'utente di Accedi ad AWS .
2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

   Per istruzioni, consulta Abilitare un dispositivo MFA virtuale per l'utente Account AWS root (console) nella Guida per l'utente IAM.

Creazione di un utente amministratore

1. Abilita Centro identità IAM.

   Per istruzioni, consulta Abilitazione di AWS IAM Identity Center nella Guida per l'utente di AWS IAM Identity Center .
2. In IAM Identity Center, assegna l'accesso amministrativo a un utente amministratore.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

  Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta AWS Accedere al portale di accesso nella Guida per l'Accedi ad AWS utente.

Installa il AWS Command Line Interface

Se non l'hai installato AWS CLI ma desideri utilizzarlo, segui le istruzioni in AWS Command Line Interface. In questa guida, supponiamo che tu abbia configurato l'endpoint, la regione e i dettagli di autenticazione e omettiamo questi parametri dai comandi di esempio.

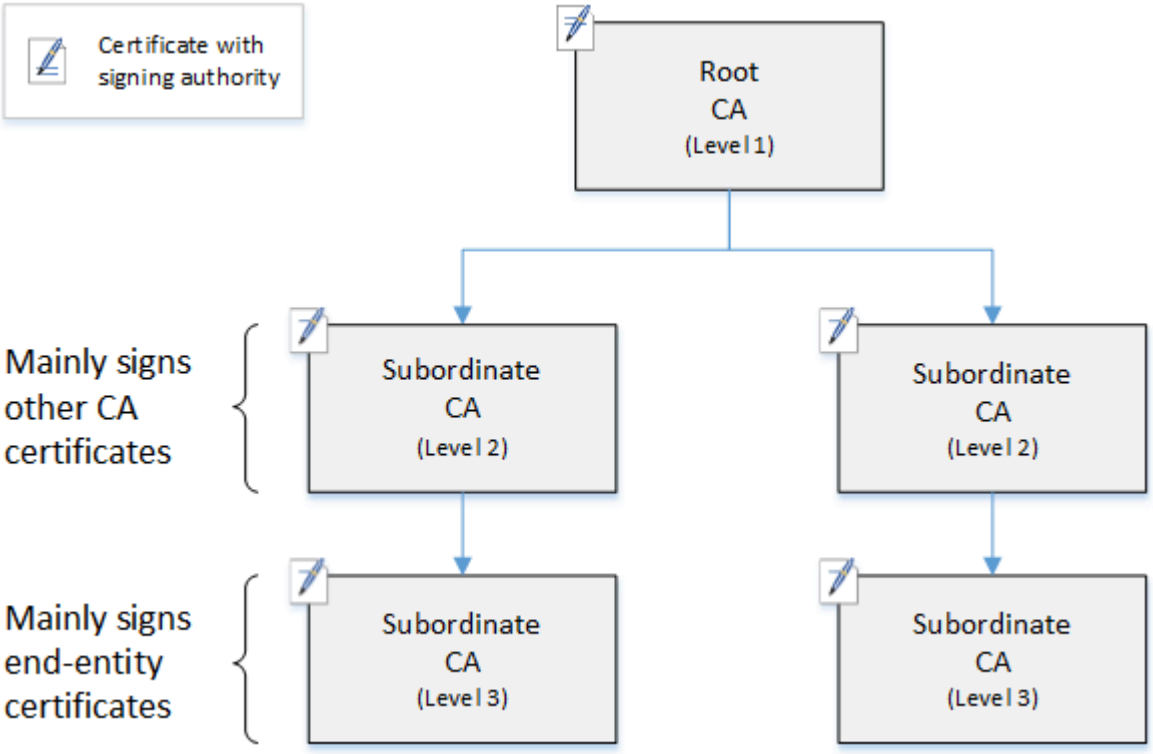
Progettazione di una gerarchia CA

Con CA privata AWS, puoi creare una gerarchia di autorità di certificazione con un massimo di cinque livelli. La CA principale, nella parte superiore di una struttura gerarchica, può avere un numero qualsiasi di rami. La CA principale può avere fino a quattro livelli di CA subordinate su ogni ramo. È inoltre possibile creare più gerarchie, ognuna con una propria root.

Una gerarchia CA ben progettata offre i seguenti vantaggi:

- Controlli granulari di sicurezza appropriati per ogni CA
- Divisione delle attività amministrative per migliorare il bilanciamento del carico e la sicurezza
- Utilizzo di CA con attendibilità limitata e revocabile per le operazioni quotidiane
- Periodi di validità e limiti del percorso del certificato

Il diagramma seguente illustra una semplice gerarchia CA a tre livelli.



Ogni CA nell'albero è supportata da un certificato X.509 v3 con autorità di firma (simboleggiata dall'icona). pen-and-paper Ciò significa che, come CA, è possibile firmare altri certificati subordinati. Quando una CA firma il certificato di una CA di livello inferiore, conferisce un'autorità limitata e revocabile al certificato firmato. La CA principale nel livello 1 firma i certificati emessi da una CA

subordinata di alto livello nel livello 2. Queste CA, a loro volta, firmano i certificati per le CA di livello 3 utilizzati dagli amministratori PKI (infrastruttura a chiave pubblica) che gestiscono i certificati delle entità finali.

La protezione in una gerarchia CA deve essere configurata per essere più forte nella parte superiore della struttura. Questa disposizione protegge il certificato emesso da una CA root e la relativa chiave privata. La CA principale ancora l'attendibilità per tutte le CA subordinate e i certificati dell'entità finale sottostanti. Sebbene il danno localizzato possa derivare dalla compromissione di un certificato di entità finale, la compromissione della root distrugge l'attendibilità nell'intera PKI. Le CA root e subordinate di alto livello vengono utilizzate solo di rado (in genere per firmare altri certificati CA). Di conseguenza, essi sono strettamente sottoposti a audit e sottoposti a audit per garantire un minor rischio di compromissione. Ai livelli inferiori della gerarchia, la protezione è meno restrittiva. Questo approccio consente le attività amministrative di routine di emissione e revoca dei certificati di entità finale per utenti, host di computer e servizi software.

> **Note**
>
> L'utilizzo di una CA root per firmare un certificato subordinata è un evento raro che si verifica solo in una manciata di circostanze:
>
> - Quando viene creata la PKI
> - Quando un'autorità di certificazione di alto livello deve essere sostituita
> - Quando è necessario configurare un risponditore elenco di revoche di certificati (CRL) o OCSP (Online Certificate Status Protocol)
>
> Root e altre CA di alto livello richiedono processi operativi altamente sicuri e protocolli di controllo degli accessi.

Argomenti

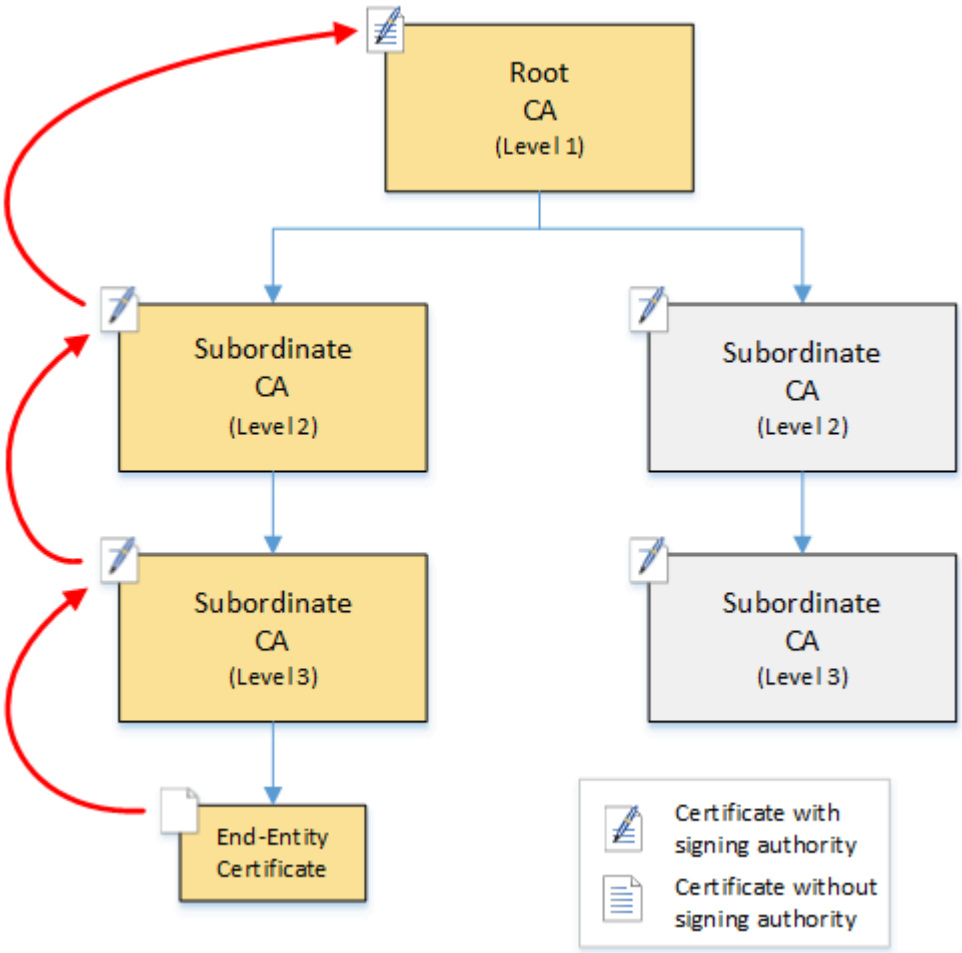
- Convalida dei certificati di entità finale
- Pianificazione della struttura di una gerarchia CA
- Impostazione dei vincoli di lunghezza sul percorso di certificazione

Convalida dei certificati di entità finale

I certificati delle entità finali derivano la loro attendibilità da un percorso di certificazione che conduce attraverso le CA subordinate a una CA root. Quando un browser Web o un altro client viene presentato con un certificato di entità finale, tenta di costruire una catena di attendibilità. Ad esempio, è possibile verificare che il nome distinto dell'emittente del certificato e il nome distinto dell'oggetto corrispondano ai campi corrispondenti del certificato emesso da una CA emittente. La corrispondenza continuerebbe a ogni livello successivo della gerarchia fino a quando il client raggiunge una root attendibile contenuta nel relativo archivio attendibili.

L'archivio attendibili è una libreria di CA attendibili che contiene il browser o il sistema operativo. Per una PKI privata, l'IT dell'organizzazione deve assicurarsi che ogni browser o sistema abbia precedentemente aggiunto la CA root privata al relativo archivio attendibili. In caso contrario, il percorso di certificazione non può essere convalidato, causando errori client.

Il diagramma successivo mostra il percorso di convalida seguito da un browser quando viene presentato con un certificato X.509 dell'entità finale. Si noti che il certificato dell'entità finale manca dell'autorità di firma e serve solo per autenticare l'entità che lo possiede.



Il browser ispeziona il certificato dell'entità finale. Il browser rileva che il certificato offre una firma da CA subordinata (livello 3) come credenziale di attendibilità. I certificati per le CA subordinate devono essere inclusi nello stesso file PEM. In alternativa, possono anche trovarsi in un file separato che contiene i certificati che compongono la catena di attendibilità. Dopo aver trovato questi, il browser controlla il certificato di CA subordinata (livello 3) e rileva che questo offre una firma da CA subordinata (livello 2). A sua volta, la CA subordinata (livello 2) offre una firma dalla CA root (livello 1) come credenziale di attendibilità. Se il browser trova una copia del certificato emesso da una CA root privata preinstallato nel relativo archivio attendibilità, convalida il certificato dell'entità finale come attendibile.

In genere, il browser controlla anche ogni certificato rispetto a un elenco di revoche di certificati (CRL). Un certificato scaduto, revocato o configurato in modo errato viene rifiutato e la convalida non riesce.

Pianificazione della struttura di una gerarchia CA

In generale, la gerarchia CA dovrebbe riflettere la struttura dell'organizzazione. Cerca di stabilire una lunghezza del percorso (ovvero il numero di livelli di CA) non superiore a quella necessaria per delegare i ruoli amministrativi e di sicurezza. Aggiungere una CA alla gerarchia significa aumentare il numero di certificati nel percorso di certificazione, aumentando così il tempo di convalida. Mantenendo al minimo la lunghezza del percorso si riduce anche il numero di certificati inviati dal server al client durante la convalida di un certificato di entità finale.

In teoria, una CA principale, priva di pathLenConstraintparametri, può autorizzare livelli illimitati di CA subordinate. Una CA subordinata può avere tutte le CA subordinate secondarie consentite dalla sua configurazione interna. CA privata AWS le gerarchie gestite supportano percorsi di certificazione CA fino a cinque livelli di profondità.

Le strutture CA ben progettate hanno diversi vantaggi:

- Controlli amministrativi separati per diverse unità organizzative
- Possibilità di delegare l'accesso alle CA subordinate
- Struttura gerarchica che protegge le CA di livello superiore con controlli di sicurezza aggiuntivi

Due strutture CA comuni realizzano tutto questo:

- Due livelli CA: CA root e CA subordinata

  Si tratta della struttura CA più semplice che consente di separare i criteri di amministrazione, controllo e sicurezza per la CA root e una CA subordinata. È possibile mantenere controlli e criteri restrittivi per la CA principale, consentendo al contempo un accesso più permissivo per la CA subordinata. Quest'ultimo viene utilizzato per l'emissione in blocco di certificati di entità finale.

- Tre livelli CA: CA root e due livelli di CA subordinata

  Analogamente a quanto sopra, questa struttura aggiunge un ulteriore livello CA per separare ulteriormente la CA root dalle operazioni CA di basso livello. Il livello CA intermedio viene utilizzato solo per firmare CA subordinate che eseguono l'emissione di certificati di entità finale.

Le strutture CA meno comuni sono le seguenti:

- Quattro o più livelli CA

Sebbene meno comuni delle gerarchie a tre livelli, le gerarchie CA con quattro o più livelli sono possibili e potrebbero essere necessarie per consentire la delega amministrativa.

- Un livello CA: solo CA root

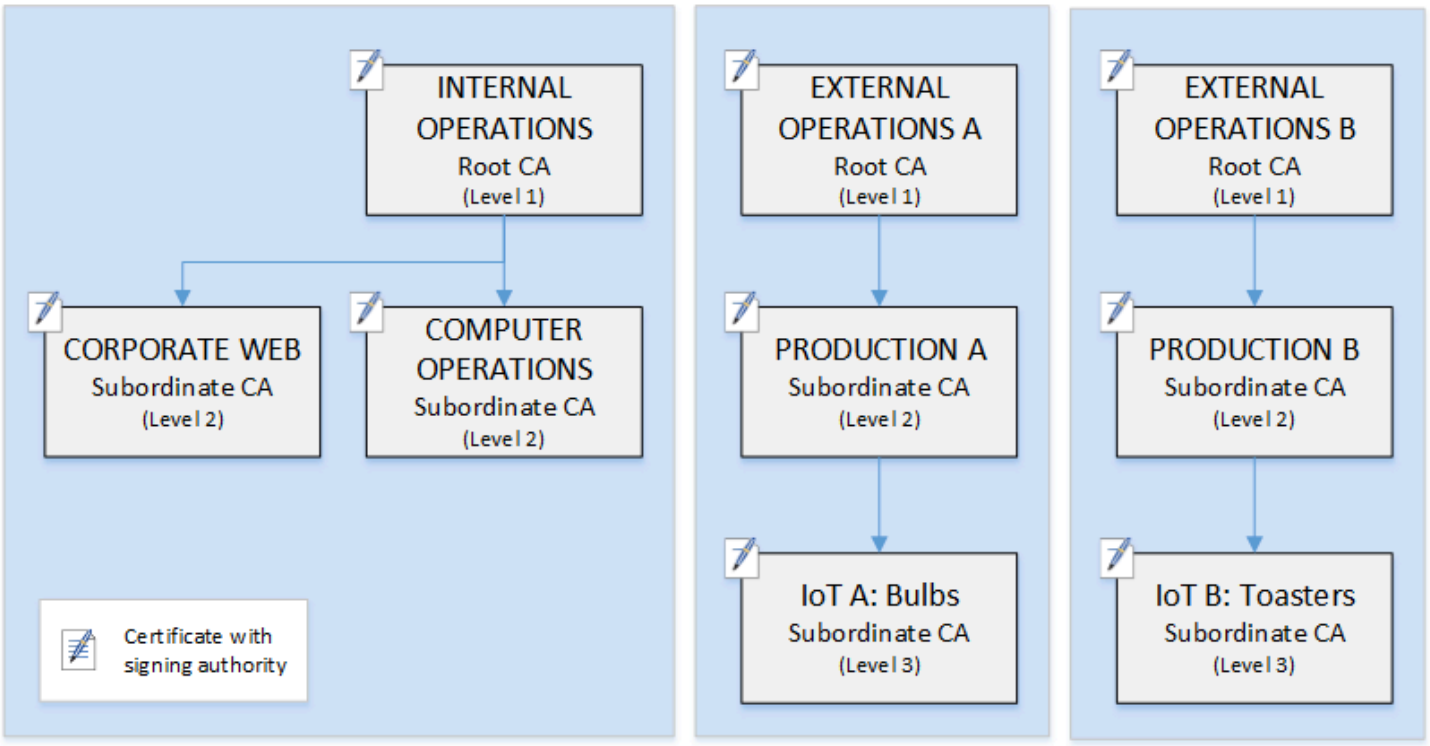
  Questa struttura è comunemente utilizzata per lo sviluppo e il test quando non è richiesta una catena completa di attendibilità. Utilizzata nella produzione, è atipica. Inoltre, viola la procedura consigliata per la gestione di criteri di protezione separati per la CA principale e le CA che rilasciano certificati di entità finale.

  Tuttavia, se state già emettendo certificati direttamente da una CA principale, potete migrare a. CA privata AWS In questo modo si ottengono vantaggi in termini di sicurezza e controllo rispetto all'utilizzo di una CA root gestita con OpenSSL o altro software.

Esempio di PKI privata per un produttore

In questo esempio, un'ipotetica azienda tecnologica produce due prodotti Internet delle cose (IoT), una lampadina intelligente e un tostapane intelligente. Durante la produzione, per ogni dispositivo viene rilasciato un certificato di entità finale in modo che possa comunicare in modo sicuro via Internet con il produttore. La PKI della società protegge anche la sua infrastruttura informatica, incluso il sito web interno e vari servizi informatici auto-ospitati che gestiscono operazioni finanziarie e commerciali.

Di conseguenza, la gerarchia CA modella da vicino questi aspetti amministrativi e operativi del business.



Questa gerarchia contiene tre radici, una per le operazioni interne e due per le operazioni esterne (una CA root per ogni linea di prodotto). Illustra inoltre la lunghezza del percorso di certificazione multiplo, con due livelli di CA per le operazioni interne e tre livelli per le operazioni esterne.

L'uso di CA root separate e livelli CA subordinati aggiuntivi sul lato delle operazioni esterne è una decisione progettuale che risponde alle esigenze aziendali e di sicurezza. Con più alberi di CA, la PKI è a prova di futuro contro le riorganizzazioni aziendali, le cessioni o le acquisizioni. Quando si verificano modifiche, un'intera gerarchia CA root può spostarsi in modo pulito con la divisione che protegge. E con due livelli di CA subordinata, le CA principali hanno un alto livello di isolamento rispetto alle CA di livello 3 responsabili della firma in blocco dei certificati per migliaia o milioni di articoli prodotti.

Sul lato interno, le operazioni aziendali sul Web e sul computer interno completano una gerarchia a due livelli. Questi livelli consentono agli amministratori Web e ai tecnici operativi di gestire l'emissione di certificati in modo indipendente per i propri domini di lavoro. La compartimentazione di PKI in domini funzionali distinti è una best practice di sicurezza e protegge ciascuno da una compromissione che potrebbe influire sull'altro. Gli amministratori Web rilasciano certificati di entità finale da utilizzare dai browser Web in tutta l'azienda, autenticando e crittografando le comunicazioni sul sito web interno. I tecnici operativi emettono certificati delle entità finali che autenticano gli host del data center e i servizi informatici reciprocamente. Questo sistema aiuta a proteggere i dati sensibili crittografandoli sulla LAN.

Impostazione dei vincoli di lunghezza sul percorso di certificazione

La struttura di una gerarchia di CA è definita e applicata dall'estensione dei vincoli di base contenuta in ogni certificato. L'estensione definisce due vincoli:

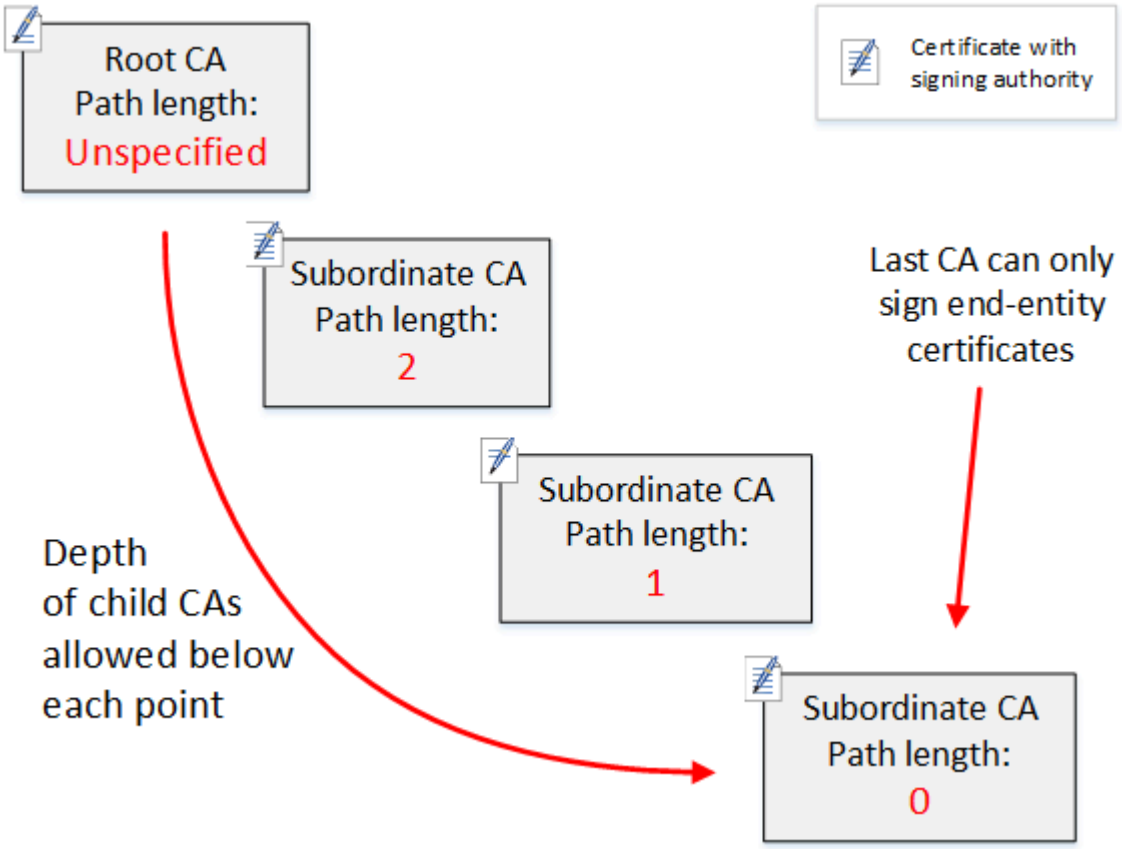
- `cA`— Se il certificato definisce una CA. Se questo valore è false (impostazione predefinita), il certificato è un certificato di entità finale.
- `pathLenConstraint`— Il numero massimo di CA subordinate di livello inferiore che possono esistere in una catena di fiducia valida. Il certificato dell'entità finale non viene conteggiato perché non è un certificato CA.

Un certificato emesso da una CA root richiede la massima flessibilità e non include un vincolo di lunghezza del percorso. Ciò consente alla radice di definire un percorso di certificazione di qualsiasi lunghezza.

> **Note**
>
> CA privata AWS limita il percorso di certificazione a cinque livelli.

Le CA subordinate hanno valori `pathLenConstraint` uguali o maggiori di zero, a seconda della posizione nella gerarchia e delle funzionalità desiderate. Ad esempio, in una gerarchia con tre CA, non viene specificato alcun vincolo di percorso per la CA root. La prima CA subordinata ha una lunghezza di percorso pari a 1 e può quindi firmare CA figli. Ognuna di queste CA figlio deve necessariamente avere un valore `pathLenConstraint` pari a zero. Ciò significa che possono firmare certificati di entità finale ma non possono emettere certificati emessi da una CA aggiuntivi. Limitare la potenza necessaria per creare nuove CA è un importante controllo di sicurezza.

Il diagramma seguente illustra questa propagazione di autorità limitata lungo la gerarchia.



In questa gerarchia a quattro livelli, la root non è vincolata (come sempre). Ma la prima CA subordinata ha un valore `pathLenConstraint` di 2, il che limita le CA figlio ad andare più di due livelli più in profondità. Di conseguenza, per un percorso di certificazione valido, il valore del vincolo deve diminuire a zero nei due livelli successivi. Se un browser Web rileva un certificato di entità finale di questo ramo con una lunghezza di percorso maggiore di quattro, la convalida non riesce. Tale certificato potrebbe essere il risultato di una CA creata accidentalmente, di una CA configurata in modo errato o di un'emissione non autorizzata.

Gestione della lunghezza del percorso con modelli

CA privata AWS fornisce modelli per l'emissione di certificati root, subordinati e di entità finale. Questi modelli incapsulano le best practice per i valori dei vincoli di base, inclusa la lunghezza del percorso. I modelli includono:

- RootCACertificate/V1
- SubordinateCAertificate_ 0/V1 PathLen
- Certificato CA subordinato_ 1/V1 PathLen
- Certificato CA subordinato_ 2/V1 PathLen

- Certificato CA subordinato_ 3/V1 PathLen
- EndEntityCertificate/V1

L'API `IssueCertificate` restituirà un errore se si tenta di creare una CA con una lunghezza di percorso maggiore o uguale alla lunghezza del percorso del certificato emesso da una CA emittente.

Per ulteriori informazioni sui modelli di certificato, consulta Informazioni sui modelli di certificato.

Automatizzazione della configurazione della gerarchia CA con AWS CloudFormation

Una volta definito un progetto per la gerarchia delle CA, è possibile testarlo e metterlo in produzione utilizzando un modello. AWS CloudFormation Per un esempio di tale modello, vedere Dichiarazione di una gerarchia CA privata nella Guida per l'utente AWS CloudFormation .

Gestione del ciclo di vita della CA privata

I certificati emessi da una CA hanno una durata fissa o un periodo di validità. Quando un certificato emesso da una CA scade, tutti i certificati emessi direttamente o indirettamente dalle CA subordinate sottostanti nella gerarchia CA diventano non validi. È possibile evitare la scadenza del certificato emesso da una CA pianificando in anticipo.

Scelta dei periodi di validità

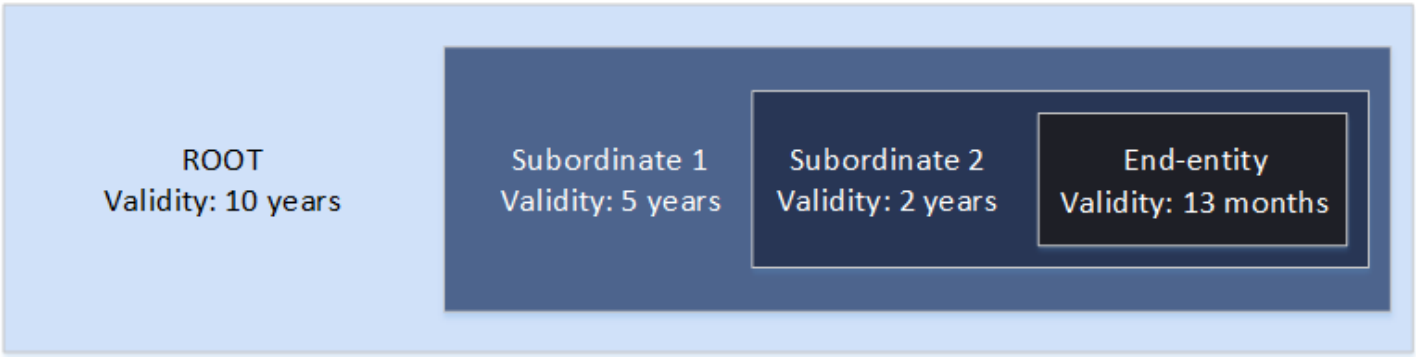
Il periodo di validità di un certificato X.509 è un campo certificato di base obbligatorio. Determina l'intervallo di tempo durante il quale l'autorità di certificazione emittente certifica che il certificato può essere considerato attendibile, a meno che non lo revochi. (Un certificato root, autofirmato, certifica il proprio periodo di validità.)

CA privata AWS e AWS Certificate Manager assiste nella configurazione dei periodi di validità dei certificati soggetti ai seguenti vincoli:

- Un certificato gestito da CA privata AWS deve avere un periodo di validità inferiore o uguale al periodo di validità della CA che lo ha emesso. In altre parole, le CA figlio e i certificati dell'entità finale non possono sopravvivere ai certificati padre. Il tentativo di utilizzare l'API `IssueCertificate` per emettere un certificato emesso da una CA con un periodo di validità maggiore o uguale alla CA padre non riesce.
- I certificati emessi e gestiti da AWS Certificate Manager (quelli per i quali ACM genera la chiave privata) hanno un periodo di validità di 13 mesi (395 giorni). ACM gestisce il processo di rinnovo di

questi certificati. Se utilizzi l'emissione diretta CA privata AWS di certificati, puoi scegliere qualsiasi periodo di validità.

Il diagramma seguente mostra una configurazione tipica dei periodi di validità nidificati. Il certificato principale è il più longevo; i certificati di entità finale sono relativamente brevi e le CA subordinate variano tra questi estremi.



Quando si pianifica la gerarchia della CA, determinare la durata ottimale per i certificati emessi da una CA. Lavorare a ritroso dalla durata desiderata dei certificati di entità finale che si desidera emettere.

Certificati entità finali

I certificati dell'entità finale devono avere un periodo di validità adeguato al caso d'uso. Una breve durata riduce al minimo l'esposizione di un certificato nel caso in cui la sua chiave privata venga persa o rubata. Tuttavia, brevi periodi di vita significano frequenti rinnovi. Il mancato rinnovo di un certificato in scadenza può causare tempi di inattività.

L'uso distribuito di certificati di entità finale può anche presentare problemi logistici in caso di violazione della sicurezza. La pianificazione deve tenere conto dei certificati di rinnovo e distribuzione, della revoca di certificati compromessi e della rapidità di propagazione delle revoche ai client che si basano sui certificati.

Il periodo di validità predefinito per un certificato di entità finale emesso tramite ACM è di 13 mesi (395 giorni). In CA privata AWS, puoi utilizzare l'`IssueCertificate`API per applicare qualsiasi periodo di validità purché inferiore a quello della CA emittente.

Certificati emessi da una CA subordinata

I certificati emessi da una CA subordinata devono avere periodi di validità significativamente più lunghi rispetto ai certificati emessi. Un intervallo valido per la validità di un certificato emesso da una

CA è da due a cinque volte il periodo di qualsiasi certificato emesso da una CA figlio o il certificato dell'entità finale emesso. Si supponga, ad esempio, di disporre di una gerarchia CA a due livelli (CA root e CA subordinata). Se si desidera emettere certificati di entità finale con una durata di un anno, è possibile configurare la durata della CA di emissione subordinata su tre anni. Questo è il periodo di validità predefinito per un certificato CA subordinato in. CA privata AWS I certificati emessi da una CA subordinata possono essere modificati senza sostituire il certificato emesso da una CA root.

Certificati emessi da una CA root

Le modifiche apportate a un certificato emesso da una CA root influiscono sull'intera infrastruttura PKI (infrastruttura a chiave pubblica) e richiedono l'aggiornamento di tutti gli archivi attendibili client dipendenti e del browser. Per ridurre al minimo l'impatto operativo, è necessario scegliere un periodo di validità lungo per il certificato principale. L' CA privata AWS impostazione predefinita per i certificati root è dieci anni.

Gestione della successione CA

È possibile gestire la successione CA in due modi: sostituire la CA precedente o riemettere la CA con un nuovo periodo di validità.

Sostituzione di una vecchia CA

Per sostituire una CA precedente, creare una nuova CA e concatenarla alla stessa CA padre. Successivamente, si emettono certificati dalla nuova CA.

I certificati emessi dalla nuova CA hanno una nuova catena di CA. Una volta stabilita la nuova CA, è possibile disabilitare la vecchia CA per impedire l'emissione di nuovi certificati. Sebbene disattivata, la vecchia CA supporta la revoca dei vecchi certificati emessi dalla CA e, se configurata in tal senso, continua a convalidare i certificati tramite OCSP e/o elenchi di revoca dei certificati (CRL). Quando scade l'ultimo certificato rilasciato dalla vecchia CA, è possibile eliminare la vecchia CA. È possibile generare un report di audit per tutti i certificati emessi dalla CA per confermare che tutti i certificati emessi sono scaduti. Se la vecchia CA dispone di CA subordinate, è necessario sostituirle anche perché le CA subordinate scadono contemporaneamente o prima della relativa CA padre. Iniziare sostituendo la CA più alta nella gerarchia che deve essere sostituita. Quindi creare nuove CA subordinate sostitutive ad ogni livello inferiore successivo.

AWS consiglia di includere un identificatore di generazione CA nei nomi delle CA, se necessario. Ad esempio, si supponga di denominare la CA di prima generazione «Corporate Root CA». Quando crei la CA di seconda generazione, chiamala «Corporate Root CA G2". Questa semplice convenzione di denominazione può aiutare a evitare confusione quando entrambe le CA non sono scadute.

Questo metodo di successione CA è preferito perché ruota la chiave privata della CA. La rotazione della chiave privata è una procedura consigliata per le chiavi CA. La frequenza di rotazione dovrebbe essere proporzionale alla frequenza di utilizzo della chiave: le CA che emettono più certificati dovrebbero essere ruotate più frequentemente.

> **Note**
>
> I certificati privati emessi tramite ACM non possono essere rinnovati se si sostituisce la CA. Se si utilizza ACM per l'emissione e il rinnovo, è necessario emettere nuovamente il certificato CA per prolungarne la durata.

Riemissione di una vecchia CA

Quando una CA si avvicina alla scadenza, un metodo alternativo per prolungarne la durata consiste nel riemettere il certificato CA con una nuova data di scadenza. La riemissione lascia invariati tutti i metadati CA e preserva le chiavi private e pubbliche esistenti. In questo scenario, la catena di certificati esistente e i certificati di entità finale non scaduti emessi dalla CA rimangono validi fino alla scadenza. L'emissione di nuovi certificati può inoltre continuare senza interruzioni. Per aggiornare una CA con un certificato riemesso, segui le normali procedure di installazione descritte in. Creazione e installazione del certificato CA

> **Note**
>
> Consigliamo di sostituire una CA in scadenza anziché riemetterne il certificato, per via dei vantaggi in termini di sicurezza derivanti dal passaggio a una nuova key pair.

Revoca di una CA

Si revoca una CA revocando il relativo certificato sottostante. Ciò inoltre revoca di fatto tutti i certificati emessi dalla CA. Le informazioni sulla revoca vengono distribuite ai clienti tramite OCSP o CRL. È necessario revocare un certificato CA solo se si desidera revocare tutti i certificati CA di entità finale e subordinati emessi.

Impostazione di un metodo di revoca dei certificati

Quando pianifichi la tua PKI privata CA privata AWS, dovresti considerare come gestire le situazioni in cui non desideri più che gli endpoint considerino attendibile un certificato emesso, ad esempio quando la chiave privata di un endpoint è esposta. Gli approcci più comuni a questo problema consistono nell'utilizzare certificati di breve durata o nel configurare la revoca dei certificati. I certificati di breve durata scadono in un periodo di tempo così breve, in ore o giorni, che la revoca non ha senso, poiché il certificato diventa non valido all'incirca nello stesso tempo necessario per notificare un endpoint di revoca. Questa sezione descrive le opzioni di revoca per i CA privata AWS clienti, inclusa la configurazione e le migliori pratiche.

I clienti che cercano un metodo di revoca possono scegliere l'Online Certificate Status Protocol (OCSP), gli elenchi di revoca dei certificati (CRL) o entrambi.

> **Note**
>
> Se crei la tua CA senza configurare la revoca, puoi sempre configurarla in un secondo momento. Per ulteriori informazioni, consulta Aggiornamento della CA privata.

- Online Certificate Status Protocol (OCSP)

  CA privata AWS fornisce una soluzione OCSP completamente gestita per notificare agli endpoint che i certificati sono stati revocati senza che i clienti debbano gestire autonomamente l'infrastruttura. I clienti possono abilitare OCSP su CA nuove o esistenti con un'unica operazione utilizzando la CA privata AWS console, l'API, la CLI o tramite. AWS CloudFormation Mentre i CRL vengono archiviati ed elaborati sull'endpoint e possono diventare obsoleti, i requisiti di archiviazione ed elaborazione OCSP vengono gestiti in modo sincrono sul backend del risponditore.

  Quando abiliti OCSP per una CA, CA privata AWS include l'URL del risponditore OCSP nell'estensione Authority Information Access (AIA) di ogni nuovo certificato emesso. L'estensione consente a client come i browser Web di interrogare il risponditore e determinare se è attendibile un certificato CA subordinato o di entità finale. Il risponditore restituisce un messaggio di stato firmato crittograficamente per garantirne l'autenticità.

  Il risponditore CA privata AWS OCSP è conforme alla RFC 5019.

  Considerazioni OCSP

  - I messaggi di stato OCSP vengono firmati utilizzando lo stesso algoritmo di firma per cui è stata configurata la CA emittente. Le CA create nella CA privata AWS console utilizzano l'algoritmo di firma SHA256WITHRSA per impostazione predefinita. Altri algoritmi supportati sono disponibili nella documentazione dell'API. CertificateAuthorityConfiguration
  - I modelli di certificato APIPassThrough e CSRPassThrough non funzioneranno con l'estensione AIA se il risponditore OCSP è abilitato.
  - L'endpoint del servizio OCSP gestito è accessibile sulla rete Internet pubblica. I clienti che desiderano OCSP ma preferiscono non avere un endpoint pubblico dovranno gestire la propria infrastruttura OCSP.
- Elenchi di revoca dei certificati (CRL)

  Un CRL contiene un elenco di certificati revocati. Quando si configura una CA per generare CRL, CA privata AWS include l'estensione CRL Distribution Points in ogni nuovo certificato emesso. Questa estensione fornisce l'URL per il CRL. L'estensione consente a client come i browser Web di interrogare il CRL e determinare se un certificato CA subordinato o di entità finale può essere considerato attendibile.

Poiché un client deve scaricare i CRL ed elaborarli localmente, il loro utilizzo richiede più memoria rispetto a OCSP. I CRL possono consumare meno larghezza di banda di rete perché l'elenco dei CRL viene scaricato e memorizzato nella cache, rispetto a OCSP che controlla lo stato di revoca per ogni nuovo tentativo di connessione.

> **Note**
>
> Sia gli OCSP che i CRL presentano un certo ritardo tra la revoca e la disponibilità della modifica dello stato.
>
> - Le risposte OCSP possono richiedere fino a 60 minuti per riflettere il nuovo stato quando si revoca un certificato. In generale, OCSP tende a supportare una distribuzione più rapida delle informazioni di revoca perché, a differenza dei CRL che possono essere memorizzati nella cache dai client per giorni, le risposte OCSP in genere non vengono memorizzate nella cache dai client.
> - Generalmente un CRL viene aggiornato circa 30 minuti dopo che un certificato viene revocato. Se per qualsiasi motivo un aggiornamento del CRL fallisce, effettua ulteriori tentativi ogni 15 minuti. CA privata AWS

Requisiti generali per le configurazioni di revoca

I seguenti requisiti si applicano a tutte le configurazioni di revoca.

- Una configurazione che disabilita i CRL o l'OCSP deve contenere solo il parametro `Enabled=False` e non riesce se vengono inclusi altri parametri, ad esempio `CustomCname` o `ExpirationInDays`.
- In una configurazione CRL, il `S3BucketName` parametro deve essere conforme alle regole di denominazione dei bucket di Amazon Simple Storage Service.
- Una configurazione contenente un parametro Canonical Name (CNAME) personalizzato per CRL o OCSP deve essere conforme alle restrizioni RFC7230 sull'uso di caratteri speciali in un CNAME.
- In una configurazione CRL o OCSP, Il valore di un parametro CNAME non deve includere un prefisso di protocollo come “http://” o “https://”.

Argomenti

- Pianificazione di un elenco di revoca dei certificati (CRL)
- Configurazione di un URL personalizzato per OCSP CA privata AWS

Pianificazione di un elenco di revoca dei certificati (CRL)

Prima di poter configurare un CRL come parte del processo di creazione della CA, potrebbe essere necessaria una configurazione preliminare. Questa sezione spiega i prerequisiti e le opzioni da comprendere prima di creare una CA con un CRL allegato.

Per informazioni sull'utilizzo dell'Online Certificate Status Protocol (OCSP) come alternativa o supplemento a un CRL, vedere e. Opzioni di revoca del certificato Configurazione di un URL personalizzato per OCSP CA privata AWS

Argomenti

- Struttura CRL
- Politiche di accesso per i CRL in Amazon S3
- Abilitazione di S3 Block Public Access (BPA) con CloudFront
- Crittografia dei CRL
- Determinazione dell'URI del punto di distribuzione CRL (CDP)

Struttura CRL

Ogni CRL è un file con codifica DER. Per scaricare il file e utilizzare OpenSSL per visualizzarlo, usa un comando simile al seguente:

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

I CRL hanno il seguente formato:

```
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**
>
> Il CRL verrà depositato in Amazon S3 solo dopo l'emissione di un certificato che lo riguarda. In precedenza, nel bucket Amazon S3 sarà visibile solo un `acm-pca-permission-test-key` file.

Politiche di accesso per i CRL in Amazon S3

Se intendi creare un CRL, devi preparare un bucket Amazon S3 in cui archiviarlo. CA privata AWS deposita automaticamente il CRL nel bucket Amazon S3 designato e lo aggiorna periodicamente. Per ulteriori informazioni, consulta Creazione di un bucket.

Il tuo bucket S3 deve essere protetto da una politica di autorizzazioni IAM allegata. Gli utenti e i responsabili del servizio autorizzati richiedono `Put` l'autorizzazione per consentire di CA privata AWS inserire oggetti nel bucket e l'autorizzazione per recuperarli. `Get` Durante la procedura della console per la creazione di una CA, puoi scegliere di consentire la CA privata AWS creazione di un nuovo bucket e applicare una politica di autorizzazioni predefinita.

> **Note**
>
> La configurazione della policy IAM dipende dal soggetto coinvolto. Regioni AWS Le regioni si dividono in due categorie:
>
> - Regioni abilitate per impostazione predefinita: regioni abilitate per impostazione predefinita per tutti. Account AWS
> - Regioni disabilitate per impostazione predefinita: aree disattivate per impostazione predefinita, ma che possono essere abilitate manualmente dal cliente.

Per ulteriori informazioni e un elenco delle regioni disabilitate per impostazione predefinita, consulta Gestione. Regioni AWS Per una discussione sui principali servizi nel contesto di IAM, consulta AWS Service Principles in opt-in Regions.
Quando configuri i CRL come metodo di revoca dei certificati, CA privata AWS crea un CRL e lo pubblica in un bucket S3. Il bucket S3 richiede una policy IAM che consenta al responsabile del servizio di scrivere nel CA privata AWS bucket. Il nome del service principal varia in base alle regioni utilizzate e non tutte le possibilità sono supportate.

PCA	S3	Principale del servizio
Entrambi nella stessa regione		`acm-pca.amazonaws.com`
Abilitato	Abilitato	`acm-pca.amazonaws.com`
Disabilitato	Abilitato	`acm-pca.Region.amazonaws.com`
Abilitato	Disabilitato	Non supportato

La politica predefinita non applica alcuna `SourceArn` restrizione alla CA. Ti consigliamo di applicare manualmente la politica meno permissiva mostrata di seguito, che limita l'accesso sia a un AWS account specifico che a una CA privata specifica. Per ulteriori informazioni, consulta Aggiungere una policy sui bucket utilizzando la console Amazon S3.

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

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

Se scegli di consentire la politica predefinita, puoi sempre modificarla in un secondo momento.

Abilitazione di S3 Block Public Access (BPA) con CloudFront

I nuovi bucket Amazon S3 sono configurati per impostazione predefinita con la funzionalità Block Public Access (BPA) attivata. Incluso nelle best practice di sicurezza di Amazon S3, BPA è un set di controlli di accesso che i clienti possono utilizzare per ottimizzare l'accesso agli oggetti nei loro bucket S3 e ai bucket nel loro insieme. Quando il BPA è attivo e configurato correttamente, solo AWS gli utenti autorizzati e autenticati hanno accesso a un bucket e al suo contenuto.

AWS consiglia l'uso del BPA su tutti i bucket S3 per evitare l'esposizione di informazioni sensibili a potenziali avversari. Tuttavia, è necessaria una pianificazione aggiuntiva se i client PKI recuperano i CRL sulla rete Internet pubblica (ovvero senza aver effettuato l'accesso a un account). AWS Questa sezione descrive come configurare una soluzione PKI privata utilizzando Amazon CloudFront, una rete di distribuzione dei contenuti (CDN), per servire i CRL senza richiedere l'accesso client autenticato a un bucket S3.

> **Note**
>
> L'utilizzo CloudFront comporta costi aggiuntivi per il tuo account. AWS Per ulteriori informazioni, consulta la pagina CloudFront dei prezzi di Amazon.

Se scegli di archiviare il tuo CRL in un bucket S3 con BPA abilitato e non lo usi CloudFront, devi creare un'altra soluzione CDN per garantire che il tuo client PKI abbia accesso al tuo CRL.

Configura Amazon S3 con BPA

In S3, crea un nuovo bucket per il tuo CRL, come al solito, quindi abilita BPA su di esso.

Per configurare un bucket Amazon S3 che blocchi l'accesso pubblico al tuo CRL

1. Crea un nuovo bucket S3 utilizzando la procedura descritta in Creazione di un bucket. Durante la procedura, seleziona l'opzione Blocca tutti gli accessi pubblici.

   Per ulteriori informazioni, consulta Bloccare l'accesso pubblico allo storage Amazon S3.
2. Una volta creato il bucket, scegli il nome dall'elenco, vai alla scheda Autorizzazioni, scegli Modifica nella sezione Proprietà dell'oggetto e seleziona Preferito proprietario del bucket.
3. Inoltre, nella scheda Autorizzazioni, aggiungi una policy IAM al bucket come descritto in. Politiche di accesso per i CRL in Amazon S3

Configurazione per BPA CloudFront

Crea una CloudFront distribuzione che abbia accesso al tuo bucket S3 privato e possa fornire CRL a client non autenticati.

Per configurare una distribuzione per il CRL CloudFront

1. Crea una nuova CloudFront distribuzione utilizzando la procedura descritta in Creazione di una distribuzione nell'Amazon CloudFront Developer Guide.

   Durante il completamento della procedura, applica le seguenti impostazioni:

   - In Origin Domain Name, scegli il tuo bucket S3.
   - Scegli Sì per limitare l'accesso ai bucket.
   - Scegli Crea una nuova identità per Origin Access Identity.
   - Scegli Sì, aggiorna la politica del bucket in Concedi autorizzazioni di lettura su Bucket.

> **Note**
>
> In questa procedura, CloudFront modifica la policy del bucket per consentirle di accedere agli oggetti bucket. Valuta la possibilità di modificare questa politica per consentire l'accesso solo agli oggetti all'interno della cartella. `crl`

2. Dopo l'inizializzazione della distribuzione, individua il relativo nome di dominio nella CloudFront console e salvalo per la procedura successiva.

> **Note**
>
> Se il tuo bucket S3 è stato appena creato in una regione diversa da us-east-1, potresti ricevere un errore di reindirizzamento temporaneo HTTP 307 quando accedi all'applicazione pubblicata tramite. CloudFront Potrebbero essere necessarie diverse ore prima che l'indirizzo del bucket si propaghi.

Configura la tua CA per BPA

Durante la configurazione della nuova CA, includi l'alias nella distribuzione. CloudFront

Per configurare la tua CA con un CNAME per CloudFront

- Crea la tua CA utilizzandoProcedura per la creazione di una CA (CLI) .

  Quando si esegue la procedura, il file di revoca `revoke_config.txt` deve includere le seguenti righe per specificare un oggetto CRL non pubblico e fornire un URL all'endpoint di distribuzione in: CloudFront

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

  In seguito, quando si emettono certificati con questa CA, questi conterranno un blocco come il seguente:

```
X509v3 CRL Distribution Points:
 Full Name:
```

```
 URI:http://abcdef012345.cloudfront.net/crl/01234567-89ab-
cdef-0123-456789abcdef.crl
```

> **Note**
>
> Se disponi di certificati precedenti emessi da questa CA, non saranno in grado di accedere al CRL.

Crittografia dei CRL

Facoltativamente, puoi configurare la crittografia sul bucket Amazon S3 contenente i tuoi CRL. CA privata AWS supporta due modalità di crittografia per gli asset in Amazon S3:

- Crittografia automatica lato server con chiavi AES-256 gestite da Amazon S3.
- Crittografia gestita dal cliente utilizzando AWS Key Management Service e configurata secondo le tue specifiche. AWS KMS key

> **Note**
>
> CA privata AWS non supporta l'utilizzo di chiavi KMS predefinite generate automaticamente da S3.

Nelle procedure seguenti viene descritto come impostare ciascuna delle opzioni di crittografia.

Per configurare la crittografia automatica

Completare la procedura seguente per abilitare la crittografia lato server S3.

1. Apri la console Amazon S3 all'indirizzo https://console.aws.amazon.com/s3/.
2. Nella tabella Bucket, scegli il bucket che conterrà i tuoi asset. CA privata AWS
3. Nella pagina relativa al bucket scegliere la scheda Proprietà .
4. Scegliere la scheda di crittografia predefinita.
5. Scegli Abilita .
6. Scegli la chiave Amazon S3 (SSE-S3).

7. Seleziona Salva modifiche.

Per configurare la crittografia personalizzata

Completa i seguenti passaggi per abilitare la crittografia utilizzando una chiave personalizzata.

1. Apri la console Amazon S3 all'indirizzo https://console.aws.amazon.com/s3/.
2. Nella tabella Bucket, scegli il bucket che conterrà i tuoi CA privata AWS asset.
3. Nella pagina relativa al bucket scegliere la scheda Proprietà .
4. Scegliere la scheda di crittografia predefinita.
5. Scegli Abilita .
6. Scegli la AWS Key Management Service chiave (SSE-KMS).
7. Scegli tra AWS KMS le tue chiavi o Inserisci AWS KMS key ARN.
8. Seleziona Salva modifiche.
9. (Facoltativo) Se non disponi già di una chiave KMS, creane una utilizzando il seguente comando AWS CLI create-key:

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

   L'output contiene l'ID della chiave e l'Amazon Resource Name (ARN) della chiave KMS. Di seguito è riportato un esempio di 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. Utilizzando i passaggi seguenti, si concede al CA privata AWS servizio l'autorizzazione principale a utilizzare la chiave KMS. Per impostazione predefinita, tutte le chiavi KMS sono private; solo il

proprietario della risorsa può utilizzare una chiave KMS per crittografare e decrittografare i dati. Tuttavia, il proprietario della risorsa può concedere ad altri utenti e risorse le autorizzazioni per accedere alla chiave KMS. Il responsabile del servizio deve trovarsi nella stessa regione in cui è archiviata la chiave KMS.

a. Innanzitutto, salva la politica predefinita per la tua chiave KMS `policy.json` utilizzando il seguente comando: get-key-policy

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

b. Apri il file `policy.json` in un editor di testo. Seleziona una delle seguenti dichiarazioni politiche e aggiungila alla politica esistente.

Se la tua chiave bucket Amazon S3 è abilitata, usa la seguente dichiarazione:

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

Se la tua chiave bucket Amazon S3 è disabilitata, usa la seguente dichiarazione:

```
{
   "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. Infine, applica la policy aggiornata utilizzando il seguente comando: put-key-policy

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

Determinazione dell'URI del punto di distribuzione CRL (CDP)

Se utilizzi il bucket S3 come CDP per la tua CA, l'URI CDP può avere uno dei seguenti formati.

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

Se la CA è stata configurata con un CNAME personalizzato, l'URI CDP includerà il CNAME, ad esempio `http://alternative.example.com/crl/CA-ID.crl`

Configurazione di un URL personalizzato per OCSP CA privata AWS

> **Note**
>
> Questo argomento è destinato ai clienti che desiderano personalizzare l'URL pubblico dell'endpoint di risposta OCSP per scopi di branding o altri scopi. Se prevedi di utilizzare la

configurazione predefinita dell'OCSP CA privata AWS gestito, puoi saltare questo argomento e seguire le istruzioni di configurazione in Configurare la revoca.

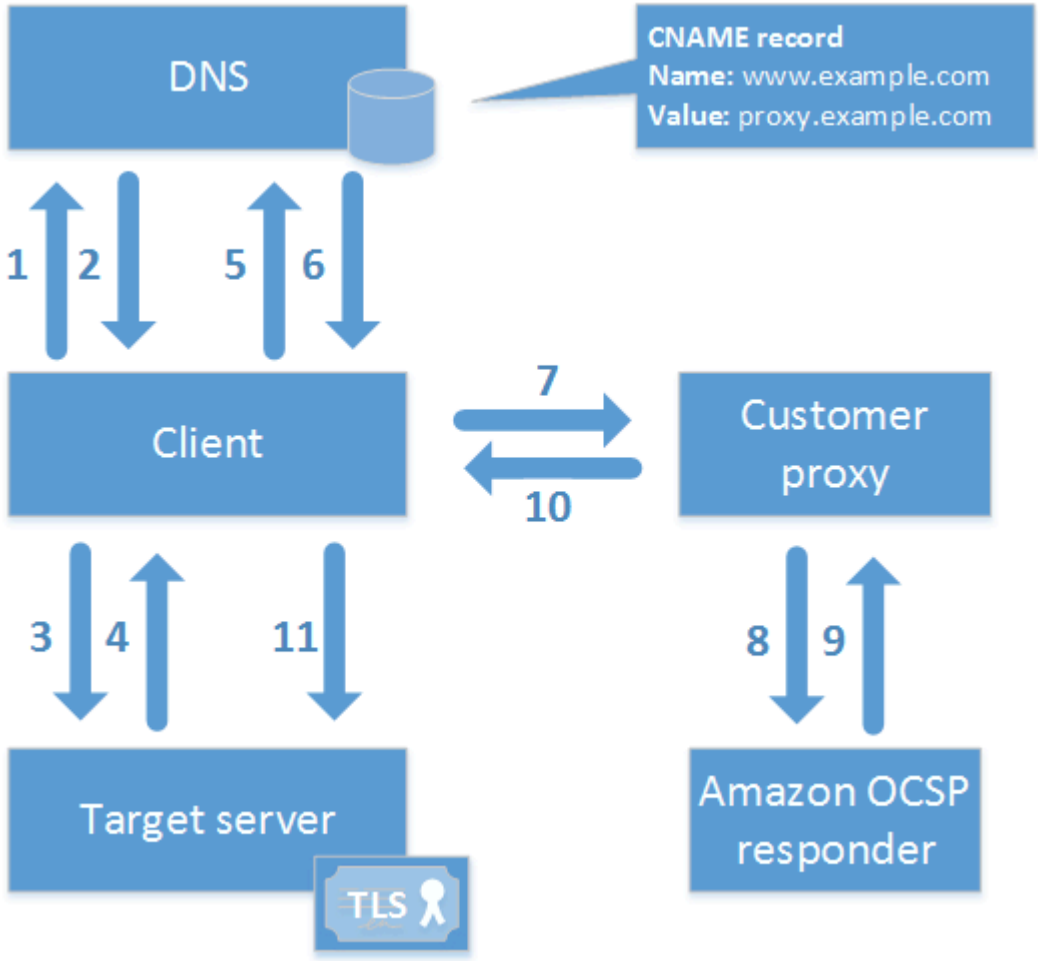
Per impostazione predefinita, quando abiliti OCSP per CA privata AWS, ogni certificato emesso contiene l'URL del risponditore OCSP. AWS Ciò consente ai client che richiedono una connessione crittograficamente sicura di inviare direttamente le query di convalida OCSP a. AWS Tuttavia, in alcuni casi potrebbe essere preferibile indicare un URL diverso nei certificati e comunque inviare le query OCSP a. AWS

> **Note**
>
> Per informazioni sull'utilizzo di un elenco di revoca dei certificati (CRL) come alternativa o supplemento a OCSP, vedere Configurazione della revoca e Pianificazione di un elenco di revoca dei certificati (CRL).

Nella configurazione di un URL personalizzato per OCSP sono coinvolti tre elementi.

- Configurazione CA: specifica un URL OCSP personalizzato `RevocationConfiguration` per la tua CA, come descritto in. Esempio 2: creare una CA con OCSP e un CNAME personalizzato abilitato Procedura per la creazione di una CA (CLI)
- DNS: aggiungi un record CNAME alla configurazione del tuo dominio per mappare l'URL che appare nei certificati a un URL del server proxy. Per ulteriori informazioni, consulta Esempio 2: creare una CA con OCSP e un CNAME personalizzato abilitato in Procedura per la creazione di una CA (CLI) .
- Server proxy di inoltro: configura un server proxy in grado di inoltrare in modo trasparente il traffico OCSP ricevuto al risponditore OCSP. AWS

Il diagramma seguente illustra come questi elementi interagiscono.



Come illustrato nel diagramma, il processo di convalida OCSP personalizzato prevede i seguenti passaggi:

1. Il client interroga il DNS per il dominio di destinazione.
2. Il client riceve l'IP di destinazione.
3. Il client apre una connessione TCP con target.
4. Il client riceve il certificato TLS di destinazione.
5. Il client richiede il DNS per il dominio OCSP elencato nel certificato.
6. Il client riceve l'IP proxy.
7. Il client invia una query OCSP al proxy.
8. Il proxy inoltra la richiesta al risponditore OCSP.
9. Il risponditore restituisce lo stato del certificato al proxy.
10. Il proxy inoltra lo stato del certificato al client.

11. Se il certificato è valido, il client avvia l'handshake TLS.

> **Tip**
>
> Questo esempio può essere implementato utilizzando Amazon CloudFront e Amazon Route 53 dopo aver configurato una CA come descritto sopra.
>
> 1. In CloudFront, crea una distribuzione e configurala come segue:
>   - Crea un nome alternativo che corrisponda al tuo CNAME personalizzato.
>   - Associa il tuo certificato ad esso.
>   - Imposta ocsp.acm-pca. <region>.amazonaws.com come origine.
>   - Applica la politica. `Managed-CachingDisabled`
>   - Imposta la politica del protocollo Viewer su HTTP e HTTPS.
>   - Imposta i metodi HTTP consentiti su GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE.
> 2. In Route 53, crea un record DNS che mappa il tuo CNAME personalizzato all'URL della CloudFront distribuzione.

Modalità di autorità di certificazione

CA privata AWS supporta la creazione di una CA in una delle due modalità. Le modalità GENERAL_PURPOSE e SHORT_LIVED_CERTIFICATE influiscono sul periodo di validità consentito dei certificati emessi dalla CA.

> **Note**
>
> CA privata AWS non esegue controlli di validità sui certificati CA root.

GENERAL_PURPOSE (impostazione predefinita)

Questa modalità consente alla CA di emettere certificati con qualsiasi periodo di validità. La maggior parte delle applicazioni utilizza certificati di questo tipo. In genere, la CA specifica anche un meccanismo di revoca.

SHORT_LIVED_CERTIFICATE

Questa modalità definisce una CA che emette esclusivamente certificati con un periodo di validità massimo di sette giorni. Questi certificati di breve durata scadono così rapidamente da poter essere implementati senza che sia in atto un meccanismo di revoca. Per alcune applicazioni, è più sensato implementare spesso certificati di breve durata piuttosto che incorrere nel sovraccarico di rete e di elaborazione della revoca.

Le CA con modalità SHORT_LIVED_CERTIFICATE costano meno delle CA per uso generico. Per ulteriori informazioni, consulta AWS Private Certificate Authority la sezione Prezzi.

Per creare una CA che emetta certificati di breve durata, imposta il `UsageMode` parametro su SHORT_LIVED_CERTIFICATE utilizzando la procedura per la creazione di una CA. AWS CLI

> **Note**
>
> AWS Certificate Manager non può emettere certificati firmati da una CA privata con modalità di breve durata.

L'uso di certificati di breve durata è supportato dai seguenti servizi: AWS

- Amazon AppStream
- Amazon WorkSpaces

Pianificazione della resilienza

L'infrastruttura AWS globale è costruita attorno a AWS regioni e zone di disponibilità. AWS Le regioni forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, puoi progettare e gestire applicazioni e database che eseguono automaticamente il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta Global Infrastructure.AWS

Ridondanza e disaster recovery

Prendi in considerazione la ridondanza e il DR quando pianifichi la gerarchia delle CA. CA privata AWS è disponibile in più regioni, il che consente di creare CA ridondanti in più regioni. Il CA privata AWS servizio funziona con un contratto sul livello di servizio (SLA) con una disponibilità del 99,9%. Esistono almeno due approcci che è possibile prendere in considerazione per la ridondanza e il disaster recovery. È possibile configurare la ridondanza nella CA principale o nella CA subordinata più alta. Ogni approccio ha pro e contro.

1. È possibile creare due CA principali in due AWS regioni diverse per la ridondanza e il disaster recovery. Con questa configurazione, ogni CA principale opera in modo indipendente in una AWS regione, proteggendo l'utente in caso di emergenza in un'unica regione. La creazione di CA root ridondanti, tuttavia, aumenta la complessità operativa: sarà necessario distribuire entrambi i certificati emessi da una CA root negli archivi attendibili dei browser e dei sistemi operativi nell'ambiente.
2. Puoi anche creare CA subordinate ridondanti da distribuire in ciascuna delle tue AWS regioni e concatenarle alla stessa CA principale unica in una singola regione. AWS Il vantaggio di questo approccio è che è necessario distribuire solo un singolo certificato emesso da una CA root negli archivi attendibili dell'ambiente. Il limite è che non si dispone di una CA radice ridondante in caso di emergenza che riguardi la AWS regione in cui si trova la CA principale.

Best practice di CA privata AWS

Le best practice sono raccomandazioni che possono aiutare a utilizzare CA privata AWS in modo efficiente. Le seguenti best practice sono basate sull'esperienza pratica dei clienti AWS Certificate Manager e CA privata AWS attuali.

Documentazione della struttura e delle politiche della CA

AWS consiglia di documentare tutte le policy e le procedure per il funzionamento della CA. Ciò potrebbe includere:

- Ragionamento per le tue decisioni sulla struttura CA
- Diagramma che mostra le CA e le loro relazioni
- Policy sui periodi di validità di una CA
- Pianificazione della successione CA
- Criteri sulla lunghezza del percorso
- Catalogo delle autorizzazioni
- Descrizione delle strutture di controllo amministrativo
- Sicurezza

È possibile acquisire queste informazioni in due documenti, noti come Certification Policy (CP) e Certification Practices Statement (CPS). Fare riferimento alla RFC 3647 per un framework per acquisire informazioni importanti sulle operazioni della CA.

Se possibile, riduci al minimo l'uso della CA principale

In generale, una CA root deve essere utilizzata solo per emettere certificati per CA intermedie. Ciò consente alla CA root di essere archiviata in modo sicuro mentre le CA intermedie eseguono l'attività quotidiana di emissione di certificati di entità finale.

Tuttavia, se la prassi corrente dell'organizzazione consiste nel rilasciare certificati di entità finale direttamente da una CA root, CA privata AWS può supportare questo flusso di lavoro migliorando al contempo la sicurezza e i controlli operativi. L'emissione di certificati di entità finale in questo scenario richiede una policy di autorizzazioni IAM che consenta alla CA principale di utilizzare un modello di

certificato dell'entità finale. Per ulteriori informazioni sulle policy IAM, consulta Identity and Access Management (IAM) per AWS Private Certificate Authority.

> **Note**
>
> Questa configurazione impone limitazioni che potrebbero comportare problemi operativi. Ad esempio, se la CA principale viene compromessa o persa, è necessario creare una nuova CA root e distribuirla a tutti i client dell'ambiente. Fino al completamento del processo di ripristino, non sarà possibile emettere nuovi certificati. L'emissione di certificati direttamente da una CA root impedisce inoltre di limitare l'accesso e limitare il numero di certificati emessi dalla root, che sono entrambe considerate procedure consigliate per la gestione di una CA root.

Assegna alla CA principale la sua Account AWS

Si consiglia di creare una CA principale e una CA subordinata in due AWS account diversi. In questo modo è possibile fornire protezione aggiuntiva e controlli di accesso per la CA root. È possibile eseguire questa operazione esportando la CSR dalla CA subordinata in un account e firmandola con una CA root in un account diverso. Il vantaggio di questo approccio è che è possibile separare il controllo delle CA per account. Lo svantaggio è che non è possibile utilizzare la AWS Management Console procedura guidata per semplificare il processo di firma del certificato CA di una CA subordinata dalla CA principale.

> **Important**
>
> Consigliamo vivamente di utilizzare l'autenticazione a più fattori (MFA) ogni volta che si accede. CA privata AWS

Ruoli di amministratore ed emittente separati

Il ruolo di amministratore della CA deve essere separato dagli utenti che devono solo emettere certificati di entità finale. Se l'amministratore della CA e l'emittente del certificato risiedono nella stessa sedeAccount AWS, puoi limitare le autorizzazioni dell'emittente creando un utente IAM specifico a tale scopo.

Implementa la revoca gestita dei certificati

La revoca gestita invia automaticamente una notifica ai client dei certificati quando un certificato viene revocato. Potrebbe essere necessario revocare un certificato se le relative informazioni crittografiche sono state compromesse o se è stato emesso per errore. I client in genere rifiutano di accettare certificati revocati. CA privata AWSoffre due opzioni standard per la revoca gestita: Online Certificate Status Protocol (OCSP) e elenchi di revoca dei certificati (CRL). Per ulteriori informazioni, consulta Impostazione di un metodo di revoca dei certificati.

Attivare AWS CloudTrail

Attiva la CloudTrail registrazione prima di creare e iniziare a gestire una CA privata. Con CloudTrail, puoi recuperare una cronologia delle chiamate AWS API per il tuo account per monitorare le tue implementazioni. AWS Questa cronologia include le chiamate API effettuate dai servizi AWS Management Console AWS SDK e di livello superioreAWS Command Line Interface. AWS È anche possibile identificare quali utenti e account hanno richiamato le operazioni API PCA, l'indirizzo IP di origine da cui sono state effettuate le chiamate e quando sono avvenute. Puoi integrarti CloudTrail nelle applicazioni utilizzando l'API, automatizzare la creazione di percorsi per la tua organizzazione, controllare lo stato dei percorsi e controllare come gli amministratori attivano e disattivano CloudTrail la registrazione. Per ulteriori informazioni, consultare l'articolo relativo alla Creazione di un trail. Vai a Usando CloudTrail per visualizzare i trail di esempio per le operazioni CA privata AWS.

Ruota la chiave privata CA

È una best practice aggiornare periodicamente la chiave privata della CA privata. Puoi aggiornare una chiave importando un nuovo certificato emesso da una CA oppure sostituire la CA privata con una nuova CA.

> **Note**
>
> Se sostituisci la CA stessa, tieni presente che l'ARN della CA cambia. Ciò causerebbe il fallimento dell'automazione che si basa su un ARN codificato.

Eliminare le CA non utilizzate

È possibile eliminare definitivamente una CA privata. Questa operazione può essere eseguita se la CA non è più necessaria o se si desidera sostituirla con una CA che dispone di una chiave privata più recente. Per eliminare correttamente una CA, si consiglia di seguire il processo descritto in Eliminazione della CA privata.

> **Note**
>
> AWS addebita per una CA fino a quando questa non viene eliminata.

Blocca l'accesso pubblico ai tuoi CRL

CA privata AWSconsiglia di utilizzare la funzionalità Block Public Access (BPA) di Amazon S3 su bucket che contengono CRL. In questo modo si evita di esporre inutilmente i dettagli della tua PKI privata a potenziali avversari. BPA è una best practice di S3 ed è abilitato per impostazione predefinita sui nuovi bucket. In alcuni casi è necessaria una configurazione aggiuntiva. Per ulteriori informazioni, consulta Abilitazione di S3 Block Public Access (BPA) con CloudFront.

Best practice per le applicazioni Amazon EKS

Quando lo utilizzi CA privata AWS per fornire ad Amazon EKS certificati X.509, segui i consigli per proteggere gli ambienti multi-tenant nelle guide alle best practice di Amazon EKS. Per informazioni generali sull'integrazione CA privata AWS con Kubernetes, consulta. Proteggere Kubernetes con CA privata AWS

Amministrazione privata della CA

Utilizzando CA privata AWS, è possibile creare una gerarchia interamente AWS ospitata di autorità di certificazione (CA) principali e subordinate per uso interno da parte dell'organizzazione. Per gestire la revoca dei certificati, è possibile abilitare l'Online Certificate Status Protocol (OCSP), gli elenchi di revoca dei certificati (CRL) o entrambi. CA privata AWS archivia e gestisce i certificati CA, i CRL e le risposte OCSP e le chiavi private delle autorità principali vengono archiviate in modo sicuro da. AWS

> **Note**
>
> L'implementazione OCSP in CA privata AWS non supporta le estensioni di richiesta OCSP. Se si invia una query batch OCSP contenente più certificati, il risponditore AWS OCSP elabora solo il primo certificato in coda e elimina gli altri. Una revoca potrebbe richiedere fino a un'ora prima che venga visualizzata nelle risposte OCSP.

È possibile accedere CA privata AWS utilizzando il AWS Management Console AWS CLI, il e l' CA privata AWS API. I seguenti argomenti illustrano come utilizzare la console e la CLI. Per ulteriori informazioni sull'API, consulta l'AWS Private Certificate Authority API Reference. Per esempi Java che illustrano come utilizzare l'API, consultare Utilizzo dell'CA privata AWSAPI (esempi Java).

Argomenti

Creazione di una CA privata

È possibile utilizzare le procedure descritte in questa sezione per creare CA principali o CA subordinate, ottenendo una gerarchia verificabile di relazioni di fiducia che soddisfi le esigenze organizzative. È possibile creare una CA utilizzando la parte PCA di AWS Management Console, o. AWS CLI AWS CloudFormation

Per informazioni sull'aggiornamento della configurazione di una CA già creata, vedereAggiornamento della CA privata.

Per informazioni sull'utilizzo di una CA per firmare certificati di entità finale per utenti, dispositivi e applicazioni, vedere. Emissione di certificati privati per entità finali

> **Note**
>
> Al tuo account viene addebitato un prezzo mensile per ogni CA privata a partire dal momento in cui la crei.
> Per le informazioni più recenti CA privata AWS sui prezzi, consulta AWS Private Certificate Authority Prezzi. Puoi anche utilizzare il calcolatore dei AWS prezzi per stimare i costi.

Argomenti

- Procedura per la creazione di una CA (console)
- Procedura per la creazione di una CA (CLI)
- Utilizzo AWS CloudFormation per creare una CA

Procedura per la creazione di una CA (console)

Completa i seguenti passaggi per creare una CA privata utilizzando. AWS Management Console

Per iniziare a utilizzare la console

Accedi al tuo AWS account e apri la CA privata AWS console all'indirizzo**https://console.aws.amazon.com/acm-pca/home**.

- Se apri la console in una regione in cui non disponi di CA private, viene visualizzata la pagina introduttiva. Scegli Crea una CA privata.

- Se stai aprendo la console in una regione in cui hai già creato una CA, si apre la pagina Autorità di certificazione private con un elenco delle tue CA. Scegli Crea CA.

Altre opzioni

Nella sezione Opzioni di modalità della console, scegli la modalità di scadenza dei certificati emessi dalla tua CA.

- Utilizzo generico: emette certificati che possono essere configurati con qualsiasi data di scadenza. Questa è l'impostazione predefinita.
- Certificato di breve durata: rilascia certificati con un periodo di validità massimo di sette giorni. Un breve periodo di validità può sostituire in alcuni casi un meccanismo di revoca.

Opzioni per tipo CA

Nella sezione Opzioni di tipo della console, scegli il tipo di autorità di certificazione privata che desideri creare.

- La scelta di Root stabilisce una nuova gerarchia CA. Questa CA è supportata da un certificato autofirmato. Funge da autorità di firma definitiva per le altre CA e i certificati di entità finale nella gerarchia.
- Scegliendo Subordinate si crea una CA che deve essere firmata da una CA principale che si trova al di sopra di essa nella gerarchia. Le CA subordinate vengono in genere utilizzate per creare altre CA subordinate o per emettere certificati di entità finale a utenti, computer e applicazioni.

> **Note**
>
> CA privata AWS fornisce un processo di firma automatizzato quando la CA principale della CA subordinata è ospitata anche da. CA privata AWS Tutto ciò che devi fare è scegliere la CA principale da utilizzare.
> Potrebbe essere necessario che la CA subordinata sia firmata da un fornitore di servizi fiduciari esterno. In tal caso, CA privata AWS fornisce una richiesta di firma del certificato (CSR) da scaricare e utilizzare per ottenere un certificato CA firmato. Per ulteriori informazioni, consulta Installazione di un certificato CA subordinato firmato da una CA principale esterna.

Opzioni relative al nome distinto del soggetto

In Opzioni relative al nome distinto del soggetto, configura il nome dell'oggetto della tua CA privata. È necessario immettere un valore per almeno una delle seguenti opzioni:

- Organizzazione (O): ad esempio, il nome di una società
- Unità organizzativa (OU): ad esempio, una divisione all'interno di un'azienda
- Nome del paese (C): un codice del paese di due lettere
- Nome dello stato o della provincia: nome completo di uno stato o di una provincia
- Nome località: il nome di una città
- Nome comune (CN): una stringa leggibile dall'uomo per identificare la CA.

> **Note**
>
> È possibile personalizzare ulteriormente il nome dell'oggetto di un certificato applicando un modello ApiPassthrough al momento dell'emissione. Per ulteriori informazioni e un esempio dettagliato, vedere. Emetti un certificato con un nome oggetto personalizzato utilizzando un modello ApiPassThrough

Poiché il certificato di supporto è autofirmato, le informazioni sull'oggetto fornite per una CA privata sono probabilmente più scarse di quelle che conterrebbe una CA pubblica. Per ulteriori informazioni su ciascuno dei valori che compongono il nome distinto del soggetto, vedere RFC 5280.

Opzioni chiave dell'algoritmo

In Opzioni dell'algoritmo chiave, scegli l'algoritmo chiave e la dimensione in bit della chiave. Il valore predefinito è un algoritmo RSA con una lunghezza di chiave di 2048 bit. Puoi scegliere tra i seguenti algoritmi:

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

Opzioni di revoca del certificato

In Opzioni di revoca dei certificati, puoi scegliere tra due metodi per condividere lo stato di revoca con i client che utilizzano i tuoi certificati:

- Attiva la distribuzione CRL
- Attiva OCSP

È possibile configurare una, nessuna delle due o entrambe queste opzioni di revoca per la CA. Sebbene facoltativa, la revoca gestita è consigliata come best practice. Prima di completare questo passaggio, consulta Impostazione di un metodo di revoca dei certificati le informazioni sui vantaggi di ciascun metodo, sulla configurazione preliminare che potrebbe essere richiesta e sulle funzionalità di revoca aggiuntive.

> **Note**
>
> Se si crea la CA senza configurare la revoca, è sempre possibile configurarla in un secondo momento. Per ulteriori informazioni, consulta Aggiornamento della CA privata.

Per configurare un CRL

1. In Opzioni di revoca del certificato, scegli Attiva la distribuzione CRL.
2. Per creare un bucket Amazon S3 per le voci del CRL, scegli Crea un nuovo bucket S3 e digita un nome di bucket univoco. (Non è necessario includere il percorso del bucket.) Altrimenti, in URI del bucket S3, scegli un bucket esistente dall'elenco.

   Quando crei un nuovo bucket tramite la console, CA privata AWS tenta di allegare al bucket la politica di accesso richiesta e di disabilitare l'impostazione Block Public Access (BPA) predefinita di S3 su di esso. Se invece specifichi un bucket esistente, devi assicurarti che BPA sia disabilitato per l'account e per il bucket. In caso contrario, l'operazione di creazione della CA avrà esito negativo. Se la CA viene creata correttamente, è comunque necessario allegarle manualmente una policy prima di poter iniziare a generare CRL. Utilizza uno dei modelli di policy descritti inPolitiche di accesso per i CRL in Amazon S3 . Per ulteriori informazioni, consulta Aggiungere una policy sui bucket utilizzando la console Amazon S3.

> **Important**
>
> Un tentativo di creare una CA utilizzando la CA privata AWS console fallisce se si verificano tutte le seguenti condizioni:
>
> - Stai configurando un CRL.
> - Chiedi di CA privata AWS creare automaticamente un bucket S3.
> - Stai applicando le impostazioni BPA in S3.
>
> In questa situazione, la console crea un bucket, ma tenta e fallisce di renderlo accessibile al pubblico. Controlla le impostazioni di Amazon S3 se ciò si verifica, disabilita BPA secondo necessità, quindi ripeti la procedura per la creazione di una CA. Per ulteriori informazioni, consulta Bloccare l'accesso pubblico allo storage Amazon S3.

3. Espandi le impostazioni CRL per opzioni di configurazione aggiuntive.

   - Aggiungi un nome CRL personalizzato per creare un alias per il tuo bucket Amazon S3. Questo nome è contenuto nei certificati emessi dalla CA nell'estensione «CRL Distribution Points» definita da RFC 5280.
   - Digita la Validità in giorni il tuo CRL rimarrà valido. Il valore predefinito è 7 giorni. Per i CRL online, è comune un periodo di validità di 2-7 giorni. CA privata AWS tenta di rigenerare il CRL a metà del periodo specificato.

4. Espandi le impostazioni S3 per la configurazione opzionale del controllo delle versioni di Bucket e della registrazione degli accessi a Bucket.

Per configurare OCSP

1. In Opzioni di revoca del certificato, scegli Attiva OCSP.
2. Nel campo Endpoint OCSP personalizzato - opzionale, puoi fornire un nome di dominio completo (FQDN) per un endpoint non Amazon OCSP.

   Quando fornisci un FQDN in questo campo, CA privata AWS inserisce il nome di dominio completo nell'estensione Authority Information Access di ciascun certificato emesso al posto dell'URL predefinito per il risponditore OCSP. AWS Quando un endpoint riceve un certificato contenente l'FQDN personalizzato, richiede a tale indirizzo una risposta OCSP. Affinché questo meccanismo funzioni, è necessario eseguire due azioni aggiuntive:

- Utilizzate un server proxy per inoltrare il traffico che arriva al vostro FQDN personalizzato al risponditore AWS OCSP.
- Aggiungi un record CNAME corrispondente al tuo database DNS.

> **Tip**
>
> Per ulteriori informazioni sull'implementazione di una soluzione OCSP completa utilizzando un CNAME personalizzato, vedere. Configurazione di un URL personalizzato per OCSP CA privata AWS

Ad esempio, ecco un record CNAME per OCSP personalizzato come apparirebbe in Amazon Route 53.

Nome record	Type	Policy di routing	Differenziatore	Valore/instradamento traffico a
alternative.example.com	CNAME	Semplice	-	proxy.example.com

> **Note**
>
> Il valore del CNAME non deve includere un prefisso di protocollo come «http://» o «https://».

Aggiunta di tag

In Aggiungi tag, puoi opzionalmente taggare la tua CA. I tag sono coppie chiave-valore che fungono da metadati per identificare e organizzare le risorse AWS . Per un elenco dei parametri dei CA privata AWS tag e per istruzioni su come aggiungere tag alle CA dopo la creazione, consulta. Gestione dei tag per la tua CA privata

> **Note**
>
> Per allegare tag a una CA privata durante la procedura di creazione, un amministratore CA deve prima associare una policy IAM in linea all'`CreateCertificateAuthority`azione e consentire esplicitamente l'etichettatura. Per ulteriori informazioni, consulta Tag-on-create: Allegare tag a una CA al momento della creazione.

Opzioni di autorizzazione CA

Nelle opzioni di autorizzazione CA, è possibile delegare facoltativamente le autorizzazioni di rinnovo automatico al responsabile del servizio. AWS Certificate Manager ACM può rinnovare automaticamente i certificati privati di entità finale generati da questa CA solo se questa autorizzazione viene concessa. Puoi assegnare le autorizzazioni di rinnovo in qualsiasi momento con l' CA privata AWS CreatePermissionAPI o il comando CLI create-permission.

L'impostazione predefinita consiste nell'abilitare queste autorizzazioni.

> **Note**
>
> AWS Certificate Manager non supporta il rinnovo automatico dei certificati di breve durata.

Prezzi

Nella sezione Prezzi, conferma di aver compreso i prezzi di una CA privata.

> **Note**
>
> Per le informazioni più recenti CA privata AWS sui prezzi, consulta la sezione AWS Private Certificate Authority Prezzi. Puoi anche utilizzare il calcolatore dei AWS prezzi per stimare i costi.

Crea CA

Scegli Crea CA dopo aver verificato l'accuratezza di tutte le informazioni inserite. Si apre la pagina dei dettagli della CA e ne mostra lo stato come Certificato in sospeso.

> **Note**
>
> Nella pagina dei dettagli, è possibile completare la configurazione della CA scegliendo Azioni, Installa certificato CA oppure tornare in un secondo momento all'elenco delle autorità di certificazione private e completare la procedura di installazione applicabile al caso specifico:
>
> - Installazione di un certificato CA root
> - Installazione di un certificato CA subordinato ospitato da CA privata AWS
> - Installazione di un certificato CA subordinato firmato da una CA principale esterna

Procedura per la creazione di una CA (CLI)

Utilizza il comando create-certificate-authority per creare una CA privata. È necessario specificare la configurazione della CA (contenente informazioni sull'algoritmo e sul nome dell'oggetto), la configurazione di revoca (se si prevede di utilizzare OCSP e/o un CRL) e il tipo di CA (principale o subordinato). I dettagli della configurazione e della revoca sono contenuti in due file forniti come argomenti del comando. Facoltativamente, è anche possibile configurare la modalità di utilizzo della CA (per l'emissione di certificati standard o di breve durata), allegare tag e fornire un token di idempotenza.

Se stai configurando un CRL, devi disporre di un bucket Amazon S3 sicuro prima di emettere il comando. create-certificate-authority Per ulteriori informazioni, consulta Politiche di accesso per i CRL in Amazon S3 .

Il file di configurazione CA specifica le seguenti informazioni:

- Il nome dell'algoritmo
- La dimensione della chiave da utilizzare per creare la chiave privata CA
- Il tipo di algoritmo di firma che la CA utilizza per firmare
- Informazioni sull'oggetto X.500

La configurazione di revoca per OCSP definisce un `OcspConfiguration` oggetto con le seguenti informazioni:

- Il `Enabled` flag impostato su «true».
- (Facoltativo) Un CNAME personalizzato dichiarato come valore per`OcspCustomCname`.

La configurazione di revoca per un CRL definisce un `CrlConfiguration` oggetto con le seguenti informazioni:

- Il `Enabled` flag impostato su «true».
- Il periodo di scadenza del CRL in giorni (periodo di validità del CRL).
- Il bucket Amazon S3 che conterrà il CRL.
- (Facoltativo) Un ObjectAcl valore S3 che determina se il CRL è accessibile al pubblico. Nell'esempio qui presentato, l'accesso pubblico è bloccato. Per ulteriori informazioni, consulta Abilitazione di S3 Block Public Access (BPA) con CloudFront.
- (Facoltativo) Un alias CNAME per il bucket S3 incluso nei certificati emessi dalla CA. Se il CRL non è accessibile al pubblico, ciò indicherà un meccanismo di distribuzione come Amazon CloudFront.
- (Facoltativo) Un `CrlDistributionPointExtensionConfiguration` oggetto con le seguenti informazioni:
  - Il `OmitExtension` flag impostato su «true» o «false». Questo controlla se il valore predefinito per l'estensione CDP verrà scritto su un certificato emesso dalla CA. Per ulteriori informazioni sull'estensione CDP, vedere. Determinazione dell'URI del punto di distribuzione CRL (CDP) A CustomCname non può essere impostato se OmitExtension è «true».

> **Note**
>
> È possibile abilitare entrambi i meccanismi di revoca sulla stessa CA definendo sia un `OcspConfiguration` oggetto che un `CrlConfiguration` oggetto. Se non si fornisce alcun --revocation-configuration parametro, entrambi i meccanismi sono disabilitati per impostazione predefinita. Se in un secondo momento è necessario il supporto per la convalida della revoca, consulta. Aggiornamento di una CA (CLI)

Gli esempi seguenti presuppongono che la directory di `.aws` configurazione sia stata configurata con una regione, un endpoint e delle credenziali predefiniti validi. Per informazioni sulla configurazione AWS CLI dell'ambiente, consulta Configurazione e impostazioni dei file di credenziali. Per motivi di leggibilità, forniamo l'input di configurazione e revoca della CA come file JSON nei comandi di esempio. Modificate i file di esempio in base alle vostre esigenze.

Tutti gli esempi utilizzano il seguente file `ca_config.txt` di configurazione, salvo diversa indicazione.

File: ca_config.txt

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

Esempio 1: creare una CA con OCSP abilitato

In questo esempio, il file di revoca abilita il supporto OCSP predefinito, che utilizza il CA privata AWS risponditore per verificare lo stato del certificato.

File: revoke_config.txt per OCSP

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

Comando

```
$ 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
```

In caso di successo, questo comando genera l'Amazon Resource Name (ARN) della nuova CA.

```
{
 "CertificateAuthorityArn":"arn:aws:acm-pca:region:account:
```

```
       certificate-authority/CA_ID"
}
```

Comando

```
$ 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
```

In caso di successo, questo comando genera l'Amazon Resource Name (ARN) della CA.

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

Usa il seguente comando per controllare la configurazione della tua 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
```

Questa descrizione deve contenere la sezione seguente.

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

Esempio 2: creare una CA con OCSP e un CNAME personalizzato abilitato

In questo esempio, il file di revoca abilita il supporto OCSP personalizzato. Il `OcspCustomCname` parametro accetta come valore un nome di dominio completo (FQDN).

Quando si fornisce un FQDN in questo campo, CA privata AWS inserisce il nome di dominio completo nell'estensione Authority Information Access di ciascun certificato emesso al posto dell'URL predefinito per il risponditore OCSP. AWS Quando un endpoint riceve un certificato contenente l'FQDN personalizzato, richiede a tale indirizzo una risposta OCSP. Affinché questo meccanismo funzioni, è necessario eseguire due azioni aggiuntive:

- Utilizzate un server proxy per inoltrare il traffico che arriva al vostro FQDN personalizzato al risponditore AWS OCSP.
- Aggiungi un record CNAME corrispondente al tuo database DNS.

> **Tip**
>
> Per ulteriori informazioni sull'implementazione di una soluzione OCSP completa utilizzando un CNAME personalizzato, vedere. Configurazione di un URL personalizzato per OCSP CA privata AWS

Ad esempio, ecco un record CNAME per OCSP personalizzato come apparirebbe in Amazon Route 53.

Nome record	Type	Policy di routing	Differenziatore	Valore/instradamento traffico a
alternative.example.com	CNAME	Semplice	-	proxy.example.com

> **Note**
>
> Il valore del CNAME non deve includere un prefisso di protocollo come «http://» o «https://».

File: revoke_config.txt per OCSP

```
{
   "OcspConfiguration":{
      "Enabled":true,
```

```
      "OcspCustomCname":"alternative.example.com"
   }
}
```

Comando

```
$ 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
```

In caso di successo, questo comando genera l'Amazon Resource Name (ARN) della CA.

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

Usa il seguente comando per controllare la configurazione della tua 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
```

Questa descrizione deve contenere la sezione seguente.

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

Esempio 3: creare una CA con un CRL allegato

In questo esempio, la configurazione di revoca definisce i parametri CRL.

File: revoke_config.txt

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

Comando

```
$ 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
```

In caso di successo, questo comando genera l'Amazon Resource Name (ARN) della CA.

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

Usa il seguente comando per controllare la configurazione della tua 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
```

Questa descrizione deve contenere la sezione seguente.

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

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

Esempio 4: creazione di una CA con un CRL allegato e un CNAME personalizzato abilitato

In questo esempio, la configurazione di revoca definisce i parametri CRL che includono un CNAME personalizzato.

File: revoke_config.txt

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

Comando

```
$ 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
```

In caso di successo, questo comando genera l'Amazon Resource Name (ARN) della CA.

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

Usa il seguente comando per controllare la configurazione della tua 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
```

Questa descrizione deve contenere la sezione seguente.

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

Esempio 5: creare una CA e specificare la modalità di utilizzo

In questo esempio, la modalità di utilizzo della CA viene specificata durante la creazione di una CA. Se non è specificato, il parametro della modalità di utilizzo è predefinito su GENERAL_PURPOSE. In questo esempio, il parametro è impostato su SHORT_LIVED_CERTIFICATE, il che significa che la CA emetterà certificati con un periodo di validità massimo di sette giorni. In situazioni in cui è scomodo configurare la revoca, un certificato di breve durata che è stato compromesso scade rapidamente nell'ambito delle normali operazioni. Di conseguenza, questo esempio di CA non dispone di un meccanismo di revoca.

> **Note**
>
> CA privata AWS non esegue controlli di validità sui certificati 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
```

Utilizzare il describe-certificate-authoritycomando in AWS CLI per visualizzare i dettagli sulla CA risultante, come illustrato nel comando seguente:

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

Esempio 6: creazione di una CA per l'accesso ad Active Directory

È possibile creare una CA privata adatta all'uso nell'archivio Enterprise NTAuth di Microsoft Active Directory (AD), dove può emettere certificati card-logon o domain-controller. Per informazioni sull'importazione di un certificato CA in AD, vedere Come importare certificati di autorità di certificazione (CA) di terze parti nell'archivio Enterprise NTAuth.

Lo strumento Microsoft certutil può essere utilizzato per pubblicare certificati CA in AD richiamando l'opzione. -dspublish Un certificato pubblicato su AD con certutil è considerato affidabile in tutta la foresta. Utilizzando i criteri di gruppo, è inoltre possibile limitare l'attendibilità a un sottoinsieme dell'intera foresta, ad esempio un singolo dominio o un gruppo di computer in un dominio. Affinché l'accesso funzioni, la CA emittente deve essere pubblicata anche nell'archivio NTAuth. Per ulteriori informazioni, consulta Distribuire certificati ai computer client utilizzando criteri di gruppo.

Questo esempio utilizza il seguente file `ca_config_AD.txt` di configurazione.

File: 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"
         }
      ]
   }
}
```

Comando

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

In caso di successo, questo comando genera l'Amazon Resource Name (ARN) della CA.

```
{
```

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

Usa il seguente comando per controllare la configurazione della tua 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
```

Questa descrizione deve contenere la sezione seguente.

```
...

"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"
      }
   ]
}
...
```

Esempio 7: creazione di una Matter CA con un CRL allegato e l'estensione CDP omessa dai certificati emessi

È possibile creare una CA privata adatta all'emissione di certificati per lo standard Matter per la casa intelligente. In questo esempio, la configurazione CA in `ca_config_PAA.txt` definisce una Matter Product Attestation Authority (PAA) con il Vendor ID (VID) impostato su FFF1.

File: ca_config_PAA.txt

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

La configurazione di revoca abilita i CRL e configura la CA in modo da omettere l'URL CDP predefinito da tutti i certificati emessi.

File: revoke_config.txt

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

Comando

```
$ 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
```

In caso di successo, questo comando genera l'Amazon Resource Name (ARN) della CA.

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

Usa il seguente comando per controllare la configurazione della tua 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
```

Questa descrizione deve contenere la sezione seguente.

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

Utilizzo AWS CloudFormation per creare una CA

Per informazioni sulla creazione di una CA privata utilizzando AWS CloudFormation, consulta CA privata AWS Resource Type Reference nella Guida per l'AWS CloudFormation utente.

Creazione e installazione del certificato CA

Completare le procedure seguenti per creare e installare il certificato emesso da una CA privata. La CA sarà quindi pronta per l'uso.

CA privata AWS supporta tre scenari per l'installazione di un certificato CA:

- Installazione di un certificato per una CA root ospitata da CA privata AWS
- Installazione di un certificato emesso da una CA subordinata la cui autorità padre è ospitata da CA privata AWS
- Installazione di un certificato emesso da una CA subordinata la cui autorità padre è ospitata esternamente

Nelle sezioni seguenti vengono descritte le procedure per ogni scenario. Le procedure della console iniziano nella pagina della console CA private.

Algoritmi di firma compatibili

Il supporto dell'algoritmo di firma per i certificati CA dipende dall'algoritmo di firma della CA principale e da. Regione AWS I seguenti vincoli si applicano sia alla console che alle operazioni. AWS CLI

- Una CA principale con l'algoritmo di firma RSA può emettere certificati con i seguenti algoritmi:
  - SHA256 RSA
  - SHA384 RSA
  - SHA512 RSA
- In una versione precedente Regione AWS, una CA principale con l'algoritmo di firma EDCSA può emettere certificati con i seguenti algoritmi:
  - SHA256 ECDSA
  - SHA384 ECDSA
  - SHA512 ECDSA

  L'eredità Regioni AWS include:

Nome Regione	Ubicazione geografica
eu-north-1	Europa (Stoccolma)
me-south-1	Medio Oriente (Bahrein)

Nome Regione	Ubicazione geografica
ap-south-1	Asia Pacifico (Mumbai)
eu-west-3	Europa (Parigi)
us-east-2	Stati Uniti orientali (Ohio)
af-south-1	Africa (Città del Capo)
eu-west-1	Europa (Irlanda)
eu-central-1	Europa (Francoforte)
sa-east-1	Sud America (San Paolo)
ap-east-1	Asia Pacifico (Hong Kong)
us-east-1	Stati Uniti orientali (Virginia settentrionale)
ap-northeast-2	Asia Pacifico (Seul)
eu-west-2	Europa (Londra)
ap-northeast-1	Asia Pacifico (Tokyo)
us-gov-east-1	AWS GovCloud (Stati Uniti orientali)
us-gov-west-1	AWS GovCloud (Stati Uniti occidentali)

Nome Regione	Ubicazione geografica
us-west-2	US West (Oregon)
us-west-1	Stati Uniti occidentali (California settentrionale)
ap-southeast-1	Asia Pacifico (Singapore)
ap-southeast-2	Asia Pacifico (Sydney)

- In un caso non preesistente Regione AWS, all'EDCSA si applicano le seguenti regole:
  - Una CA principale con l'algoritmo di firma EC_Prime256v1 può emettere certificati con ECDSA P256.
  - Una CA principale con l'algoritmo di firma EC_SECP384R1 può emettere certificati con ECDSA P384.

Installazione di un certificato CA root

È possibile installare un certificato CA root da AWS Management Console o da AWS CLI.

Per creare e installare un certificato per la CA (console) root privata

1. (Facoltativo) Se non sei già nella pagina dei dettagli della CA, apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home. Nella pagina Autorità di certificazione private, scegli una CA root con stato Certificato in sospeso o Attivo.
2. Scegli Azioni, Installa certificato CA per aprire la pagina Installa certificato CA root.
3. In Specificare i parametri del certificato CA principale, specificare i seguenti parametri del certificato:
   - Validità: specifica la data e l'ora di scadenza del certificato CA. Il periodo di validità CA privata AWS predefinito per un certificato CA principale è di 10 anni.
   - Algoritmo di firma: specifica l'algoritmo di firma da utilizzare quando la CA principale emette nuovi certificati. Le opzioni disponibili variano a seconda del Regione AWS luogo in cui si

crea la CA. Per ulteriori informazioni, vedere Algoritmi di firma compatibiliAlgoritmi crittografici supportati, e SigningAlgorithmin CertificateAuthorityConfiguration.

- SHA256 RSA
- SHA384 RSA
- SHA512 RSA

Controlla le impostazioni per verificarne la correttezza, quindi scegli Conferma e installa. CA privata AWS esporta una CSR per la tua CA, genera un certificato utilizzando un modello di certificato CA principale e firma automaticamente il certificato. CA privata AWS quindi importa il certificato CA root autofirmato.

4. La pagina dei dettagli della CA mostra lo stato dell'installazione (riuscita o fallita) nella parte superiore. Se l'installazione ha avuto esito positivo, la CA root appena completata visualizza lo stato Attivo nel riquadro Generale.

Per creare e installare un certificato per la vostra CA root privata (AWS CLI)

1. Genera una richiesta di firma del certificato (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
```

Il file risultanteca.csr, un file PEM codificato in formato base64, ha il seguente aspetto.

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

Puoi usare OpenSSL per visualizzare e verificare il contenuto della CSR.

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

Ciò produce un output simile al seguente.

```
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. Utilizzando la CSR del passaggio precedente come argomento per il `--csr` parametro, emettete il certificato principale.

> **Note**
>
> Se utilizzate la AWS CLI versione 1.6.3 o successiva, utilizzate il prefisso `fileb://` quando specificate il file di input richiesto. Ciò garantisce che i dati codificati in Base64 vengano CA privata AWS analizzati correttamente.

```
$ 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. Recupera il certificato principale.

```
$ 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
```

Il file risultante`cert.pem`, un file PEM codificato in formato base64, ha il seguente aspetto.

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

È possibile utilizzare OpenSSL per visualizzare e verificare il contenuto del certificato.

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

Ciò produce un output simile al seguente.

```
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. Importa il certificato CA principale per installarlo sulla CA.

> **Note**
>
> Se si utilizza la AWS CLI versione 1.6.3 o successiva, utilizzare il prefisso per specificare `fileb://` il file di input richiesto. Ciò garantisce che i dati codificati in Base64 vengano CA privata AWS analizzati correttamente.

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

Ispeziona il nuovo stato della 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
```

Lo stato ora appare come ATTIVO.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-05T14:24:12.867000-08:00",
        "LastStateChangeAt": "2021-03-08T12:37:14.235000-08:00",
```

```
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T07:46:27-08:00",
        "NotAfter": "2022-03-08T08:46:27-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "CustomCname": "alternative.example.com",
                "S3BucketName": "DOC-EXAMPLE-BUCKET1"
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

Installazione di un certificato CA subordinato ospitato da CA privata AWS

È possibile utilizzare il AWS Management Console per creare e installare un certificato per la CA subordinata CA privata AWS ospitata.

Per creare e installare un certificato per la CA subordinata CA privata AWS ospitata

1. (Facoltativo) Se non sei già nella pagina dei dettagli della CA, apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home. Nella pagina Autorità di certificazione private, scegli una CA subordinata con lo stato Certificato in sospeso o Attivo.
2. Scegli Azioni, Installa certificato CA per aprire la pagina Installa certificato CA subordinato.

3. Nella pagina Installa certificato CA subordinato, in Seleziona il tipo di CA, scegli AWS Private CAdi installare un certificato gestito da. CA privata AWS
4. In Seleziona CA principale, scegli una CA dall'elenco delle CA private principali. L'elenco viene filtrato per visualizzare le CA che soddisfano i seguenti criteri:
   - L'utente è autorizzato a utilizzare la CA.
   - La CA non si firmerebbe da sola.
   - La CA è in stato`ACTIVE`.
   - La modalità CA è`GENERAL_PURPOSE`.
5. In Specificare i parametri del certificato CA subordinato, specificare i seguenti parametri del certificato:
   - Validità: specifica la data e l'ora di scadenza del certificato CA.
   - Algoritmo di firma: specifica l'algoritmo di firma da utilizzare quando la CA principale emette nuovi certificati. Le opzioni sono:
     - SHA256 RSA
     - SHA384 RSA
     - SHA512 RSA
   - Lunghezza del percorso: il numero di livelli di fiducia che la CA subordinata può aggiungere quando firma nuovi certificati. Una lunghezza del percorso pari a zero (impostazione predefinita) significa che è possibile creare solo certificati di entità finale e non certificati CA. Una lunghezza del percorso di uno o più significa che la CA subordinata può emettere certificati per creare ulteriori CA subordinate ad essa.
   - ARN del modello: visualizza l'ARN del modello di configurazione per questo certificato CA. Il modello cambia se si modifica la lunghezza del percorsospecificata. Se si crea un certificato utilizzando il comando CLI issue-certificate o l'IssueCertificateazione API, è necessario specificare l'ARN manualmente. Per informazioni sui modelli di certificato emessi da una CA disponibili, consulta Informazioni sui modelli di certificato.
6. Controlla la correttezza delle impostazioni, quindi scegli Conferma e installa. CA privata AWS esporta una CSR, genera un certificato utilizzando un modello di certificato CA subordinato e lo firma con la CA principale selezionata. CA privata AWS quindi importa il certificato CA subordinato firmato.

7. La pagina dei dettagli della CA mostra lo stato dell'installazione (riuscita o fallita) nella parte superiore. Se l'installazione ha avuto esito positivo, la CA subordinata appena completata visualizza lo stato Attivo nel riquadro Generale.

Installazione di un certificato CA subordinato firmato da una CA principale esterna

Dopo aver creato una CA privata subordinata come descritto in Procedura per la creazione di una CA (console) oppureProcedura per la creazione di una CA (CLI) , è possibile attivarla installando un certificato CA firmato da un'autorità di firma esterna. La firma del certificato CA subordinato con una CA esterna richiede innanzitutto la configurazione di un provider di servizi fiduciari esterno come autorità di firma o l'utilizzo di un provider di terze parti.

> **Note**
>
> Le procedure per creare o ottenere un fornitore esterno di servizi fiduciari non rientrano nell'ambito di questa guida.

Dopo aver creato una CA subordinata e aver avuto accesso a un'autorità di firma esterna, completa le seguenti attività:

1. Ottieni una richiesta di firma del certificato (CSR) da. CA privata AWS
2. Invia la CSR all'autorità di firma esterna e ottieni un certificato CA firmato insieme a tutti i certificati della catena.
3. Importa il certificato CA e la catena in CA privata AWS per attivare la CA subordinata.

Per le procedure dettagliate, consulta Certificati CA privati firmati esternamente .

Controllo dell'accesso a una CA privata

Qualsiasi utente con le autorizzazioni necessarie su una CA privata CA privata AWS può utilizzare tale CA per firmare altri certificati. Il proprietario della CA può emettere certificati o delegare le autorizzazioni necessarie per l'emissione di certificati a un utente AWS Identity and Access Management (IAM) che risiede nella stessa. Account AWSUn utente che risiede in un AWS account

diverso può anche emettere certificati se autorizzato dal proprietario della CA tramite una politica basata sulle risorse.

Gli utenti autorizzati, indipendentemente dal fatto che si tratti di account singoli o multiaccount, possono utilizzare CA privata AWS le nostre risorse per l'emissione dei certificati. AWS Certificate Manager I certificati emessi dall' CA privata AWS IssueCertificateAPI o dal comando CLI issue-certificate non sono gestiti. Tali certificati richiedono l'installazione manuale sui dispositivi di destinazione e il rinnovo manuale alla scadenza. I certificati emessi dalla console ACM, dall'RequestCertificateAPI ACM o dal comando CLI request-certificate vengono gestiti. Tali certificati possono essere facilmente installati in servizi integrati con ACM. Se l'amministratore della CA lo consente e l'account dell'emittente ha un ruolo collegato al servizio per ACM, i certificati gestiti vengono rinnovati automaticamente alla scadenza.

Argomenti

- Crea autorizzazioni per account singolo per un utente IAM
- Allega una politica per l'accesso tra più account

Crea autorizzazioni per account singolo per un utente IAM

Quando l'amministratore della CA (ovvero il proprietario della CA) e l'emittente del certificato risiedono in un unico AWS account, è consigliabile separare i ruoli di emittente e amministratore creando un utente AWS Identity and Access Management (IAM) con autorizzazioni limitate. Per informazioni sull'utilizzo di IAM con CA privata AWS, oltre ad esempi di autorizzazioni, consulta. Identity and Access Management (IAM) per AWS Private Certificate Authority

Caso 1 con account singolo: emissione di un certificato non gestito

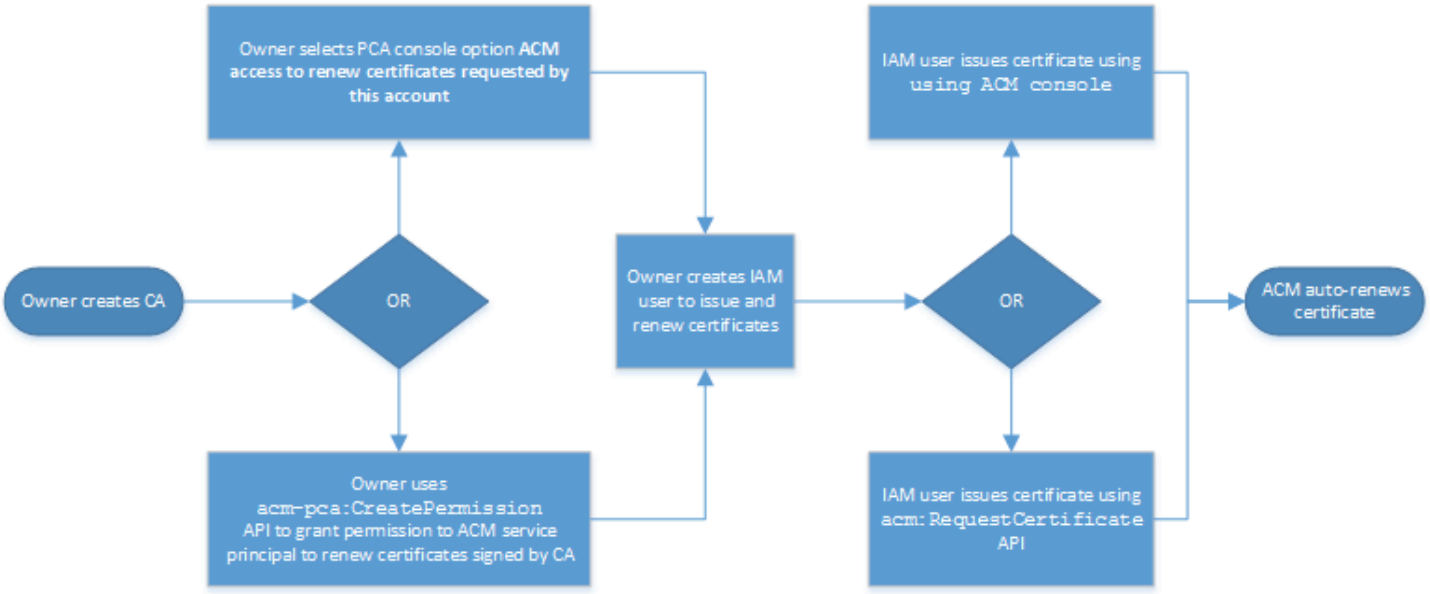
In questo caso, il proprietario dell'account crea una CA privata e quindi crea un utente IAM con l'autorizzazione a emettere certificati firmati dalla CA privata. L'utente IAM emette un certificato chiamando l' CA privata AWS `IssueCertificate`API.

I certificati emessi in questo modo non sono gestiti, il che significa che un amministratore deve esportarli e installarli sui dispositivi in cui sono destinati a essere utilizzati. Inoltre, devono essere rinnovati manualmente quando scadono. L'emissione di un certificato utilizzando questa API richiede

una richiesta di firma del certificato (CSR) e una coppia di chiavi generati all'esterno da CA privata AWS OpenSSL o da un programma simile. Per ulteriori informazioni, consulta la documentazione. `IssueCertificate`

Caso 2 con account singolo: emissione di un certificato gestito tramite ACM

Questo secondo caso riguarda le operazioni API di ACM e PCA. Il proprietario dell'account crea un utente CA e IAM privato come in precedenza. Il proprietario dell'account concede quindi l'autorizzazione al responsabile del servizio ACM per rinnovare automaticamente tutti i certificati firmati da questa CA. L'utente IAM emette nuovamente il certificato, ma questa volta chiamando l'`RequestCertificate`API ACM, che gestisce la CSR e la generazione delle chiavi. Quando il certificato scade, ACM automatizza il flusso di lavoro per il rinnovo.



Il proprietario dell'account ha la possibilità di concedere l'autorizzazione al rinnovo tramite la console di gestione durante o dopo la creazione della CA o utilizzando l'API `CreatePermission` PCA. I certificati gestiti creati da questo flusso di lavoro possono essere utilizzati con AWS servizi integrati con ACM.

La sezione seguente contiene le procedure per la concessione delle autorizzazioni di rinnovo.

Assegna le autorizzazioni per il rinnovo dei certificati ad ACM

Con Managed Renewal in AWS Certificate Manager (ACM), puoi automatizzare il processo di rinnovo dei certificati sia per i certificati pubblici che per quelli privati. Affinché ACM possa rinnovare automaticamente i certificati generati da una CA privata, al responsabile del servizio ACM devono essere concesse tutte le autorizzazioni possibili dalla CA stessa. Se queste autorizzazioni di rinnovo

non sono presenti per ACM, il proprietario della CA (o un rappresentante autorizzato) deve riemettere manualmente ogni certificato privato alla scadenza.

> **Important**
>
> Queste procedure per l'assegnazione delle autorizzazioni di rinnovo si applicano solo quando il proprietario della CA e l'emittente del certificato risiedono nello stesso account. AWS Per scenari che coinvolgono più account, consulta. Allega una politica per l'accesso tra più account

Le autorizzazioni di rinnovo possono essere delegate durante la creazione di CA privata o modificate in qualsiasi momento dopo tutto il tempo in cui la CA si trova nello stato `ACTIVE`.

Puoi gestire le autorizzazioni CA private dalla Console CA privata AWS, dall'AWS Command Line Interface (AWS CLI), o dall'API CA privata AWS:

Per assegnare autorizzazioni CA private ad ACM (console)

1. Accedi al tuo AWS account e apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home.
2. Nella pagina Autorità di certificazione private, scegli la tua CA privata dall'elenco.
3. Scegli Azioni, Configura le autorizzazioni CA.
4. Seleziona Autorizza l'accesso ACM per rinnovare i certificati richiesti da questo account.
5. Selezionare Salva.

Per gestire le autorizzazioni ACM in () CA privata AWSAWS CLI

Utilizzate il comando create-permission per assegnare le autorizzazioni ad ACM. È necessario assegnare le autorizzazioni necessarie (`IssueCertificateGetCertificate`, e`ListPermissions`) affinché ACM possa rinnovare automaticamente i certificati.

```
$ 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
```

Utilizza il comando list-permissions per elencare le autorizzazioni delegate da una CA.

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

Utilizzate il comando delete-permission per revocare le autorizzazioni assegnate da una CA a un responsabile del servizio. AWS

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

Allega una politica per l'accesso tra più account

Quando l'amministratore della CA e l'emittente del certificato risiedono in AWS account diversi, l'amministratore della CA deve condividere l'accesso alla CA. A tale scopo, è necessario allegare alla CA una policy basata sulle risorse. La policy concede le autorizzazioni di rilascio a un principale specifico, che può essere il proprietario di un AWS account, un utente IAM, un ID o l'ID di un'unità organizzativa. AWS Organizations

Un amministratore CA può allegare e gestire le policy nei seguenti modi:

- Nella console di gestione, utilizzando AWS Resource Access Manager (RAM), che è un metodo standard per la condivisione di AWS risorse tra account. Quando si condivide una risorsa CA AWS RAM con un responsabile di un altro account, la politica basata sulle risorse richiesta viene allegata automaticamente alla CA. Per ulteriori informazioni sulla RAM, consulta la Guida per l'AWS RAM utente.

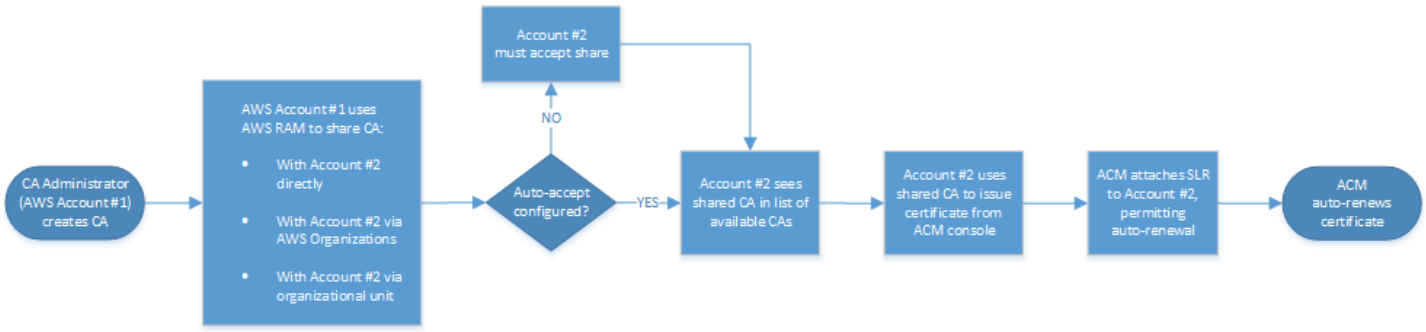
  > **Note**
  >
  > È possibile aprire facilmente la console RAM scegliendo una CA e quindi scegliendo Azioni, Gestisci condivisioni di risorse.

- A livello di programmazione, utilizzando le API PutPolicyPCA, e. GetPolicyDeletePolicy
- Manualmente, utilizzando i comandi PCA put-policy, get-policy e delete-policy in. AWS CLI

Solo il metodo della console richiede l'accesso alla RAM.

Caso 1 su più account: emissione di un certificato gestito dalla console

In questo caso, l'amministratore della CA utilizza AWS Resource Access Manager (AWS RAM) per condividere l'accesso CA con un altro AWS account, il che consente a tale account di emettere certificati ACM gestiti. Il diagramma mostra che AWS RAM è possibile condividere la CA direttamente con l'account o indirettamente tramite un AWS Organizations ID di cui l'account è membro.

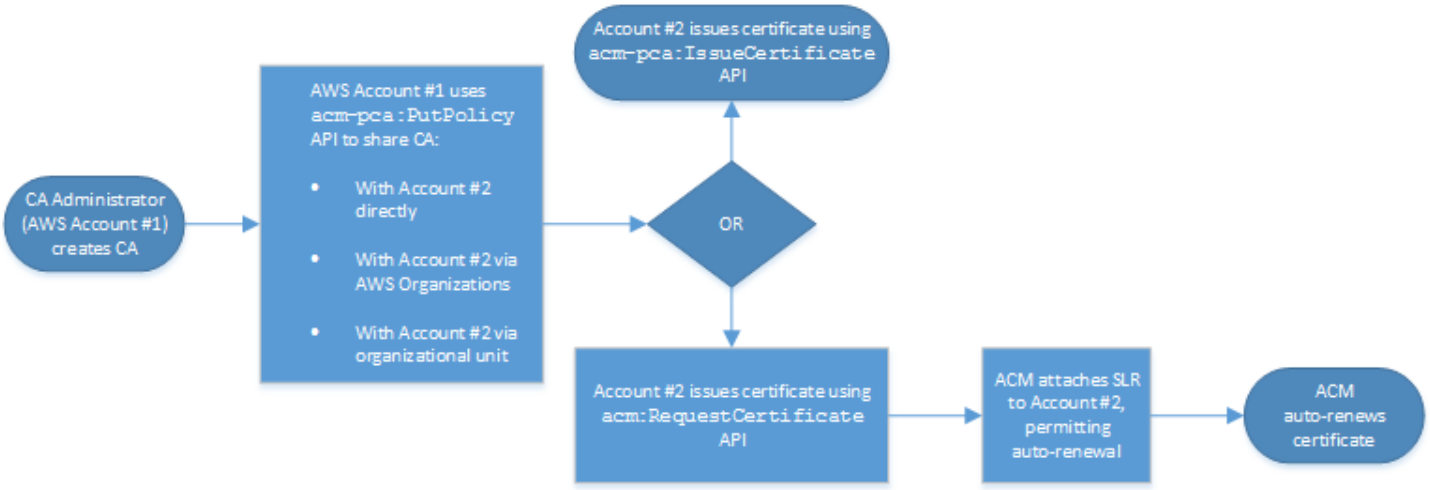


Dopo che RAM ha condiviso una risorsa AWS Organizations, il destinatario principale deve accettare la risorsa affinché questa abbia effetto. Il destinatario può AWS Organizations configurare l'accettazione automatica delle azioni offerte.

> **Note**
>
> L'account del destinatario è responsabile della configurazione del rinnovo automatico in ACM. In genere, alla prima volta che viene utilizzata una CA condivisa, ACM installa un ruolo collegato al servizio che consente di effettuare chiamate automatiche ai certificati. CA privata AWS Se questa operazione fallisce (di solito a causa di un'autorizzazione mancante), i certificati della CA non vengono rinnovati automaticamente. Solo l'utente ACM può risolvere il problema, non l'amministratore della CA. Per ulteriori informazioni, vedere Using a Service Linked Role (SLR) con ACM.

Caso 2 per più account: emissione di certificati gestiti e non gestiti utilizzando l'API o la CLI

Questo secondo caso illustra le opzioni di condivisione ed emissione possibili utilizzando l'API and. AWS Certificate Manager CA privata AWS Tutte queste operazioni possono essere eseguite anche utilizzando i comandi corrispondenti AWS CLI .



Poiché le operazioni API vengono utilizzate direttamente in questo esempio, l'emittente del certificato può scegliere tra due operazioni API per emettere un certificato. L'azione dell'API PCA `IssueCertificate` produce un certificato non gestito che non verrà rinnovato automaticamente e deve essere esportato e installato manualmente. L'azione API ACM RequestCertificateproduce un certificato gestito che può essere facilmente installato sui servizi integrati ACM e si rinnova automaticamente.

> **Note**
>
> L'account del destinatario è responsabile della configurazione del rinnovo automatico in ACM. In genere, alla prima volta che viene utilizzata una CA condivisa, ACM installa un ruolo collegato al servizio che consente di effettuare chiamate automatiche ai certificati. CA privata AWS Se questa operazione fallisce (di solito a causa di un'autorizzazione mancante), i certificati della CA non si rinnoveranno automaticamente e solo l'utente ACM può risolvere il problema, non l'amministratore della CA. Per ulteriori informazioni, vedere Using a Service Linked Role (SLR) con ACM.

Elenco delle CA private

Puoi usare la CA privata AWS console o AWS CLI elencare le CA private che possiedi o a cui hai accesso.

Per elencare le CA disponibili utilizzando la console

1. Accedi al tuo AWS account e apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home.
2. Consulta le informazioni nell'elenco delle autorità di certificazione private. È possibile navigare tra più pagine di CA utilizzando i numeri di pagina in alto a destra. Ogni CA occupa una riga con alcune o tutte le seguenti colonne visualizzate per ognuna di esse:

- Oggetto: riepilogo delle informazioni sui nomi distinti della CA.
- Id: identificatore univoco esadecimale a 32 byte della CA.
- Status: stato della CA. I valori possibili sono Creating, Pending certificate, Active, Deleted, Disabled, Expired e Failed.
- Tipo: il tipo di CA. I valori possibili sono Root e Subordinate.
- Modalità: la modalità della CA. I valori possibili sono General-purpose (emette certificati che possono essere configurati con qualsiasi data di scadenza) e Shortlived certificate (emette certificati con un periodo di validità massimo di sette giorni). Un breve periodo di validità può sostituire in alcuni casi un meccanismo di revoca. L'impostazione predefinita è General-purpose.
- Proprietario: l' AWS account che possiede la CA. Può trattarsi del tuo account o di un account che ti ha delegato le autorizzazioni di gestione della CA.
- Algoritmo chiave: l'algoritmo a chiave pubblica supportato dalla CA. I valori possibili sono RSA_2048, RSA_4096, EC_Prime256v1 e EC_SECP384R1.
- Algoritmo di firma: l'algoritmo utilizzato dalla CA per firmare le richieste di certificati. (Da non confondere con il `SigningAlgorithm` parametro utilizzato per firmare i certificati al momento dell'emissione). I valori possibili sono SHA256WITHECDSA, SHA384WITHECDSA, SHA512WITHECDSA, SHA256WITHRSA, SHA384WITHRSA e SHA512WITHRSA.

> **Note**
>
> Puoi personalizzare le colonne che desideri visualizzare, così come altre impostazioni, scegliendo l'icona delle impostazioni nell'angolo in alto a destra della console.

Per elencare le CA disponibili, utilizzare AWS CLI

Utilizzate il list-certificate-authoritiescomando per elencare le CA disponibili, come illustrato nell'esempio seguente:

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

Questo comando restituisce informazioni simili alle seguenti:

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

Visualizzazione di una CA privata

È possibile utilizzare la console ACM o la AWS CLI per visualizzare metadati dettagliati su una CA privata e modificare diversi valori in base alle esigenze. Per informazioni dettagliate sull'aggiornamento delle CA, consulta. Aggiornamento della CA privata

Per visualizzare i dettagli della CA nella console

1. Accedi al tuo AWS account e apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home.
2. Consulta l'elenco delle autorità di certificazione private. È possibile navigare tra più pagine di CA utilizzando i numeri di pagina in alto a destra.
3. Per mostrare i metadati dettagliati per una CA elencata, scegli il pulsante di opzione accanto alla CA che desideri controllare. Si apre un riquadro dei dettagli con le seguenti visualizzazioni a schede:

   - Scheda Oggetto: informazioni sul nome distinto della CA. Per ulteriori informazioni, consulta Opzioni relative al nome distinto del soggetto. I campi visualizzati includono:
     - Oggetto: riepilogo dei campi di informazioni sul nome forniti
     - Organizzazione (O): ad esempio, il nome di una società
     - Unità organizzativa (OU): ad esempio, una divisione all'interno di un'azienda
     - Nome del paese (C): un codice del paese di due lettere
     - Nome dello stato o della provincia: nome completo di uno stato o di una provincia
     - Nome località: il nome di una città
     - Nome comune (CN): una stringa leggibile dall'uomo per identificare la CA.
   - Scheda certificato CA: informazioni sulla validità del certificato CA
     - Valido fino a: la data e l'ora di validità del certificato CA
     - Scade tra: il numero di giorni che mancano alla scadenza
   - Scheda di configurazione della revoca: le opzioni attualmente selezionate per la revoca dei certificati. Scegli Modifica per aggiornare.
     - Distribuzione dell'elenco di revoca dei certificati (CRL): stato di Abilitato o Disabilitato
     - Online Certificate Status Protocol (OCSP) : stato di abilitato o disabilitato
   - Scheda Autorizzazioni: la selezione attuale delle autorizzazioni per il rinnovo dei certificati per questa CA tramite AWS Certificate Manager (ACM). Scegli Modifica per aggiornare.
   - Autorizzazione ACM per i rinnovi: stato di autorizzato o non autorizzato
   - Scheda Tag: la tua attuale assegnazione di etichette personalizzabili per questa CA. Scegli l'opzione Gestisci tag da aggiornare.

- Scheda Condivisioni di risorse: l'attuale assegnazione delle condivisioni di risorse per questa CA tramite AWS Resource Access Manager (RAM). Scegli Gestisci condivisioni di risorse da aggiornare.
  - Nome: nome della condivisione di risorse
  - Stato: stato della condivisione di risorse

4. Scegli il campo ID della CA che desideri controllare per aprire il riquadro Generale. L'identificatore univoco esadecimale a 32 byte della CA viene visualizzato nella parte superiore. Il riquadro fornisce le seguenti informazioni aggiuntive:

   - Stato: stato della CA. I valori possibili sono Creating, Pending certificate, Active, Deleted, Disabled, Expired e Failed.
   - ARN: il nome della risorsa Amazon per la CA.
   - Proprietario: l' AWS account che possiede la CA. Può trattarsi del tuo account (Self) o di un account che ti ha delegato le autorizzazioni di gestione della CA.
   - Tipo di CA: il tipo di CA. I valori possibili sono Root e Subordinate.
   - Creato in: la data e l'ora in cui è stata creata la CA.
   - Data di scadenza: data e ora di scadenza del certificato CA.
   - Modalità: la modalità della CA. I valori possibili sono General-purpose (certificati che possono essere configurati con qualsiasi data di scadenza) e Shortlived Certificate (certificati con un periodo di validità massimo di sette giorni). In alcuni casi un breve periodo di validità può sostituire un meccanismo di revoca. L'impostazione predefinita è General-purpose.
   - Algoritmo chiave: l'algoritmo a chiave pubblica supportato dalla CA. I valori possibili sono RSA 2048, RSA 4096, ECDSA P2567 ed ECDSA P384.
   - Algoritmo di firma: l'algoritmo utilizzato dalla CA per firmare le richieste di certificati. (Da non confondere con il `SigningAlgorithm` parametro utilizzato per firmare i certificati al momento dell'emissione). I valori possibili sono SHA256 ECDSA, SHA384 ECDSA, SHA512 ECDSA, SHA256 RSA, SHA384 RSA e SHA512 RSA
   - Standard di sicurezza dello storage chiave: livello di conformità agli standard federali di elaborazione delle informazioni. I valori possibili sono FIPS 140-2 livello 3 o superiore e FIPS 140-2 livello 3 o superiore. Questo parametro varia in base alla regione. AWS

Per visualizzare e modificare i dettagli della CA utilizzando AWS CLI

Utilizzare il describe-certificate-authoritycomando in AWS CLI per visualizzare i dettagli su una CA, come illustrato nel comando seguente:

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

Questo comando restituisce informazioni simili alle seguenti:

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

Per informazioni sull'aggiornamento di una CA privata dalla riga di comando, vedereAggiornamento di una CA (CLI).

Gestione dei tag per la tua CA privata

I tag sono parole o frasi che fungono da metadati per l'identificazione e l'organizzazione delle risorse AWS . Ciascun tag è formato da una chiave e da un valore, Puoi utilizzare la CA privata AWS

console, AWS Command Line Interface (AWS CLI) o l'API PCA per aggiungere, visualizzare o rimuovere tag per CA private.

Puoi aggiungere o rimuovere tag personalizzati per la tua CA privata in qualsiasi momento. Ad esempio, è possibile etichettare le CA private con coppie chiave-valore come `Environment=Prod` o `Environment=Beta` per identificare a quale ambiente è destinata la CA. Per ulteriori informazioni, consulta Creare una CA privata.

> **Note**
>
> Per allegare tag a una CA privata durante la procedura di creazione, un amministratore CA deve prima associare una policy IAM in linea all'`CreateCertificateAuthority`azione e consentire esplicitamente l'etichettatura. Per ulteriori informazioni, consulta Tag-on-create: Allegare tag a una CA al momento della creazione.

Anche altre AWS risorse supportano il tagging. È possibile assegnare lo stesso tag a risorse diverse per indicare che tali risorse sono correlate. Ad esempio, puoi assegnare un tag come `Website=example.com` alla tua CA, al sistema di bilanciamento del carico Elastic Load Balancing e ad altre risorse correlate. Per ulteriori informazioni sull'etichettatura AWS delle risorse, consulta Tagging your Amazon EC2 Resources nella Amazon EC2 User Guide for Linux Instances.

Le seguenti restrizioni di base si applicano ai tag: CA privata AWS

- Il numero massimo di tag per CA privata è 50.
- La lunghezza massima di una chiave di tag è 128 caratteri.
- La lunghezza massima di un valore di tag è 256 caratteri.
- La chiave tag e il valore possono contenere i seguenti caratteri: A-Z, a-z e.:+= @_%- (trattino).
- Per le chiavi e i valori dei tag viene fatta la distinzione tra maiuscole e minuscole.
- I prefissi `aws:` e `rds:` sono riservati all'uso da parte di AWS : non è possibile aggiungere, modificare o eliminare tag la cui chiave inizia con `aws:` o `rds:`. Tag predefiniti che iniziano con `aws:` e `rds:` non vengono conteggiati ai fini della tags-per-resource quota.
- Se prevedi di utilizzare il tuo schema di tagging su più servizi e risorse, ricorda che altri servizi potrebbero avere restrizioni diverse per i caratteri consentiti. Consultare la documentazione per quel servizio.
- CA privata AWS i tag non sono disponibili per l'uso in Resource Groups e Tag Editor in AWS Management Console.

Puoi applicare tag a una CA privata dalla Console CA privata AWS, dall'AWS Command Line Interface (AWS CLI) o dall'API CA privata AWS:

Per applicare tag a una CA privata (console)

1. Accedi al tuo AWS account e apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home.
2. Nella pagina Autorità di certificazione private, scegli la tua CA privata dall'elenco.
3. Nell'area dei dettagli sotto l'elenco, scegli la scheda Tag. Viene visualizzato un elenco di tag esistenti.
4. Scegliere Gestisci tag.
5. Scegliere Aggiungi nuovo tag.
6. Digitare una chiave e una coppia di valori.
7. Selezionare Salva.

Per applicare tag a una CA privata (AWS CLI)

Usa il tag-certificate-authoritycomando per aggiungere tag alla tua CA privata.

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

Utilizza il comando list-tags per elencare i tag per una CA privata.

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

Usa il untag-certificate-authoritycomando per rimuovere i tag da una CA privata.

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

Aggiornamento della CA privata

È possibile aggiornare lo stato di una CA privata o modificarne la configurazione di revoca dopo averla creata. Questo argomento fornisce dettagli sullo stato della CA e sul ciclo di vita della CA, oltre a esempi di aggiornamenti della console e della CLI delle CA.

Aggiornamento dello stato della CA

Lo stato di una CA gestita da CA privata AWS risulta da un'azione dell'utente o, in alcuni casi, da un'azione di servizio. Ad esempio, lo stato di una CA cambia quando scade. Le opzioni di stato disponibili per gli amministratori della CA variano a seconda dello stato corrente della CA.

CA privata AWS può riportare i seguenti valori di stato. La tabella mostra le funzionalità CA disponibili in ogni stato.

> **Note**
>
> Per tutti i valori di stato tranne `DELETED` e`FAILED`, ti viene addebitata la CA.

Stato	Emetti certificati	Convalidi certificati con OCSP	Genera CRL	Genera audit	È possibile aggiornare il certificato CA	I certificati possono essere revocati	Ti viene addebitata la CA
`CREATING`— La CA è in fase di creazione.	No	No	No	No	No	No	Sì
`PENDING_CERTIFICATE` — La CA è stata creata e necessita di un certificato per essere operativa. *	No	No	No	No	No	No	Sì

Stato	Emetti certificati	Convali i certificati con OCSP	Genera CRL	Genera audit	È possibile aggiornare il certificato CA	I certificati possono essere revocati	Ti viene addebitata la CA
`ACTIVE`	Sì	Sì	Sì	Sì	Sì	Sì	Sì
`DISABLED`— La CA è stata disattivata manualmente.	No	Sì	Sì	Sì	No	Sì	Sì
`EXPIRED`— Il certificato CA è scaduto. **	No	No	No	No	Sì	No	Sì
`FAILED`	L'`CreateCertificateAuthority` azione non è riuscita. Ciò può verificarsi a causa di un'interruzione della rete, di un AWS errore del backend o di altri errori. Impossibile ripristinare una CA non riuscita. Eliminare la CA e crearne una nuova.						No
`DELETED`	La CA rientra nel periodo di ripristino, che può avere una durata di 7-30 giorni. Dopo questo periodo, viene eliminato definitivamente. • Se si chiama l'API `RestoreCertificateAuthority` su una CA con stato `DELETED` e un certificato scaduto, la CA verrà impostata su `EXPIRED`. • Per ulteriori informazioni sull'eliminazione di una CA, consulta Eliminazione della CA privata.						No

* Per completare l'attivazione, è necessario generare una CSR, ottenere un certificato CA firmato da una CA e importare il certificato in. CA privata AWS La CSR può essere inviata alla nuova CA (per la firma automatica) o a una CA principale o subordinata locale. Per ulteriori informazioni, consulta Creazione e installazione del certificato CA.

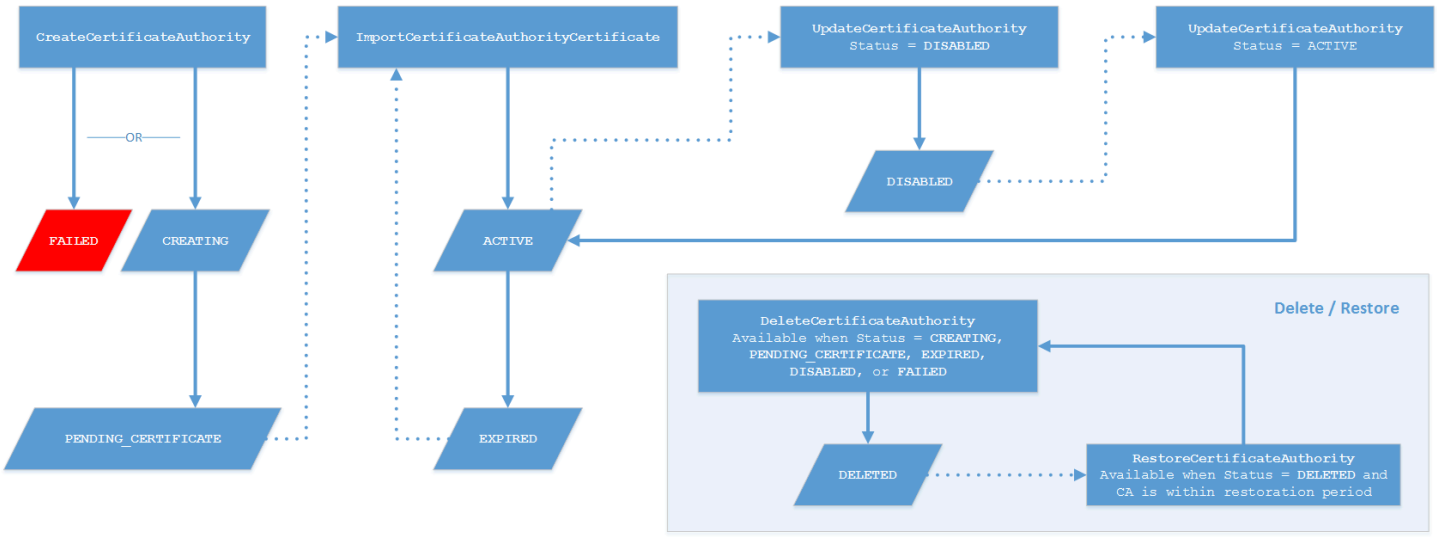
** Non è possibile modificare direttamente lo stato di una CA scaduta. Se si importa un nuovo certificato per la CA, CA privata AWS reimposta lo stato a `ACTIVE` meno che non fosse impostato `DISABLED` prima della scadenza del certificato.

Considerazioni aggiuntive sui certificati CA scaduti:

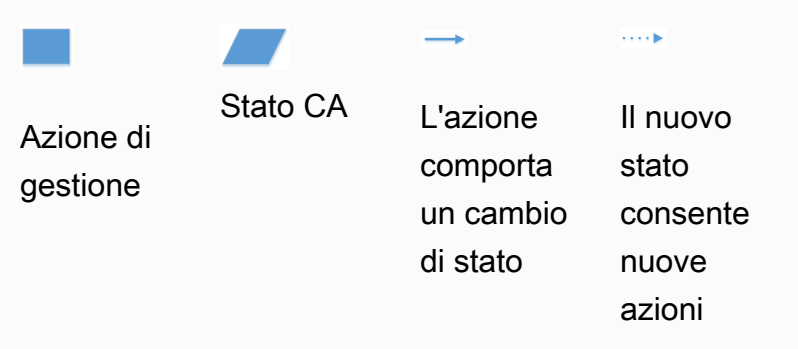
- I certificati CA non vengono rinnovati automaticamente. Per informazioni sull'automazione del rinnovo tramite AWS Certificate Manager. Assegna le autorizzazioni per il rinnovo dei certificati ad ACM
- Se si tenta di emettere un nuovo certificato con una CA scaduta, l'API `IssueCertificate` restituisce `InvalidStateException`. Una CA root scaduta deve autofirmare un nuovo certificato emesso da una CA root prima di poter emettere nuovi certificati subordinati.
- The `ListCertificateAuthorities` e le API `DescribeCertificateAuthority` restituiscono lo stato `EXPIRED` se il certificato emesso da una CA è scaduto, indipendentemente dal fatto che lo stato CA sia impostato su `ACTIVE` o `DISABLED`. Tuttavia, se la CA scaduta è stata impostata su `DELETED`, lo stato restituito è `DELETED`.
- L'API `UpdateCertificateAuthority` non può aggiornare lo stato di una CA scaduta.
- L'`RevokeCertificate`API non può essere utilizzata per revocare alcun certificato scaduto, incluso un certificato CA.

Stato della CA e ciclo di vita della CA

Il diagramma seguente illustra il ciclo di vita della CA come interazione delle azioni di gestione con lo stato della CA.



Chiave del diagramma



Nella parte superiore del diagramma, le operazioni di gestione vengono applicate tramite la console CA privata AWS , l'interfaccia a riga di comando o l'API. Le azioni intraprese dalla CA attraverso la creazione, l'attivazione, la scadenza e il rinnovo. Lo stato della CA cambia in risposta (come mostrato dalle linee continue) ad azioni manuali o aggiornamenti automatici. Nella maggior parte dei casi, un nuovo stato porta a una nuova azione possibile (mostrata da una linea tratteggiata) che l'amministratore della CA può applicare. L'inserto in basso a destra mostra i possibili valori di stato che consentono le azioni di eliminazione e ripristino.

Aggiornamento di una CA (console)

Le seguenti procedure mostrano come aggiornare le configurazioni CA esistenti utilizzando. AWS Management Console

Aggiornare lo stato della CA (console)

In questo esempio, lo stato di una CA abilitata viene modificato in disabilitato.

Per aggiornare lo stato di una CA

1. Accedi al tuo AWS account e apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home
2. Nella pagina Autorità di certificazione private, scegli una CA privata attualmente attiva dall'elenco.
3. Nel menu Azioni, scegli Disabilita per disabilitare la CA privata.

Aggiornamento della configurazione di revoca di una CA (console)

È possibile aggiornare la configurazione di revoca per la CA privata, ad esempio aggiungendo o rimuovendo il supporto OCSP o CRL o modificandone le impostazioni.

> **Note**
>
> Le modifiche alla configurazione di revoca di una CA non influiscono sui certificati già emessi. Affinché la revoca gestita funzioni, i certificati precedenti devono essere riemessi.

Per OCSP, è possibile modificare le seguenti impostazioni:

- Abilita o disabilita OCSP.
- Abilita o disabilita un nome di dominio completo (FQDN) OCSP personalizzato.
- Cambia il nome di dominio completo.

Per un CRL, puoi modificare una delle seguenti impostazioni:

- Se la CA privata genera un elenco di revoche di certificati (CRL)
- Il numero di giorni prima della scadenza di un CRL. Tieni presente che CA privata AWS inizia il tentativo di rigenerare il CRL alla metà del numero di giorni specificato.
- Il nome del bucket Amazon S3 in cui è salvato il tuo CRL.
- Un alias per nascondere il nome del tuo bucket Amazon S3 dalla visualizzazione pubblica.

> **Important**
>
> La modifica di uno qualsiasi dei parametri precedenti può avere effetti negativi. Gli esempi includono la disabilitazione della generazione di CRL, la modifica del periodo di validità o la modifica del bucket S3 dopo aver messo in produzione la CA privata. Tali modifiche possono interrompere i certificati esistenti che dipendono dal CRL e dalla configurazione CRL corrente. La modifica dell'alias può essere effettuata in modo sicuro finché l'alias precedente rimane collegato al bucket corretto.

Per aggiornare le impostazioni di revoca

1. Accedi al tuo AWS account e apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home.
2. Nella pagina Autorità di certificazione private, scegli una CA privata dall'elenco. Si apre il pannello dei dettagli per la CA.
3. Scegli la scheda Configurazione della revoca, quindi scegli Modifica.
4. In Opzioni di revoca del certificato, vengono visualizzate due opzioni:

   - Attiva la distribuzione CRL
   - Attiva OCSP

   È possibile configurare uno, nessuno dei due o entrambi questi meccanismi di revoca per la CA. Sebbene facoltativa, la revoca gestita è consigliata come best practice. Prima di completare questo passaggio, consulta Impostazione di un metodo di revoca dei certificati le informazioni sui vantaggi di ciascun metodo, sulla configurazione preliminare che potrebbe essere richiesta e sulle funzionalità di revoca aggiuntive.

Per configurare un CRL

1. Seleziona Attiva la distribuzione CRL.
2. Per creare un bucket Amazon S3 per le voci del CRL, seleziona Crea un nuovo bucket S3. Fornisci un nome univoco per il bucket. (Non è necessario includere il percorso del bucket.) Altrimenti, lascia questa opzione deselezionata e scegli un bucket esistente dall'elenco dei nomi dei bucket S3.

   Se crei un nuovo bucket, CA privata AWS crea e allega la politica di accesso richiesta. Se decidi di utilizzare un bucket esistente, devi allegare una policy di accesso prima di poter iniziare a generare CRL. Utilizza uno dei modelli di policy descritti in. Politiche di accesso per i CRL in Amazon S3 Per informazioni su come allegare una policy, consulta Aggiungere una bucket policy utilizzando la console Amazon S3.

   > **Note**
   >
   > Quando usi la CA privata AWS console, un tentativo di creare una CA fallisce se si verificano entrambe le seguenti condizioni:

- Stai applicando le impostazioni Block Public Access sul tuo bucket o account Amazon S3.
- Hai chiesto CA privata AWS di creare automaticamente un bucket Amazon S3.

In questa situazione, la console tenta, per impostazione predefinita, di creare un bucket accessibile pubblicamente e Amazon S3 rifiuta questa azione. Controlla le impostazioni di Amazon S3 se ciò si verifica. Per ulteriori informazioni, consulta Bloccare l'accesso pubblico allo storage Amazon S3.

3. Espandere Avanzate per ulteriori opzioni di configurazione.
   - Aggiungi un nome CRL personalizzato per creare un alias per il tuo bucket Amazon S3. Questo nome è contenuto nei certificati emessi dalla CA nell'estensione «CRL Distribution Points» definita da RFC 5280.
   - Digitare il numero di giorni per i quali il CRL rimarrà valido. Il valore predefinito è 7 giorni. Per i CRL online, è comune un periodo di validità di 2-7 giorni. CA privata AWS tenta di rigenerare il CRL a metà del periodo specificato.
4. Al termine, scegli Salva le modifiche.

Per configurare OCSP

1. Nella pagina di revoca del certificato, scegli Attiva OCSP.
2. (Facoltativo) Nel campo Endpoint OCSP personalizzato, fornisci un nome di dominio completo (FQDN) per l'endpoint OCSP.

   Quando fornisci un FQDN in questo campo, CA privata AWS inserisce il nome di dominio completo nell'estensione Authority Information Access di ciascun certificato emesso al posto dell'URL predefinito per il risponditore OCSP. AWS Quando un endpoint riceve un certificato contenente l'FQDN personalizzato, richiede a tale indirizzo una risposta OCSP. Affinché questo meccanismo funzioni, è necessario eseguire due azioni aggiuntive:
   - Utilizzate un server proxy per inoltrare il traffico che arriva al vostro FQDN personalizzato al risponditore AWS OCSP.
   - Aggiungi un record CNAME corrispondente al tuo database DNS.

> **Tip**
>
> Per ulteriori informazioni sull'implementazione di una soluzione OCSP completa utilizzando un CNAME personalizzato, vedere. Configurazione di un URL personalizzato per OCSP CA privata AWS

Ad esempio, ecco un record CNAME per OCSP personalizzato come apparirebbe in Amazon Route 53.

Nome record	Type	Policy di routing	Differenziatore	Valore/instradamento traffico a
alternative.example.com	CNAME	Semplice	-	proxy.example.com

> **Note**
>
> Il valore del CNAME non deve includere un prefisso di protocollo come «http://» o «https://».

3. Al termine, scegli Salva le modifiche.

Aggiornamento di una CA (CLI)

Le procedure seguenti mostrano come aggiornare la configurazione dello stato e della revoca di una CA esistente utilizzando. AWS CLI

> **Note**
>
> Le modifiche alla configurazione di revoca di una CA non influiscono sui certificati già emessi. Affinché la revoca gestita funzioni, i certificati precedenti devono essere riemessi.

Per aggiornare lo stato della tua CA privata ()AWS CLI

Utilizza il comando update-certificate-authority.

Ciò è utile quando si dispone di una CA esistente con uno stato su DISABLED cui si desidera impostareACTIVE. Per iniziare, confermate lo stato iniziale della CA con il seguente comando.

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

Il risultato è un output simile al seguente.

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

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

Il comando seguente imposta lo stato della CA privata su`ACTIVE`. Ciò è possibile solo se sulla CA è installato un certificato valido.

```
$ 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"
```

Controlla il nuovo stato della 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
```

Lo stato ora appare come`ACTIVE`.

```
{
    "CertificateAuthority": {
        "Arn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566",
        "CreatedAt": "2021-03-05T14:24:12.867000-08:00",
        "LastStateChangeAt": "2021-03-08T13:23:09.352000-08:00",
        "Type": "ROOT",
        "Serial": "serial_number",
        "Status": "ACTIVE",
        "NotBefore": "2021-03-08T07:46:27-08:00",
        "NotAfter": "2022-03-08T08:46:27-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
```

```
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "CustomCname": "alternative.example.com",
                "S3BucketName": "DOC-EXAMPLE-BUCKET1"
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

In alcuni casi, potresti avere una CA attiva senza alcun meccanismo di revoca configurato. Se si desidera iniziare a utilizzare un elenco di revoca dei certificati (CRL), utilizzare la procedura seguente.

Per aggiungere un CRL a una CA esistente ()AWS CLI

1. Utilizzate il seguente comando per controllare lo stato corrente della 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
```

   L'output conferma che la CA ha uno stato `ACTIVE` ma non è configurata per utilizzare un 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. Crea e salva un file con un nome tale `revoke_config.txt` da definire i parametri di configurazione CRL.

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

> **Note**
>
> Quando si aggiorna una CA di attestazione del dispositivo Matter per abilitare i CRL, è necessario configurarla in modo da omettere l'estensione CDP dai certificati emessi

per contribuire alla conformità allo standard Matter corrente. A tale scopo, definisci i parametri di configurazione CRL come illustrato di seguito:

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

3. Utilizzate il update-certificate-authoritycomando e il file di configurazione della revoca per aggiornare la 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. Controllate nuovamente lo stato della 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
```

L'output conferma che CA è ora configurata per utilizzare un CRL.

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

```
        "NotAfter": "2022-03-08T14:46:50-08:00",
        "CertificateAuthorityConfiguration": {
            "KeyAlgorithm": "RSA_2048",
            "SigningAlgorithm": "SHA256WITHRSA",
            "Subject": {
                "Country": "US",
                "Organization": "Example Corp",
                "OrganizationalUnit": "Sales",
                "State": "WA",
                "CommonName": "www.example.com",
                "Locality": "Seattle"
            }
        },
        "RevocationConfiguration": {
            "CrlConfiguration": {
                "Enabled": true,
                "ExpirationInDays": 7,
                "S3BucketName": "DOC-EXAMPLE-BUCKET1",
            },
            "OcspConfiguration": {
                "Enabled": false
            }
        }
    }
}
```

In alcuni casi, è possibile aggiungere il supporto per la revoca OCSP anziché abilitare un CRL come nella procedura precedente. In tal caso, utilizzare la procedura seguente.

Per aggiungere il supporto OCSP a una CA esistente ()AWS CLI

1. Crea e salva un file con un nome tale `revoke_config.txt` da definire i parametri OCSP.

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

2. Utilizza il update-certificate-authoritycomando e il file di configurazione della revoca per aggiornare la 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. Controllate nuovamente lo stato della 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
```

L'output conferma che CA è ora configurata per utilizzare 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**
>
> È inoltre possibile configurare il supporto CRL e OCSP su una CA.

Eliminazione della CA privata

È possibile eliminare una CA privata da AWS Management Console o AWS CLI definitivamente. È possibile eliminare una CA per sostituirla, ad esempio, per sostituirla con una nuova CA che dispone di una nuova chiave privata. Per eliminare una CA in modo sicuro, attenersi alla seguente procedura:

1. Creare la CA di sostituzione.
2. Una volta che la nuova CA privata è in produzione, disabilitare quella precedente ma non eliminarla immediatamente.
3. Mantenere la vecchia CA disabilitata finché non sono scaduti tutti i certificati che ha emesso.
4. Eliminare la vecchia CA.

CA privata AWS non verifica che tutti i certificati emessi siano scaduti prima di elaborare una richiesta di eliminazione. Puoi generare un report di audit per determinare quali sono i certificati scaduti. Mentre la CA è disabilitata, è possibile revocare i certificati ma non è possibile emetterne di nuovi.

Se devi eliminare una CA privata prima che tutti i certificati emessi siano scaduti, ti consigliamo di revocare anche il certificato CA. Il certificato CA verrà elencato nel CRL della CA principale e la CA privata sarà considerata non attendibile dai client.

> **Important**
>
> Una CA privata può essere eliminata se è nello stato `PENDING_CERTIFICATE`, `CREATING`, `EXPIRED`, `DISABLED` o `FAILED`. Per eliminare una CA nello stato `ACTIVE`, devi prima disabilitarla o la richiesta di eliminazione genererà un'eccezione. Se si elimina una CA privata `DISABLED` nello stato `PENDING_CERTIFICATE` o, è possibile impostare la durata del periodo

di ripristino su 7-30 giorni, con 30 come impostazione predefinita. Durante questo periodo, lo stato è impostato su `DELETED` e la CA è ripristinabile. Una CA privata eliminata `FAILED` nello stato `CREATING` o non ha un periodo di ripristino assegnato e non può essere ripristinata. Per ulteriori informazioni, consulta Ripristino di una CA privata.
Non è previsto alcun addebito per una CA privata dopo che è stata eliminata. Tuttavia, se una CA privata viene ripristinata, ti verrà addebitata per l'intervallo di tempo compreso tra l'eliminazione e il ripristino. Per ulteriori informazioni, consulta Prezzi.

Per eliminare una CA privata (console)

1. Accedi al tuo AWS account e apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home.
2. Nella pagina Autorità di certificazione private, scegli la tua CA privata dall'elenco.
3. Se la tua CA si trova nello `ACTIVE` stato, devi prima disabilitarla. Dal menu Actions (Operazioni) scegliere Disable (Disabilita). Quando richiesto, scegli Comprendo il rischio, continua.
4. Per una CA che non si trova nello `ACTIVE` stato, scegli Azioni, Elimina.
5. Se la CA si trova nello `PENDING_CERTIFICATE` stato`DISABLED`, o`EXPIRED`,,,,,, la pagina Elimina CA consente di specificare un periodo di ripristino di 7-30 giorni. Se la CA privata non si trova in uno di questi stati, non può essere ripristinata in un secondo momento e l'eliminazione è permanente.
6. Scegli Elimina.
7. Se si è certi di eliminare la CA privata, scegliere Permanently delete (Elimina definitivamente) quando richiesto. Lo stato della CA privata cambia in `DELETED`. Tuttavia, è possibile ripristinare la CA privata prima della fine del periodo di ripristino. Per verificare il periodo di ripristino di una CA privata nello `DELETED` stato, chiama l'operazione DescribeCerticateAuthorityo ListCertificateAuthoritiesAPI.

Per eliminare una CA privata (AWS CLI)

Utilizzate il delete-certificate-authoritycomando per eliminare una CA privata.

```
$ 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
```

Ripristino di una CA privata

Puoi ripristinare una CA privata eliminata finché rimane all'interno del periodo di ripristino specificato dopo l'eliminazione. Il periodo di ripristino è compreso tra 7 e 30 giorni. Una volta terminato tale periodo, la CA privata viene eliminata definitivamente. Per ulteriori informazioni, consulta Eliminazione della CA privata. Una CA privata eliminata definitivamente non può essere ripristinata.

> **Note**
>
> Non è previsto alcun addebito per una CA privata dopo che è stata eliminata. Tuttavia, se una CA privata viene ripristinata, ti verrà addebitata per l'intervallo di tempo compreso tra l'eliminazione e il ripristino. Per ulteriori informazioni, consulta Prezzi.

Ripristino di una CA (console) privata

È possibile utilizzare il AWS Management Console per ripristinare una CA privata.

Per ripristinare una CA privata (console)

1. Accedi al tuo AWS account e apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home.
2. Nella pagina Autorità di certificazione private, scegli la CA privata eliminata dall'elenco.
3. Dal menu Actions (Operazioni) scegli Restore (Ripristina).
4. Nella pagina Restore CA, scegli nuovamente Ripristina.
5. Se l'operazione ha esito positivo, lo stato della CA privata viene impostato per la pre-eliminazione. Scegli nuovamente Azioni, Abilita e Abilita per modificarne lo stato in`ACTIVE`. Se, al momento dell'eliminazione, la CA privata si trovava nello stato `PENDING_CERTIFICATE`, è necessario importare un certificato emesso da una CA nella CA privata prima di attivarla.

Ripristino di una CA privata ()AWS CLI

Utilizzate il restore-certificate-authoritycomando per ripristinare una CA privata eliminata che si trova nello `DELETED` stato. I seguenti passaggi costituiscono la procedura necessaria per eliminare, ripristinare e riattivare una CA privata.

Per eliminare, ripristinare e riattivare una CA privata (AWS CLI)

1. Eliminare la CA privata.

   Esegui il delete-certificate-authoritycomando per eliminare la CA privata. Se lo stato della CA privata è `DISABLED` o`PENDING_CERTIFICATE`, è possibile impostare il `--permanent-deletion-time-in-days` parametro per specificare il periodo di ripristino della CA privata compreso tra 7 e 30 giorni. Se non viene specificato un periodo di ripristino, l'impostazione predefinita è 30 giorni. In caso di esito positivo, questo comando imposta lo stato della CA privata su `DELETED`.

   > **Note**
   >
   > Per essere ripristinabile, al momento dell'eliminazione la CA deve trovarsi nello stato `DISABLED` o `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. Ripristinare la CA privata.

   Esegui il restore-certificate-authoritycomando per ripristinare la CA privata. È necessario eseguire il comando prima della fine del periodo di ripristino impostato con il comando delete-certificate-authority. Se l'operazione ha esito positivo, il comando imposta la CA privata sullo stato di pre-eliminazione.

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

3. Impostare la CA privata su `ACTIVE`.

   Esegui il update-certificate-authoritycomando per modificare lo stato della CA privata in`ACTIVE`.

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

```
    --status ACTIVE
```

Amministrazione dei certificati

Dopo aver creato e attivato un'autorità di certificazione (CA) privata e aver configurato l'accesso ad essa, tu o i tuoi utenti autorizzati potete eseguire le attività descritte in questa sezione. Se non hai ancora impostato le policy AWS Identity and Access Management (IAM) per la CA, puoi saperne di più sulla loro configurazione nella sezione Identity and Access Management di questa guida. Per informazioni sulla configurazione di CA Access in scenari con account singolo e tra account, vedere. Controllo dell'accesso a una CA privata

Argomenti

Emissione di certificati privati per entità finali

Con una CA privata, puoi richiedere certificati privati di entità finale a (ACM) oAWS Certificate Manager. CA privata AWS Le funzionalità di entrambi i servizi vengono confrontate nella tabella seguente.

Funzionalità	ACM	CA privata AWS
Emetti certificati per l'entità finale	✓ (utilizzando RequestCertificateo la console)	✓ (utilizzando IssueCertificate)
Associazione con sistemi di bilanciamento del carico e servizi connessi a Internet AWS	✓	Non supportato

Funzionalità	ACM	CA privata AWS
Rinnovo gestito dei certificati	✓	Supportato indirettamente tramite ACM
Supporto della console	✓	Non supportato
Supporto API	✓	✓
Supporto per CLI	✓	✓

Quando CA privata AWS crea un certificato, segue un modello che specifica il tipo di certificato e la lunghezza del percorso. Se non viene fornito alcun ARN del modello all'API o all'istruzione CLI che crea il certificato, per impostazione predefinita viene applicato il modello EndEntityCertificate/V1. Per ulteriori informazioni sui modelli di certificato disponibili, consulta Informazioni sui modelli di certificato.

Sebbene i certificati ACM siano progettati sulla base della fiducia pubblica, soddisfano le esigenze della CA privata AWS tua PKI privata. Di conseguenza, puoi configurare i certificati utilizzando l'CA privata AWSAPI e la CLI in modi non consentiti da ACM. Questi sono i seguenti:

- Creazione di un certificato con qualsiasi nome del soggetto.
- Utilizzando uno qualsiasi degli algoritmi di chiave privata e delle lunghezze di chiave supportati.
- Utilizzando uno qualsiasi degli algoritmi di firma supportati.
- Specificando un periodo di validità per la CA privata e i certificati privati.

Dopo aver creato un certificato TLS privato utilizzandoCA privata AWS, puoi importarlo in ACM e utilizzarlo con un servizio supportato. AWS

> **Note**
>
> I certificati creati con la procedura seguente, utilizzando il issue-certificate comando o con l'azione IssueCertificateAPI, non possono essere esportati direttamente per essere utilizzati all'esterno. AWS Tuttavia, puoi utilizzare la tua CA privata per firmare i certificati emessi tramite ACM e tali certificati possono essere esportati insieme alle relative chiavi segrete.

Per ulteriori informazioni, consulta Richiesta di un certificato privato ed Esportazione di un certificato privato nella Guida per l'utente ACM.

Emissione di un certificato standard () AWS CLI

Puoi utilizzare il comando CA privata AWS CLI issue-certificate o l'azione API IssueCertificateper richiedere un certificato di entità finale. Questo comando richiede l'Amazon Resource Name (ARN) della CA privata che si desidera utilizzare per emettere il certificato. È inoltre necessario generare una richiesta di firma del certificato (CSR) utilizzando un programma come OpenSSL.

Se utilizzi l'CA privata AWSAPI o AWS CLI emetti un certificato privato, il certificato non è gestito, il che significa che non puoi utilizzare la console ACM, l'ACM CLI o l'API ACM per visualizzarlo o esportarlo e il certificato non viene rinnovato automaticamente. Tuttavia, puoi utilizzare il comando PCA get-certificate per recuperare i dettagli del certificato e, se possiedi la CA, puoi creare un rapporto di controllo.

Considerazioni sulla creazione di certificati

- In conformità con RFC 5280, la lunghezza del nome di dominio (tecnicamente, il nome comune) fornito non può superare i 64 ottetti (caratteri), compresi i punti. Per aggiungere un nome di dominio più lungo, specificalo nel campo Nome alternativo del soggetto, che supporta nomi di lunghezza massima di 253 ottetti.
- Se utilizzate la AWS CLI versione 1.6.3 o successiva, utilizzate il prefisso `fileb://` quando specificate file di input con codifica base64 come CSR. Ciò garantisce che i dati vengano CA privata AWS analizzati correttamente.

Il seguente comando OpenSSL genera una CSR e una chiave privata per un certificato:

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

È possibile esaminare il contenuto della CSR nel modo seguente:

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

L'output risultante dovrebbe essere simile al seguente esempio abbreviato:

```
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
```

Il comando seguente crea un certificato. Poiché non viene specificato alcun modello, per impostazione predefinita viene emesso un certificato di entità finale di base.

```
$ aws acm-pca issue-certificate \
      --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
      --csr fileb://csr.pem \
      --signing-algorithm "SHA256WITHRSA" \
      --validity Value=365,Type="DAYS"
```

L'ARN del certificato emesso viene restituito:

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

```
}
```

> **Note**
>
> CA privata AWSrestituisce immediatamente un ARN con un numero di serie quando riceve il issue-certificate comando. Tuttavia, l'elaborazione dei certificati avviene in modo asincrono e può comunque fallire. In tal caso, anche un get-certificate comando che utilizza il nuovo ARN avrà esito negativo.

Emetti un certificato con un nome oggetto personalizzato utilizzando un modello ApiPassThrough

In questo esempio, viene emesso un certificato contenente elementi personalizzati del nome del soggetto. Oltre a fornire un CSR come quello in dotazioneEmissione di un certificato standard () AWS CLI, si passano due argomenti aggiuntivi al issue-certificate comando: l'ARN di un modello ApiPassThrough e un file di configurazione JSON che specifica gli attributi personalizzati e i relativi identificatori di oggetto (OID). Non è possibile utilizzarlo insieme `StandardAttributes` a. Tuttavia, è possibile passare OID standard come parte di`CustomAttributes`. `CustomAttributes` Gli OID predefiniti del nome dell'oggetto sono elencati nella tabella seguente (informazioni tratte da RFC 4519 e dal database di riferimento Global OID):

Nome del soggetto	Abbreviazione	ID dell'oggetto
countryName	c	2.5.4.6
commonName	cn	2.5.4.3
DNQualifier [qualificatore di nome distinto]		2.5.4.46
Generation Qualifier		2.5.4.44
givenName		2.5.4.42
iniziali		2.5.4.43
località	l	2.5.4.7

Nome del soggetto	Abbreviazione	ID dell'oggetto
Nome dell'organizzazione	o	2.5.4.10
organizationalUnitName	ou	2,5,4,11
pseudonimo		2.5.4.65
Numero di serie		2.5.4.5
st [stato]		2.5.4.8
cognome	sn	2.5.4.4
titolo		2,5,4,12
Componente del dominio	dc	0.9.2342.19200300.100.1.25
userid		0,9,2342,19200300,1001,1

Il file di configurazione di esempio contiene il codice seguente: `api_passthrough_config.txt`

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

Utilizzate il seguente comando per emettere il certificato:

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

L'ARN del certificato emesso viene restituito:

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

Recupera il certificato localmente come segue:

```
$ 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
```

Puoi controllare il contenuto del certificato usando OpenSSL:

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

> **Note**
>
> È anche possibile creare una CA privata che trasmetta attributi personalizzati a ciascun certificato emesso.

Emetti un certificato con estensioni personalizzate utilizzando un modello ApiPassthrough

In questo esempio, viene emesso un certificato che contiene estensioni personalizzate. Per questo è necessario passare tre argomenti al issue-certificate comando: l'ARN di un modello ApiPassThrough e un file di configurazione JSON che specifica le estensioni personalizzate e un CSR come quello mostrato in. Emissione di un certificato standard () AWS CLI

Il file di configurazione di esempio contiene il codice seguente: `api_passthrough_config.txt`

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

Il certificato personalizzato viene rilasciato come segue:

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

L'ARN del certificato emesso viene restituito:

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

Recupera il certificato localmente come segue:

```
$ 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
```

Puoi controllare il contenuto del certificato usando OpenSSL:

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

Recupero di un certificato privato

Puoi utilizzare l'API CA privata AWS e l'AWS CLI per rilasciare un certificato privato. In tal caso, è possibile utilizzare l'AWS CLI o l'API CA privata AWS per recuperare tale certificato. Se hai utilizzato ACM per creare la tua CA privata e per richiedere certificati, devi utilizzare ACM per esportare il certificato e la chiave privata crittografata. Per ulteriori informazioni, consulta Esportazione di un certificato privato.

Per recuperare un certificato di entità finale

Utilizzate il AWS CLI comando get-certificate per recuperare un certificato privato di entità finale. Puoi anche utilizzare l'operazione API. GetCertificate Ti consigliamo di formattare l'output con jq, un parser simile a sed.

> **Note**
>
> Se desideri revocare un certificato, puoi utilizzare il comando get-certificate per recuperare il numero di serie in formato esadecimale. È anche possibile creare un report di audit per recuperare il numero di serie esadecimale. Per ulteriori informazioni, consulta Utilizzo dei report di controllo con la tua CA privata.

```
$ 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'
```

Questo comando restituisce il certificato e la catena di certificati nel seguente formato standard.

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

Per recuperare un certificato CA

Puoi utilizzare l'API CA privata AWS e l'AWS CLI per recuperare il certificato dell'autorità di certificazione (CA) per la CA privata. Esegui il comando get-certificate-authority-certificate. Puoi anche chiamare l'operazione GetCertificateAuthorityCertificate. Consigliamo di formattare l'output con jq, un parser simile a sed.

```
$ aws acm-pca get-certificate-authority-certificate \
     --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/11223344-1234-1122-2233-112233445566 \
     | jq -r '.Certificate'
```

Questo comando restituisce il certificato CA nel seguente formato standard.

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

Elencare i certificati privati

Per elencare i tuoi certificati privati, genera un rapporto di controllo, recuperalo dal relativo bucket S3 e analizza il contenuto del rapporto secondo necessità. Per informazioni sulla creazione di report di CA privata AWS controllo, consulta. Utilizzo dei report di controllo con la tua CA privata

Per informazioni sul recupero di un oggetto da un bucket S3, consulta Downloading an object nella Amazon Simple Storage Service User Guide.

Gli esempi seguenti illustrano gli approcci per creare report di audit e analizzarli alla ricerca di dati utili. I risultati sono formattati in JSON e i dati vengono filtrati utilizzando jq, un parser simile a un sed.

1. Crea un rapporto di controllo.

Il comando seguente genera un rapporto di controllo per una CA specificata.

```
$ 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
```

In caso di successo, il comando restituisce l'ID e la posizione del nuovo rapporto di controllo.

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

2. Recupera e formatta un rapporto di controllo.

Questo comando recupera un rapporto di controllo, ne visualizza il contenuto in uno standard output e filtra i risultati per mostrare solo i certificati emessi a partire dal 01/12/2020 o dopo.

```
$ 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")'
```

Gli articoli restituiti sono simili ai seguenti:

```
{
   "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. Salva un rapporto di controllo localmente.

Se si desidera eseguire più interrogazioni, è consigliabile salvare un rapporto di controllo in un file locale.

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

Lo stesso filtro di prima produce lo stesso risultato:

```
$ 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. Interrogazione all'interno di un intervallo di date

È possibile eseguire una query per i certificati emessi entro un intervallo di date nel modo seguente:

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

Il contenuto filtrato viene visualizzato nello standard output:

```
{
   "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. Cerca i certificati seguendo un modello specificato.

Il comando seguente filtra il contenuto del report utilizzando un modello ARN:

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

L'output mostra i record dei certificati corrispondenti:

```
{
   "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. Filtro per i certificati revocati

Per trovare tutti i certificati revocati, usa il seguente comando:

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

Un certificato revocato viene visualizzato come segue:

```
{
   "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. Filtrare utilizzando un'espressione regolare.

Il comando seguente cerca i nomi dei soggetti che contengono la stringa «leaf»:

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

I record dei certificati corrispondenti vengono restituiti come segue:

```
{
   "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"
}
```

Esportazione di un certificato privato e della relativa chiave segreta

CA privata AWSnon può esportare direttamente un certificato privato che ha firmato ed emesso. Tuttavia, è possibile utilizzare AWS Certificate Manager per esportare tale certificato insieme alla relativa chiave segreta crittografata. Il certificato è quindi completamente portatile per essere distribuito ovunque nella tua PKI privata. Per ulteriori informazioni, consulta Esportazione di un certificato privato nella Guida per l'AWS Certificate Managerutente.

Come ulteriore vantaggio, AWS Certificate Manager offre il rinnovo gestito per i certificati privati emessi utilizzando la console ACM, l'`RequestCertificate`azione dell'API ACM o il request-certificate comando nella sezione ACM del. AWS CLI Per ulteriori informazioni sui rinnovi, consulta Rinnovo dei certificati in una PKI privata.

Revoca di un certificato privato

È possibile revocare un CA privata AWS certificato utilizzando il comando revoke-certificate o l'azione APIAWS CLI. RevokeCertificate Potrebbe essere necessario revocare un certificato prima della scadenza pianificata se, ad esempio, la chiave segreta è compromessa o il dominio associato non è valido. Affinché la revoca sia effettiva, il client che utilizza il certificato deve poter verificare lo stato della revoca ogni volta che tenta di creare una connessione di rete sicura.

CA privata AWSfornisce due meccanismi completamente gestiti per supportare il controllo dello stato di revoca: Online Certificate Status Protocol (OCSP) e gli elenchi di revoca dei certificati (CRL). Con OCSP, il client interroga un database di revoca autorevole che restituisce uno stato in tempo reale. Con un CRL, il client confronta il certificato con un elenco di certificati revocati che scarica e archivia periodicamente. I client si rifiutano di accettare i certificati che sono stati revocati.

Sia OCSP che i CRL dipendono dalle informazioni di convalida incorporate nei certificati. Per questo motivo, una CA emittente deve essere configurata per supportare uno o entrambi questi meccanismi prima dell'emissione. Per informazioni sulla selezione e l'implementazione della revoca gestita tramiteCA privata AWS, vedere. Impostazione di un metodo di revoca dei certificati

I certificati revocati vengono sempre registrati nei rapporti di CA privata AWS controllo.

> **Note**
>
> Gli emittenti di certificati con più account necessitano di autorizzazioni aggiuntive per revocare i certificati emessi; in caso contrario, il proprietario della CA deve eseguire la revoca. Per consentire la revoca da parte degli emittenti con account diversi, l'amministratore della CA deve creare due condivisioni RAM, entrambe indirizzate alla stessa CA:
>
> 1. Una condivisione con l'autorizzazione. `AWSRAMRevokeCertificateCertificateAuthority`
> 2. Una condivisione con il `AWSRAMDefaultPermissionCertificateAuthority` permesso.

Per revocare un certificato

Utilizza l'azione RevokeCertificateAPI o il comando revoke-certificate per revocare un certificato PKI privato. Il numero di serie deve essere in formato esadecimale. Puoi recuperare il numero di serie chiamando il comando get-certificate. Il comando `revoke-certificate` non restituisce una risposta.

```
$ 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"
```

Certificati revocati e OCSP

Le risposte OCSP possono richiedere fino a 60 minuti per riflettere il nuovo stato quando si revoca un certificato. In generale, OCSP tende a supportare una distribuzione più rapida delle informazioni di revoca perché, a differenza dei CRL che possono essere memorizzati nella cache dai client per giorni, le risposte OCSP in genere non vengono memorizzate nella cache dai client.

Certificati revocati in un CRL

Generalmente un CRL viene aggiornato circa 30 minuti dopo che un certificato viene revocato. Se per qualsiasi motivo un aggiornamento del CRL fallisce, effettua ulteriori tentativi ogni 15 minuti. CA privata AWS

Con Amazon CloudWatch, puoi creare allarmi per le metriche `CRLGenerated` e. `MisconfiguredCRLBucket` Per ulteriori informazioni, consulta Metriche supportate CloudWatch. Per ulteriori informazioni sulla creazione e configurazione di CRL, consulta Pianificazione di un elenco di revoca dei certificati (CRL).

L'esempio seguente mostra un certificato revocato in un elenco di revoche di certificati (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
```

Certificati revocati in un report di audit

Tutti i certificati, inclusi i certificati revocati, sono inclusi nel report di audit per una CA privata. L'esempio seguente mostra un report di audit con un certificato emesso e uno revocato. Per ulteriori informazioni, consulta Utilizzo dei report di controllo con la tua CA privata.

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


Automatizzazione dell'esportazione di un certificato rinnovato

Quando crei una CA, puoi CA privata AWS importarla AWS Certificate Manager e lasciare che ACM gestisca l'emissione e il rinnovo dei certificati. Se un certificato da rinnovare è associato a un servizio integrato, il servizio applica senza problemi il nuovo certificato. Tuttavia, se il certificato è stato originariamente esportato per essere utilizzato altrove nell'ambiente PKI (ad esempio, in un server o un'appliance locale), è necessario esportarlo nuovamente dopo il rinnovo.

Per una soluzione di esempio che automatizza il processo di esportazione ACM utilizzando Amazon e EventBridge AWS Lambda, consulta Automatizzare l'esportazione di certificati rinnovati.

Informazioni sui modelli di certificato

CA privata AWSutilizza modelli di configurazione per emettere certificati CA e certificati di entità finale. Quando si emette un certificato CA dalla console PCA, viene applicato automaticamente il modello di certificato CA principale o subordinato appropriato.

Se utilizzi la CLI o l'API per emettere un certificato, puoi fornire un modello ARN come parametro dell'azione. `IssueCertificate` Se non si fornisce alcun ARN, il `EndEntityCertificate/`

V1 modello viene applicato per impostazione predefinita. Per ulteriori informazioni, consulta la documentazione relativa all'IssueCertificateAPI e ai comandi issue-certificate.

> **Note**
>
> AWS Certificate Manager(ACM) gli utenti con accesso condiviso tra più account a una CA privata possono emettere certificati gestiti firmati dalla CA. Gli emittenti con più account sono vincolati da una politica basata sulle risorse e hanno accesso solo ai seguenti modelli di certificato per entità finali:
>
> - EndEntityCertificate/V1
> - EndEntityClientAuthCertificate/V1
> - EndEntityServerAuthCertificate/V1
> - BlankEndEntityCertificate_API passa/v1
> - BlankEndEntityCertificate_APIC SRPassthrough/v1
> - Certificato CA subordinato_ 0/V1 PathLen
>
> Per ulteriori informazioni, consulta Policy basate su risorse.

Argomenti

Varietà di modelli

CA privata AWSsupporta quattro varietà di modelli.

- Modelli di base

  Modelli predefiniti in cui non sono consentiti parametri passthrough.
- Modelli CSRPassthrough

  Modelli che estendono le versioni dei modelli di base corrispondenti consentendo il passaggio CSR. Le estensioni della CSR utilizzata per emettere il certificato vengono copiate nel certificato

emesso. Nei casi in cui la CSR contenga valori di estensione in conflitto con la definizione del modello, la definizione del modello avrà sempre la priorità più alta. Per ulteriori dettagli sulla priorità, consultaModello di ordine delle operazioni.

- Modelli API Pass Through

  Modelli che estendono le versioni dei modelli di base corrispondenti consentendo il passaggio delle API. I valori dinamici noti all'amministratore o ad altri sistemi intermedi potrebbero non essere noti all'entità che richiede il certificato, essere impossibili da definire in un modello e potrebbero non essere disponibili nella CSR. L'amministratore della CA, tuttavia, può recuperare informazioni aggiuntive da un'altra fonte di dati, ad esempio Active Directory, per completare la richiesta. Ad esempio, se una macchina non sa a quale unità organizzativa appartiene, l'amministratore può cercare le informazioni in Active Directory e aggiungerle alla richiesta di certificato includendo le informazioni in una struttura JSON.

  I valori nel `ApiPassthrough` parametro dell'`IssueCertificate`azione vengono copiati nel certificato emesso. Nei casi in cui il `ApiPassthrough` parametro contenga informazioni in conflitto con la definizione del modello, la definizione del modello avrà sempre la priorità più alta. Per ulteriori dettagli sulla priorità, vedereModello di ordine delle operazioni.

- Modelli API/CSR Passthrough.

  Modelli che estendono le versioni dei modelli di base corrispondenti consentendo il passaggio sia dell'API che della CSR. Le estensioni del CSR utilizzate per emettere il certificato vengono copiate nel certificato emesso e vengono copiati anche i valori nel `ApiPassthrough` parametro dell'`IssueCertificate`azione. Nei casi in cui la definizione del modello, i valori passthrough dell'API e le estensioni passthrough CSR presentano un conflitto, la definizione del modello ha la massima priorità, seguita dai valori passthrough dell'API, seguiti dalle estensioni passthrough CSR. Per ulteriori dettagli sulla priorità, consulta. Modello di ordine delle operazioni

Le tabelle seguenti elencano tutti i tipi di modello supportati da CA privata AWS con collegamenti alle relative definizioni.

> **Note**
>
> Per informazioni sugli ARN dei modelli nelle GovCloud regioni, consulta AWS Private Certificate Authorityla Guida per l'AWS GovCloud (US)utente.

Modelli di base

Nome modello	ARN modello	Tipo di certificato
CodeSigningCertificate/V1	`arn:aws:acm-pca:::template/CodeSigningCertificate/V1`	Firma del codice
EndEntityCertificate/V1	`arn:aws:acm-pca:::template/EndEntityCertificate/V1`	Entità finale
EndEntityClientAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityClientAuthCertificate/V1`	Entità finale
EndEntityServerAuthCertificate/V1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate/V1`	Entità finale
OCSP /V1 SigningCertificate	`arn:aws:acm-pca:::template/OCSPSigningCertificate/V1`	Firma OCSP
RootCACertificate/V1	`arn:aws:acm-pca:::template/RootCACertificate/V1`	CA
Certificato CA subordinato PathLen _ 0/V1	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1`	CA
Certificato CA subordinato_ 1/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1/V1`	CA

Nome modello	ARN modello	Tipo di certificato
Certificato CA subordinato_ 2/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2/V1`	CA
Certificato CA subordinato_ 3/V1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3/V1`	CA

Modelli CSRPassthrough

Nome modello	ARN modello	Tipo di certificato
BlankEndEntityCertificate_CSRPassthrough/v1	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_CSRPassthrough/V1`	Entità finale
BlankEndEntityCertificate_ _CSRPassthrough/v1 CriticalBasicConstraints	`arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassthrough/V1`	Entità finale
BlankSubordinateCertificato CA_PathLen 0_CSRPassthrough/v1	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1`	CA
BlankSubordinateCertificato CA_PathLen 1_CSRPassThrough/v1	`arn:aws:acm-pca:::template/BlankSubordinateCACertifica`	CA

Nome modello	ARN modello	Tipo di certificato
	te_PathLen1_CSRPas sthrough/V1	
BlankSubordinateCertificato CA_ PathLen 2_CSRPassThrough/v1	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen2_CSRPas sthrough/V1	CA
BlankSubordinateCertificato CA_ PathLen 3_CSRPassThrough/v1	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen3_CSRPas sthrough/V1	CA
CodeSigningCertificate_Pass through CSR/v1	arn:aws:acm-pca::: template/CodeSigni ngCertificate_CSRP assthrough/V1	Firma del codice
EndEntityCertificate_CSRPas sthrough/v1	arn:aws:acm-pca::: template/EndEntity Certificate_CSRPas sthrough/V1	Entità finale
EndEntityClientAuthCertific ate_CSRPassthrough/v1	arn:aws:acm-pca::: template/EndEntity ClientAuthCertific ate_CSRPassthrough/V1	Entità finale
EndEntityServerAuthCertific ate_CSRPassthrough/v1	arn:aws:acm-pca::: template/EndEntity ServerAuthCertific ate_CSRPassthrough/V1	Entità finale

Nome modello	ARN modello	Tipo di certificato
OCSP _CSRPassthrough/v1 SigningCertificate	arn:aws:acm-pca:::template/OCSPSigningCertificate_CSRPassthrough/V1	Firma OCSP
Certificato CA PathLen subordinato_ 0_CSRPassThrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
Certificato CA subordinato_ 1_CSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
Certificato CA subordinato_ 2_CSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
Certificato CA subordinato_ 3_CSRPassThrough/v1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA

API Pass attraverso modelli

Nome modello	ARN modello	Tipo di certificato
BlankEndEntityCertificate_ApiPass/v1	arn:aws:acm-pca:::template/BlankEndE	Entità finale

Nome modello	ARN modello	Tipo di certificato
	ntityCertificate_APIPassthrough/V1	
BlankEndEntityCertificate_ _ApiPassThrough/v1 CriticalBasicConstraints	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough/V1	Entità finale
CodeSigningCertificate_APIPass/v1	arn:aws:acm-pca:::template/CodeSigningCertificate_APIPassthrough/V1	Firma del codice
EndEntityCertificate_APIPass/v1	arn:aws:acm-pca:::template/EndEntityCertificate_APIPassthrough/V1	Entità finale
EndEntityClientAuthCertificate_APIPass/v1	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_APIPassthrough/V1	Entità finale
EndEntityServerAuthCertificate_APIPass/v1	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APIPassthrough/V1	Entità finale
OCSP _APIPassThrough/v1 SigningCertificate	arn:aws:acm-pca:::template/OCSPSigningCertificate_APIPassthrough/V1	Firma OCSP

Nome modello	ARN modello	Tipo di certificato
RootCACertificate_APIPassThrough/v1	arn:aws:acm-pca:::template/RootCACertificate_APIPassthrough/V1	CA
BlankRootCACertificate_APIPASSTHROU/v1	arn:aws:acm-pca:::template/BlankRootCACertificate_APIPassthrough/V1	CA
BlankRootCertificato_ PathLen0_APIPassThrough/v1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen0_APIPassthrough/V1	CA
BlankRootCertificato CA_ PathLen1_APIPassThrough/v1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen1_APIPassthrough/V1	CA
BlankRootCertificato CA_ PathLen2_APIPassThrough/v1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen2_APIPassthrough/V1	CA
BlankRootCertificato CA_ PathLen3_APIPassThrough/v1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen3_APIPassthrough/V1	CA
Certificato CA subordinato_ PathLen 0_ApiPassThrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APIPassthrough/V1	CA

Nome modello	ARN modello	Tipo di certificato
BlankSubordinateCertificato CA_0_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1	CA
Certificato CA subordinato_PathLen 1_APIPassThrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
BlankSubordinateCertificato CA_1_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
Certificato CA subordinato_PathLen 2_APIPassThrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
BlankSubordinateCertificato CA_2_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
Certificato CA subordinato_PathLen 3_APIPassThrough/v1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APIPassthrough/V1	CA

Nome modello	ARN modello	Tipo di certificato
BlankSubordinateCertificato CA_3_APIPassThrough/v1 PathLen	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1	CA

Modelli API/CSR Passthrough

Nome modello	ARN modello	Tipo di certificato
BlankEndEntityCertificate_APICSRPassthrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_APICSRPassthrough/V1	Entità finale
BlankEndEntityCertificate_CriticalBasicConstraints_ApicSRPassthrough/v1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APICSRPassthrough/V1	Entità finale
CodeSigningCertificate_APICSRPassthrough/v1	arn:aws:acm-pca:::template/CodeSigningCertificate_APICSRPassthrough/V1	Firma del codice
EndEntityCertificate_APICSRPassthrough/v1	arn:aws:acm-pca:::template/EndEntityCertificate_APICSRPassthrough/V1	Entità finale
EndEntityClientAuthCertificate_APICSRPassthrough/v1	arn:aws:acm-pca:::template/EndEntity	Entità finale

Nome modello	ARN modello	Tipo di certificato
	`ClientAuthCertificate_APICSRPassthrough/V1`	
EndEntityServerAuthCertificate_APICSRPassthrough/v1	`arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APICSRPassthrough/V1`	Entità finale
OCSP _APICSRPassthrough/v1 SigningCertificate	`arn:aws:acm-pca:::template/OCSPSigningCertificate_APICSRPassthrough/V1`	Firma OCSP
Certificato CA PathLen subordinato_ 0_APICSRPassthrough/v1	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APICSRPassthrough/V1`	CA
BlankSubordinateCertificatoCA_ 0_APICSRPassthrough/v1 PathLen	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1`	CA
Certificato CA subordinato_ PathLen 1_APICSRPassThrough/v1	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APICSRPassthrough/V1`	CA

Nome modello	ARN modello	Tipo di certificato
BlankSubordinateCertificatoCA_ 1_APICSRPassThrough/v1 PathLen	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1`	CA
Certificato CA subordinato_ PathLen 2_APICSRPassThrough/3_APIPassThroughV1 PathLen	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APICSRPassthrough/V1`	CA
BlankSubordinateCertificatoCA_ 2_APICSRPassThrough/v1 PathLen	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1`	CA
Certificato CA subordinato_ PathLen 3_APICSRPassThrough/v1	`arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA
BlankSubordinateCertificatoCA_ 3_APICSRPassthrough/v1 PathLen	`arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1`	CA

Modello di ordine delle operazioni

Le informazioni contenute in un certificato emesso possono provenire da quattro fonti: la definizione del modello, il passthrough dell'API, il passthrough CSR e la configurazione della CA.

I valori di passthrough dell'API vengono rispettati solo quando si utilizza un modello di passthrough API o un modello di passthrough APICSR. Il passthrough CSR viene rispettato solo quando si utilizza un modello passthrough CSRPassthrough o APICSR. Quando queste fonti di informazioni sono in conflitto, di solito si applica una regola generale: per ogni valore di estensione, la definizione del modello ha la massima priorità, seguita dai valori passthrough dell'API, seguiti dalle estensioni passthrough CSR.

Examples (Esempi)

1. La definizione del modello per EndEntityClientAuthCertificate_ApiPassThrough definisce l' ExtendedKeyUsage estensione con un valore di «autenticazione del server Web TLS, autenticazione del client Web TLS». Se ExtendedKeyUsage è definito nella CSR o nel `IssueCertificate` `ApiPassthrough` parametro, il `ApiPassthrough` valore per ExtendedKeyUsage verrà ignorato perché la definizione del modello ha la priorità e il valore CSR per ExtendedKeyUsage valore verrà ignorato perché il modello non è una varietà passthrough CSR.

   > **Note**
   >
   > La definizione del modello copia tuttavia altri valori del CSR, come Subject e Subject Alternative Name. Questi valori sono comunque presi dalla CSR anche se il modello non è una variante CSR passthrough, perché la definizione del modello ha sempre la massima priorità.

2. La definizione del modello per EndEntityClientAuthCertificate_ApicSRPassthrough definisce l'estensione Subject Alternative Name (SAN) come copiata dall'API o dalla CSR. Se l'estensione SAN è definita nel CSR e fornita nel `IssueCertificate ApiPassthrough` parametro, il valore passthrough dell'API avrà la priorità perché i valori passthrough dell'API hanno la priorità sui valori passthrough CSR.

Definizioni dei modelli

Le seguenti sezioni forniscono dettagli di configurazione sui modelli di CA privata AWS certificato supportati.

BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

Con i modelli di certificato di entità finale vuoti, è possibile emettere certificati di entità finale con solo i vincoli X.509 Basic. Questo è il certificato per entità finale più semplice che è CA privata AWS possibile emettere, ma può essere personalizzato utilizzando la struttura dell'API. L'estensione Basic constraints definisce se il certificato è o meno un certificato CA. Un modello di certificato di entità finale vuoto impone il valore FALSE for Basic per i vincoli per garantire che venga emesso un certificato di entità finale e non un certificato CA.

È possibile utilizzare modelli passthrough vuoti per emettere certificati smart card che richiedono valori specifici per Key usage (KU) e Extended key usage (EKU). Ad esempio, l'utilizzo esteso delle chiavi può richiedere l'autenticazione del client e l'accesso con smart card, mentre l'utilizzo delle chiavi può richiedere la firma digitale, il non ripudio e la cifratura delle chiavi. A differenza di altri modelli passthrough, i modelli di certificato di entità finale vuoti consentono la configurazione di estensioni KU ed EKU, dove KU può essere uno dei nove valori supportati (DigitalSignature, NonRepudiation, KeyEncipherment, DataEncipherment, KeyAgreement, CRLSign keyCertSign, EncipherOnly e DecipherOnly) e EKU può essere uno qualsiasi dei valori supportati (ServerAuth, ClientAuth, coplanning, EmailProtection), timestamping e OCSPSigning) oltre a estensioni personalizzate.

BlankEndEntityCertificate_APIPass/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	CA: FALSE
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankEndEntityCertificateDefinizione _API/CSRPassthrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankEndEntityCertificate_APICSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	CA: FALSE
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankEndEntityCertificate_ _APICSRPassthrough/v1CriticalBasicConstraints: definizione

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankEndEntityCertificate_ CriticalBasicConstraints _ApicSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, CA:FALSE
Identificatore chiave di autorità	[SKI da certificato CA]

Parametro X509v3	Valore
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA, API o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankEndEntityCertificateDefinizione _ _ApiPassThrough/v1 CriticalBasicConstraints

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankEndEntityCertificate_ CriticalBasicConstraints _ApiPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, CA:FALSE
Identificatore chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione o dall'API CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankEndEntityCertificate_ CriticalBasicConstraints _CSRPassthrough/v1 definizione

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankEndEntityCertificate_ CriticalBasicConstraints _CSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, CA:FALSE
Identificatore chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankEndEntityCertificateDefinizione _CSRPassthrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankEndEntityCertificate_csrPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	CA: FALSE
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 0_CSRPassThrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCertificate_ PathLen 0_CSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 0`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 0_API/CSRPassthrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 0_ApiCSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 0`

Parametro X509v3	Valore
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 0_APIPassThrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 0_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 0`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

BlankSubordinateDefinizione CACertificate_ PathLen 1_APIPassThrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 1_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 1`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 1_CSRPassThrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 1_CSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 1`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 1_API/CSRPassThrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 1_ApiCSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 1`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 2_APIPassThrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 2_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 2`

Parametro X509v3	Valore
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 2_CSRPassThrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 2_CSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 2`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 2_API/CSRPassthrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 2_ApiCSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 2`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 3_APIPassThrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 3_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 3`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 3_CSRPassThrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 3_CSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 3`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

BlankSubordinateDefinizione CACertificate_ PathLen 3_API/CSRPassthrough/v1

Per informazioni generali sui modelli vuoti, vedere. BlankEndEntityCertificateDefinizione _ApiPassThrough/v1

BlankSubordinateCACertificate_ PathLen 3_APIC SRPassThrough

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 3`

Parametro X509v3	Valore
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

CodeSigningCertificateDefinizione /V1

Questo modello viene utilizzato per creare certificati per la firma del codice. È possibile utilizzare i certificati di firma del codice da CA privata AWS con qualsiasi soluzione di firma del codice basata su un'infrastruttura CA privata. Ad esempio, i clienti che utilizzano Firma codice per AWS IoT possono generare un certificato di firma del codice con CA privata AWS e importarlo in AWS Certificate Manager. Per ulteriori informazioni, consulta A cosa serve la firma del codice? AWS IoT e ottenere e importare un certificato di firma del codice.

CodeSigningCertificate/V1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	`CA:FALSE`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica
Utilizzo esteso delle chiavi	Critico, firma del codice
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

*I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

CodeSigningCertificateDefinizione _API/CSRPassthrough/v1

Questo modello estende CodeSigningCertificate /V1 per supportare i valori passthrough API e CSR.

CodeSigningCertificate_API/CSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	`CA:FALSE`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica
Utilizzo esteso delle chiavi	Critico, firma del codice
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

CodeSigningCertificateDefinizione _ApiPassThrough/v1

Questo modello è identico al `CodeSigningCertificate` modello con una differenza: in questo modello, CA privata AWS passa al certificato estensioni aggiuntive tramite l'API se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nell'API.

CodeSigningCertificate_API Passthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]

Parametro X509v3	Valore
Subject	[Passthrough da API o CSR]
Vincoli di base	`CA:FALSE`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica
Utilizzo esteso delle chiavi	Critico, firma del codice
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

CodeSigningCertificateDefinizione _CSRPassthrough/v1

Questo modello è identico al modello `CodeSigningCertificate` con una differenza: in questo modello, CA privata AWS passa estensioni aggiuntive dalla richiesta di firma certificato (CSR) al certificato se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nella CSR.

CodeSigningCertificate_CSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	`CA:FALSE`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica

Parametro X509v3	Valore
Utilizzo esteso delle chiavi	Critico, firma del codice
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

*I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityCertificateDefinizione /V1

Questo modello viene utilizzato per creare certificati per entità finali quali sistemi operativi o server Web.

EndEntityCertificate/V1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	CA:FALSE
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione server Web TLS, autenticazione client Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

*I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityCertificateDefinizione _API/CSRPassthrough/v1

Questo modello estende EndEntityCertificate /V1 per supportare i valori passthrough API e CSR.

EndEntityCertificate_API/CSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	CA:FALSE
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione server Web TLS, autenticazione client Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityCertificateDefinizione _ApiPassThrough/v1

Questo modello è identico al `EndEntityCertificate` modello con una differenza: in questo modello, CA privata AWS passa al certificato estensioni aggiuntive tramite l'API se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nell'API.

EndEntityCertificate_API Passthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	CA:FALSE

Parametro X509v3	Valore
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione server Web TLS, autenticazione client Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityCertificateDefinizione _CSRPassthrough/v1

Questo modello è identico al modello `EndEntityCertificate` con una differenza: in questo modello, CA privata AWS passa estensioni aggiuntive dalla richiesta di firma certificato (CSR) al certificato se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nella CSR.

EndEntityCertificate_CSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	CA:`FALSE`
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione server Web TLS, autenticazione client Web TLS

Parametro X509v3	Valore
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

*I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityClientAuthCertificateDefinizione /V1

Questo modello si differenzia dall'`EndEntityCertificate`unico nel valore di utilizzo della chiave estesa, che lo limita all'autenticazione del client Web TLS.

EndEntityClientAuthCertificate/V1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	CA:`FALSE`
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione client Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

*I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityClientAuthCertificateDefinizione _APICSRPassthrough/v1

Questo modello estende EndEntityClientAuthCertificate /V1 per supportare i valori passthrough API e CSR.

EndEntityClientAuthCertificate_API/CSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	CA:FALSE
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione client Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityClientAuthCertificateDefinizione _ApiPassThrough/v1

Questo modello è identico al modello `EndEntityClientAuthCertificate` con una differenza. In questo modello, CA privata AWS passa al certificato estensioni aggiuntive tramite l'API se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nell'API.

EndEntityClientAuthCertificate_API Passthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	CA:FALSE
Identificatore della chiave di autorità	[SKI da certificato CA]

Parametro X509v3	Valore
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione client Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityClientAuthCertificateDefinizione _CSRPassthrough/v1

Questo modello è identico al modello `EndEntityClientAuthCertificate` con una differenza. In questo modello, CA privata AWS passa estensioni aggiuntive dalla richiesta di firma certificato (CSR) al certificato se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nella CSR.

EndEntityClientAuthCertificate_CSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	CA:`FALSE`
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione client Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

*I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityServerAuthCertificateDefinizione /V1

Questo modello si differenzia dall'`EndEntityCertificate`unico nel valore di utilizzo della chiave estesa, che lo limita all'autenticazione del server Web TLS.

EndEntityServerAuthCertificate/V1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	CA:`FALSE`
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione del server Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

*I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityServerAuthCertificateDefinizione _API/CSRPassthrough/v1

Questo modello estende EndEntityServerAuthCertificate /V1 per supportare i valori passthrough API e CSR.

EndEntityServerAuthCertificate_API/CSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]

Parametro X509v3	Valore
Subject	[Passthrough da API o CSR]
Vincoli di base	CA:FALSE
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione del server Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityServerAuthCertificateDefinizione _ApiPassThrough/v1

Questo modello è identico al modello `EndEntityServerAuthCertificate` con una differenza. In questo modello, CA privata AWS passa al certificato estensioni aggiuntive tramite l'API se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nell'API.

EndEntityServerAuthCertificate_API Passthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	CA:FALSE
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi

Parametro X509v3	Valore
Utilizzo esteso delle chiavi	Autenticazione del server Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

EndEntityServerAuthCertificateDefinizione _CSRPassthrough/v1

Questo modello è identico al modello `EndEntityServerAuthCertificate` con una differenza. In questo modello, CA privata AWS passa estensioni aggiuntive dalla richiesta di firma certificato (CSR) al certificato se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nella CSR.

EndEntityServerAuthCertificate_CSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	CA:`FALSE`
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica, cifratura delle chiavi
Utilizzo esteso delle chiavi	Autenticazione del server Web TLS
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

*I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione OCSP SigningCertificate /V1

Questo modello viene utilizzato per creare certificati per la firma delle risposte OCSP. Il modello è identico al `CodeSigningCertificate` modello, tranne per il fatto che il valore di utilizzo della chiave estesa specifica la firma OCSP anziché la firma del codice.

OCSP /V1 SigningCertificate

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	`CA:FALSE`
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica
Utilizzo esteso delle chiavi	Firma OCSP critica
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

*I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione OCSP SigningCertificate _API/CSRPassthrough/v1

Questo modello estende SigningCertificate OCSP /V1 per supportare i valori passthrough API e CSR.

OCSP _API/CSRPassthrough/v1 SigningCertificate

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	`CA:FALSE`

Parametro X509v3	Valore
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica
Utilizzo esteso delle chiavi	Firma OCSP critica
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione OCSP SigningCertificate _APIPassThrough/v1

Questo modello è identico al modello `0CSPSigningCertificate` con una differenza. In questo modello, CA privata AWS passa al certificato estensioni aggiuntive tramite l'API se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nell'API.

OCSP SigningCertificate _APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	`CA:FALSE`
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica
Utilizzo esteso delle chiavi	Firma OCSP critica
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione OCSP SigningCertificate _CSRPassthrough/v1

Questo modello è identico al modello `OCSPSigningCertificate` con una differenza. In questo modello, CA privata AWS passa estensioni aggiuntive dalla richiesta di firma certificato (CSR) al certificato se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nella CSR.

OCSP SigningCertificate _CSRPassthrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	`CA:FALSE`
Identificatore della chiave di autorità	[SKI da certificato CA]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica
Utilizzo esteso delle chiavi	Firma OCSP critica
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

*I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione RootCACertificate/V1

Questo modello viene utilizzato per rilasciare certificati emessi da una CA root autofirmati. I certificati emessi da una CA includono un'estensione dei vincoli di base critici con il campo CA impostato su `TRUE` per indicare che il certificato può essere utilizzato per emettere certificati emessi da una CA. Il modello non specifica una lunghezza del percorso (pathLenConstraint) perché ciò potrebbe inibire le future espansioni della gerarchia. L'utilizzo esteso della chiave è escluso per impedire l'utilizzo

del certificato emesso da una CA come certificato client o server TLS. Nessuna informazione CRL è specificata perché un certificato autofirmato non può essere revocato.

RootCACertificate/V1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`
Identificatore chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica keyCertSign, segno CRL
Punti di distribuzione CRL	N/D

Definizione RootCACertificate_APIPassThrough/v1

Questo modello estende RootCAertificate/v1 per supportare i valori passthrough delle API.

RootCACertificate_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`
Identificatore della chiave di autorità	[Passthrough dall'API]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica keyCertSign, segno CRL
Punti di distribuzione CRL*	N/D

BlankRootDefinizione CACertificate_ApiPassThrough/v1

Con i modelli di certificato radice vuoti, è possibile emettere certificati radice con solo i vincoli di base X.509. Questo è il certificato radice più semplice che è CA privata AWS possibile emettere, ma può essere personalizzato utilizzando la struttura dell'API. L'estensione Basic Constraints definisce se il certificato è o meno un certificato CA. Un modello di certificato radice vuoto impone un valore pari a quattro vincoli di base `TRUE` per garantire l'emissione di un certificato CA radice.

È possibile utilizzare modelli root passthrough vuoti per emettere certificati radice che richiedono valori specifici per l'utilizzo delle chiavi (KU). Ad esempio, l'utilizzo delle chiavi potrebbe richiedere `keyCertSign` and`cRLSign`, ma non`digitalSignature`. A differenza degli altri modelli di certificato root passthrough non vuoto, i modelli di certificato radice vuoti consentono la configurazione dell'estensione KU, dove KU può essere uno qualsiasi dei nove valori supportati (`digitalSignature`non`Repudiation`,`keyEncipherment`,`dataEncipherment`,`keyAgreement`,, `keyCertSign` `cRLSign`encipher`Only`, edecipher`Only`).

BlankRootCACertificate_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`
Identificatore chiave dell'oggetto	[Derivato da CSR]

BlankRootDefinizione CACertificate_ PathLen 0_APIPassThrough/v1

Per informazioni generali sui modelli CA root vuoti, vedere. ???

BlankRootCACertificate_ PathLen 0_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 0`

Parametro X509v3	Valore
Identificatore chiave dell'oggetto	[Derivato da CSR]

BlankRootDefinizione CACertificate_ PathLen 1_APIPassThrough/v1

Per informazioni generali sui modelli CA root vuoti, vedere. ???

BlankRootCACertificate_ PathLen 1_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 1`
Identificatore chiave dell'oggetto	[Derivato da CSR]

BlankRootDefinizione CACertificate_ PathLen 2_APIPassThrough/v1

Per informazioni generali sui modelli CA root vuoti, vedere. ???

BlankRootCACertificate_ PathLen 2_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 2`
Identificatore chiave dell'oggetto	[Derivato da CSR]

BlankRootDefinizione CACertificate_ PathLen 3_APIPassThrough/v1

Per informazioni generali sui modelli CA root vuoti, vedere. ???

BlankRootCACertificate_ PathLen 3_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 3`
Identificatore chiave dell'oggetto	[Derivato da CSR]

Definizione SubordinateCAertificate_ PathLen 0/V1

Questo modello viene utilizzato per emettere certificati CA subordinati con una lunghezza del percorso di. `0` I certificati emessi da una CA includono un'estensione dei vincoli di base critici con il campo CA impostato su `TRUE` per indicare che il certificato può essere utilizzato per emettere certificati emessi da una CA. L'utilizzo esteso della chiave non è incluso, il che impedisce l'utilizzo del certificato emesso da una CA come certificato client o server TLS.

Per ulteriori informazioni sui percorsi di certificazione, consulta Impostazione dei vincoli di lunghezza nel percorso di certificazione.

Certificato CA subordinato _ 0/V1 PathLen

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 0`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

*I punti di distribuzione CRL sono inclusi nei certificati emessi con questo modello solo se la CA è configurata con la generazione di CRL abilitata.

Definizione SubordinateCACertificate_ 0_APICSRPassThrough/v1 PathLen

Questo modello estende SubordinateCAertificate_ 0/V1 per supportare i valori passthrough API e CSR. PathLen

SubordinateCAertificate_ 0_APICSRPassthrough/v1 PathLen

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 0`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione SubordinateCACertificate_ PathLen 0_APIPassThrough/v1

Questo modello estende SubordinateCAertificate_ 0/V1 per supportare i valori passthrough delle API. PathLen

PathLenSubordinateCAertificate_ 0_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]

Parametro X509v3	Valore
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 0`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione SubordinateCAertificate_ PathLen 0_CSRPassThrough/v1

Questo modello è identico al modello `SubordinateCACertificate_PathLen0` con una differenza: in questo modello, CA privata AWS passa estensioni aggiuntive dalla richiesta di firma certificato (CSR) al certificato se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nella CSR.

> **Note**
>
> È necessario creare una CSR che contenga estensioni aggiuntive personalizzate all'esterno di. CA privata AWS

SubordinateCAertificate_ PathLen 0_CSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 0`
Identificatore della chiave di autorità	[SKI di CA Certificate]

Parametro X509v3	Valore
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

*I punti di distribuzione CRL sono inclusi nei certificati emessi con questo modello solo se la CA è configurata con la generazione CRL abilitata.

Certificato CA subordinato_ 1/V1 definizione PathLen

Questo modello viene utilizzato per emettere certificati CA subordinati con una lunghezza del percorso di. 1 I certificati CA includono un'estensione fondamentale dei vincoli Basic con il campo CA impostato su `TRUE` per indicare che il certificato può essere utilizzato per emettere certificati CA. L'utilizzo esteso della chiave non è incluso, il che impedisce l'utilizzo del certificato emesso da una CA come certificato client o server TLS.

Per ulteriori informazioni sui percorsi di certificazione, consulta Impostazione dei vincoli di lunghezza nel percorso di certificazione.

PathLenCertificato CA subordinato_ 1/V1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 1`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL

Parametro X509v3	Valore
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

*I punti di distribuzione CRL sono inclusi nei certificati emessi con questo modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione subordinateCACertificate_ PathLen 1_APIC SRPassthrough/v1

Questo modello estende SubordinateCAertificate_ 1/V1 per supportare i valori passthrough API e CSR. PathLen

SubordinateCAertificate_ 1_APICSRPassthrough/v1 PathLen

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 1`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione SubordinateCACertificate_ PathLen 1_APIPassThrough/v1

Questo modello estende SubordinateCAertificate_ 0/V1 per supportare i valori passthrough delle API. PathLen

PathLenSubordinateCAertificate_ 1_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 1`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione SubordinateCACertificate_ PathLen 1_CSRPassThrough/v1

Questo modello è identico al modello `SubordinateCACertificate_PathLen1` con una differenza: in questo modello, CA privata AWS passa estensioni aggiuntive dalla richiesta di firma certificato (CSR) al certificato se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nella CSR.

> **Note**
>
> È necessario creare una CSR che contenga estensioni aggiuntive personalizzate all'esterno di. CA privata AWS

SubordinateCAertificate_ PathLen 1_CSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]

Parametro X509v3	Valore
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 1`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

*I punti di distribuzione CRL sono inclusi nei certificati emessi con questo modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione CACertificate_ 2/V1 subordinata PathLen

Questo modello viene utilizzato per emettere certificati emessi da una CA subordinata con una lunghezza di percorso pari a 2. I certificati CA includono un'estensione fondamentale dei vincoli di base con il campo CA impostato per indicare che il certificato può essere utilizzato `TRUE` per emettere certificati CA. L'utilizzo esteso della chiave non è incluso, il che impedisce l'utilizzo del certificato emesso da una CA come certificato client o server TLS.

Per ulteriori informazioni sui percorsi di certificazione, consulta Impostazione dei vincoli di lunghezza nel percorso di certificazione.

PathLenCertificato CA subordinato_ 2/V1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 2`
Identificatore della chiave di autorità	[SKI di CA Certificate]

Parametro X509v3	Valore
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

*I punti di distribuzione CRL sono inclusi nei certificati emessi con questo modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione subordinatecACertificate_ PathLen 2_APIC SRPassthrough/v1

Questo modello estende SubordinateCAertificate_ 2/V1 per supportare i valori passthrough API e CSR. PathLen

SubordinateCAertificate_ 2_APICSRPassthrough/v1 PathLen

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 2`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione SubordinateCACertificate_ PathLen 2_APIPassThrough/v1

Questo modello estende SubordinateCAertificate_ 2/V1 per supportare i valori passthrough delle API. PathLen

PathLenSubordinateCAertificate_ 2_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 2`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione SubordinateCACertificate_ PathLen 2_CSRPassThrough/v1

Questo modello è identico al modello `SubordinateCACertificate_PathLen2` con una differenza: in questo modello, CA privata AWS passa estensioni aggiuntive dalla richiesta di firma certificato (CSR) al certificato se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nella CSR.

> **Note**
>
> È necessario creare una CSR che contenga estensioni aggiuntive personalizzate all'esterno di. CA privata AWS

SubordinateCAertificate_ PathLen 2_CSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 2`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

*I punti di distribuzione CRL sono inclusi nei certificati emessi con questo modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione del certificato CA subordinato_ 3/V1 PathLen

Questo modello viene utilizzato per emettere certificati emessi da una CA subordinata con una lunghezza di percorso pari a 3. I certificati CA includono un'estensione fondamentale dei vincoli Basic con il campo CA impostato per indicare che il certificato può essere utilizzato `TRUE` per emettere certificati CA. L'utilizzo esteso della chiave non è incluso, il che impedisce l'utilizzo del certificato emesso da una CA come certificato client o server TLS.

Per ulteriori informazioni sui percorsi di certificazione, consulta Impostazione dei vincoli di lunghezza nel percorso di certificazione.

PathLenCertificato CA subordinato_ 3/V1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]

Parametro X509v3	Valore
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 3`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

*I punti di distribuzione CRL sono inclusi nei certificati emessi con questo modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione subordinateCACertificate_ PathLen 3_APIC SRPassthrough/v1

Questo modello estende SubordinateCAertificate_ 3/V1 per supportare i valori passthrough API e CSR. PathLen

SubordinateCAertificate_ 3_APICSRPassthrough/v1 PathLen

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 3`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione SubordinateCACertificate_ PathLen 3_APIPassThrough/v1

Questo modello estende SubordinateCAertificate_ 3/V1 per supportare i valori passthrough delle API. PathLen

PathLenSubordinateCAertificate_ 3_APIPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da API o CSR]
Subject	[Passthrough da API o CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 3`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA]

* I punti di distribuzione CRL sono inclusi nel modello solo se la CA è configurata con la generazione CRL abilitata.

Definizione SubordinateCACertificate_ PathLen 3_CSRPassThrough/v1

Questo modello è identico al modello `SubordinateCACertificate_PathLen3` con una differenza: in questo modello, CA privata AWS passa estensioni aggiuntive dalla richiesta di firma certificato (CSR) al certificato se le estensioni non sono specificate nel modello. Le estensioni specificate nel modello sostituiscono sempre le estensioni nella CSR.

> **Note**
>
> È necessario creare una CSR che contenga estensioni aggiuntive personalizzate all'esterno di. CA privata AWS

SubordinateCAertificate_ PathLen 3_CSRPassThrough/v1

Parametro X509v3	Valore
Nome alternativo del soggetto	[Passthrough da CSR]
Subject	[Passthrough da CSR]
Vincoli di base	Critico, `CA:TRUE`, `pathlen: 3`
Identificatore della chiave di autorità	[SKI di CA Certificate]
Identificatore della chiave dell'oggetto	[Derivato da CSR]
Utilizzo delle chiavi	Firma digitale critica`keyCertSign` , segno CRL
Punti di distribuzione CRL*	[Passthrough dalla configurazione CA o CSR]

*I punti di distribuzione CRL sono inclusi nei certificati emessi con questo modello solo se la CA è configurata con la generazione CRL abilitata.

Utilizzo dell'CA privata AWSAPI (esempi Java)

È possibile utilizzare l'API AWS Private Certificate Authority per interagire programmaticamente con il servizio inviando richieste HTTP. Il servizio restituisce risposte HTTP. Per ulteriori informazioni, consulta AWS Private Certificate AuthorityAPI Reference.

Oltre all'API HTTP, puoi utilizzare gli SDK AWS e gli strumenti a riga di comando per interagire con CA privata AWS. Questa modalità è consigliata rispetto all'API HTTP. Per ulteriori informazioni, consulta Strumenti per Amazon Web Services. I seguenti argomenti illustrano come utilizzare AWS SDK for Java per programmare l'API CA privata AWS.

I GetCertificateAuthorityCsrcamerieri di supporto DescribeCertificateAuthorityAuditReportalle operazioni e alle operazioni. GetCertificate È possibile utilizzare i waiter per controllare la progressione del codice in base alla presenza o allo stato di determinate risorse. Per ulteriori informazioni, consulta i seguenti argomenti e la sezione dedicata ai camerieri AWS SDK for Java nel blog per sviluppatori. AWS

Argomenti

- Creare e attivare una CA root a livello di programmazione
- Creare e attivare una CA subordinata a livello di codice
- CreateCertificateAuthority
- Utilizzo CreateCertificateAuthority per supportare Active Directory
- CreateCertificateAuthorityAuditReport
- CreatePermission
- DeleteCertificateAuthority
- DeletePermission
- DeletePolicy
- DescribeCertificateAuthority
- DescribeCertificateAuthorityAuditReport
- GetCertificate
- GetCertificateAuthorityCertificate
- GetCertificateAuthorityCsr
- GetPolicy
- ImportCertificateAuthorityCertificate

- IssueCertificate
- ListCertificateAuthorities
- ListPermissions
- ListTags
- PutPolicy
- RestoreCertificateAuthority
- RevokeCertificate
- TagCertificateAuthorities
- UntagCertificateAuthority
- UpdateCertificateAuthority
- Crea CA e certificati con nomi di oggetto personalizzati
- Crea certificati con estensioni personalizzate

Creare e attivare una CA root a livello di programmazione

Questo esempio di Java mostra come attivare una CA root utilizzando quanto segueCA privata AWSOperazioni dell'API:

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.ASN1Subject;
import com.amazonaws.services.acmpca.model.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.model.CertificateAuthorityType;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.model.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.KeyAlgorithm;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Tag;

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

import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrRequest;
import com.amazonaws.services.acmpca.model.GetCertificateAuthorityCsrResult;
import com.amazonaws.services.acmpca.model.GetCertificateRequest;
import com.amazonaws.services.acmpca.model.GetCertificateResult;
import
 com.amazonaws.services.acmpca.model.ImportCertificateAuthorityCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateRequest;
import com.amazonaws.services.acmpca.model.IssueCertificateResult;
import com.amazonaws.services.acmpca.model.SigningAlgorithm;
import com.amazonaws.services.acmpca.model.Validity;

import com.amazonaws.AmazonClientException;
import com.amazonaws.services.acmpca.model.CertificateMismatchException;
import com.amazonaws.services.acmpca.model.ConcurrentModificationException;
import com.amazonaws.services.acmpca.model.LimitExceededException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.MalformedCertificateException;
import com.amazonaws.services.acmpca.model.MalformedCSRException;
import com.amazonaws.services.acmpca.model.RequestFailedException;
import com.amazonaws.services.acmpca.model.RequestInProgressException;
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;
import com.amazonaws.services.acmpca.model.AWSACMPCAException;
```

```
import com.amazonaws.waiters.Waiter;
import com.amazonaws.waiters.WaiterParameters;
import com.amazonaws.waiters.WaiterTimedOutException;
import com.amazonaws.waiters.WaiterUnrecoverableException;

public class RootCAActivation {
    public static void main(String[] args) throws Exception {
        // Define the endpoint region for your sample.
        String endpointRegion = "region"; // Substitute your region here, e.g. "us-
west-2"

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

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

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

        // Define a certificate authority type
        CertificateAuthorityType CAtype = CertificateAuthorityType.ROOT;

        // ** Execute core code samples for Root CA activation in sequence **
        AWSACMPCA client = ClientBuilder(endpointRegion);
        String rootCAArn = CreateCertificateAuthority(configCA, crlConfigure, CAtype,
 client);
        String csr = GetCertificateAuthorityCsr(rootCAArn, client);
        String rootCertificateArn = IssueCertificate(rootCAArn, csr, client);
```

```
        String rootCertificate = GetCertificate(rootCertificateArn, rootCAArn, client);
        ImportCertificateAuthorityCertificate(rootCertificate, rootCAArn, client);
    }

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

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

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

        return client;
    }

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

        // Create the request object.
        CreateCertificateAuthorityRequest createCARequest = new
CreateCertificateAuthorityRequest();
        createCARequest.withCertificateAuthorityConfiguration(configCA);
```

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

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

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

        return rootCAArn;
    }

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

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

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

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

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

        return csr;
    }

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

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

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

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

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

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

        // Set the validity period for the certificate to be issued.
        Validity validity = new Validity();
        validity.withValue(3650L);
```

```
        validity.withType("DAYS");
        issueRequest.withValidity(validity);

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

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

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

        return rootCertificateArn;
    }

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

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

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

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

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

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

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

        // Get the certificate and certificate chain and display the result.
        String rootCertificate = certificateResult.getCertificate();
        System.out.println(rootCertificate);

        return rootCertificate;
    }

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

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

        ByteBuffer certByteBuffer = stringToByteBuffer(rootCertificate);
```

```
        importRequest.setCertificate(certByteBuffer);

        importRequest.setCertificateChain(null);

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

        // Import the certificate.
        try {
            client.importCertificateAuthorityCertificate(importRequest);
        } catch (CertificateMismatchException ex) {
            throw ex;
        } catch (MalformedCertificateException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (RequestInProgressException ex) {
            throw ex;
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (RequestFailedException ex) {
            throw ex;
        }

        System.out.println("Root CA certificate successfully imported.");
        System.out.println("Root CA activated successfully.");
    }

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

Creare e attivare una CA subordinata a livello di codice

Questo esempio di Java mostra come attivare una CA subordinata utilizzando le seguenti azioni CA privata AWS API:

- 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

Il seguente esempio di Java mostra come utilizzare l'CreateCerticateAuthorityoperazione.

L'operazione crea un'autorità di certificazione (CA) subordinata privata. È necessario specificare la configurazione CA, la configurazione di revoca, il tipo di CA e un token di idempotenza opzionale.

La configurazione CA specifica le seguenti informazioni:

- Il nome dell'algoritmo e la dimensione della chiave da utilizzare per creare la chiave privata CA
- Il tipo di algoritmo di firma che la CA utilizza per firmare
- Informazioni sull'oggetto X.500

La configurazione CRL specifica le seguenti informazioni:

- Il periodo di scadenza del CRL in giorni (il periodo di validità del CRL)
- Il bucket Amazon S3 che conterrà il CRL
- Un alias CNAME per il bucket S3 che è incluso nei certificati emessi dalla CA

Se eseguita correttamente, questa funzione restituisce l'Amazon Resource Name (ARN) della 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);
    }
}
```

L'output visualizzato dovrebbe essere simile al seguente:

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

Utilizzo CreateCertificateAuthority per supportare Active Directory

Il seguente esempio di Java mostra come utilizzare l'CreateCerticateAuthorityoperazione per creare una CA che può essere installata nell'archivio Enterprise NTAuth di Microsoft Active Directory (AD).

L'operazione crea un'autorità di certificazione (CA) principale privata utilizzando identificatori di oggetti (OID) personalizzati. Per ulteriori informazioni e un AWS CLI esempio di operazione equivalente, vedere Creare una CA per l'accesso ad Active Directory.

Se eseguita correttamente, questa funzione restituisce l'Amazon Resource Name (ARN) della 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);
    }
}
```

L'output visualizzato dovrebbe essere simile al seguente:

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

CreateCertificateAuthorityAuditReport

Il seguente esempio di Java mostra come utilizzare l'CreateCertificateAuthorityAuditReportoperazione.

L'operazione crea un report di audit che elenca tutte le volte in cui un certificato viene emesso o revocato. Il report viene salvato nel bucket Amazon S3 specificato in input. È possibile generare un nuovo report ogni 30 minuti.

```
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);

    }
}
```

L'output visualizzato dovrebbe essere simile al seguente:

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

CreatePermission

Il seguente esempio di Java mostra come utilizzare l'CreatePermissionoperazione.

L'operazione assegna le autorizzazioni di accesso da una CA privata a un'entità servizio AWS designata. Ai servizi può essere concessa l'autorizzazione per creare e recuperare certificati da una CA privata, nonché elencare le autorizzazioni attive concesse dalla CA privata. Per rinnovare automaticamente i certificati tramite ACM, è necessario assegnare tutte le autorizzazioni possibili (`IssueCertificateGetCertificate`, e`ListPermissions`) dalla CA al principale del servizio ACM (). `acm.amazonaws.com` È possibile trovare l'ARN di una CA chiamando la ListCertificateAuthoritiesfunzione.

Una volta creata un'autorizzazione, è possibile ispezionarla con la ListPermissionsfunzione o eliminarla con la DeletePermissionfunzione.

```
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

Il seguente esempio di Java mostra come utilizzare l'DeleteCertificateAuthorityoperazione.

Questa operazione elimina l'autorità di certificazione (CA) privata creata utilizzando l'CreateCertificateAuthorityoperazione. Per l'operazione `DeleteCertificateAuthority` è necessario fornire un ARN per l'eliminazione della CA. È possibile trovare l'ARN chiamando l'ListCertificateAuthoritiesoperazione. Puoi eliminare la CA privata immediatamente se il suo stato è `CREATING` o `PENDING_CERTIFICATE`. Tuttavia, se hai già importato il certificato, non puoi eliminarla immediatamente. È necessario innanzitutto disabilitare la CA chiamando l'UpdateCertificateAuthorityoperazione e impostare il `Status` parametro su. `DISABLED` È quindi possibile utilizzare il parametro `PermanentDeletionTimeInDays` nell'operazione `DeleteCertificateAuthority` per specificare il numero di giorni, da 7 a 30. Durante tale periodo è possibile ripristinare lo stato `disabled` della CA privata. Per impostazione predefinita, se non imposti il parametro `PermanentDeletionTimeInDays`, il periodo di ripristino è di 30 giorni. Al termine di tale periodo, la CA privata viene eliminata definitivamente e non potrà essere ripristinata. Per ulteriori informazioni, consulta Ripristino di una CA.

Per un esempio in Java che mostra come utilizzare l'RestoreCertificateAuthorityoperazione, vedereRestoreCertificateAuthority.

```
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

Il seguente esempio di Java mostra come utilizzare l'DeletePermissionoperazione.

L'operazione elimina le autorizzazioni che una CA privata ha delegato a un AWS service principal utilizzando l'operazione. CreatePermissions È possibile trovare l'ARN di una CA chiamando la ListCertificateAuthoritiesfunzione. È possibile controllare le autorizzazioni concesse da una CA chiamando la funzione. 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

Il seguente esempio di Java mostra come utilizzare l'DeletePolicyoperazione.

L'operazione elimina la politica basata sulle risorse allegata a una CA privata. Una politica basata sulle risorse viene utilizzata per abilitare la condivisione tra account CA. Puoi trovare l'ARN di una CA privata chiamando l'azione ListCertificateAuthorities.

Le azioni API correlate includono PutPolicye GetPolicy.

```
package com.amazonaws.samples;

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

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

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

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

import java.util.ArrayList;

public class CreatePermission {

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

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

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

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

      // Create a request object.
      CreatePermissionRequest req =
          new CreatePermissionRequest();

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

      // Set the permissions to give the user.
      ArrayList<String> permissions = new ArrayList<>();
      permissions.add("IssueCertificate");
      permissions.add("GetCertificate");
      permissions.add("ListPermissions");

      req.setActions(permissions);

      // Set the AWS principal.
      req.setPrincipal("acm.amazonaws.com");
```

```
      // Create a result object.
      CreatePermissionResult result = null;
      try {
         result = client.createPermission(req);
      } catch (InvalidArnException ex) {
         throw ex;
      } catch (InvalidStateException ex) {
         throw ex;
      } catch (LimitExceededException ex) {
         throw ex;
      } catch (PermissionAlreadyExistsException ex) {
         throw ex;
      } catch (RequestFailedException ex) {
         throw ex;
      } catch (ResourceNotFoundException ex) {
         throw ex;
      }
   }
}
```

DescribeCertificateAuthority

Il seguente esempio di Java mostra come utilizzare l'DescribeCertificateAuthorityoperazione.

L'operazione consente di elencare le informazioni relative all'autorità di certificazione (CA) privata. È necessario specificare l'ARN (Amazon Resource Name) della CA privata. L'output contiene lo stato della CA, che può essere uno dei seguenti:

- `CREATING`— CA privata AWS sta creando la tua autorità di certificazione privata.
- `PENDING_CERTIFICATE`— Il certificato è in sospeso. È necessario utilizzare la CA subordinata o root locale per firmare la CSR della CA privata e quindi importarla in PCA.
- `ACTIVE`— La tua CA privata è attiva.
- `DISABLED`— La tua CA privata è stata disabilitata.
- `EXPIRED`— Il tuo certificato CA privato è scaduto.
- `FAILED`— La tua CA privata non può essere creata.
- `DELETED`— La CA privata rientra nel periodo di ripristino, dopodiché verrà eliminata definitivamente.

```
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

Il seguente esempio di Java mostra come utilizzare l'DescribeCertificateAuthorityAuditReportoperazione.

L'operazione elenca le informazioni su uno specifico rapporto di controllo creato chiamando l'CreateCertificateAuthorityAuditReportoperazione. Ogni volta che viene utilizzata la chiave privata dell'autorità di certificazione (CA), vengono create le informazioni di audit. La chiave privata viene utilizzata quando si emette un certificato, si firma un CRL o si revoca un certificato.

```
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);
    }
}
```

L'output visualizzato dovrebbe essere simile al seguente:

```
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

Il seguente esempio di Java mostra come utilizzare l'GetCertificateoperazione.

L'operazione recupera un certificato dalla CA privata. L'ARN del certificato viene restituito quando si chiama l'IssueCertificateoperazione. Devi specificare sia l'ARN della CA privata sia l'ARN del certificato emesso quando chiami l'operazione `GetCertificate`. È possibile recuperare il certificato se è nello stato `ISSUED`. È possibile chiamare l'CreateCertificateAuthorityAuditReportoperazione per creare un rapporto che contenga informazioni su tutti i certificati emessi e revocati dalla CA privata.

```
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);
```

```
    }
}
```

L'output deve essere una catena di certificati simile alla seguente per l'autorità di certificazione (CA) e per il certificato specificato.

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

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

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

GetCertificateAuthorityCertificate

Il seguente esempio di Java mostra come utilizzare l'GetCertificateAuthorityCertificateoperazione.

Questa operazione consente di recuperare il certificato e la catena di certificati per l'autorità di certificazione (CA) privata. Sia il certificato che la catena sono stringhe con codifica base64 in formato PEM. La catena non include il certificato CA. Ogni certificato nella catena firma il certificato precedente.

```
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);
    }
}
```

L'output deve essere un certificato e una catena di certificati simili ai seguenti per l'autorità di certificazione (CA) specificata.

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

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

GetCertificateAuthorityCsr

Il seguente esempio di Java mostra come utilizzare l'GetCertificateAuthorityCsroperazione.

Questa operazione consente di recuperare la richiesta di firma del certificato (CSR) per l'autorità di certificazione (CA) privata. La CSR viene creata quando si chiama l'CreateCertificateAuthorityoperazione. Porta la CSR nell'infrastruttura locale X.509 e firmala utilizzando la CA root o subordinata. Quindi importa nuovamente il certificato firmato in ACM PCA richiamando l'operazione. ImportCertificateAuthorityCertificate Il CSR viene restituito come stringa con codifica Base64 in formato 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);
   }
}
```

L'output deve essere simile al seguente per l'autorità di certificazione (CA) specificata. La richiesta di firma del certificato (CSR) è con codifica base64 nel formato PEM. Salvala in un file locale, portala nell'infrastruttura locale X.509 e firmala utilizzando la CA root o subordinata.

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

GetPolicy

Il seguente esempio di Java mostra come utilizzare l'GetPolicyoperazione.

L'operazione recupera la policy basata sulle risorse allegata a una CA privata. Viene utilizzata una politica basata sulle risorse per abilitare la condivisione tra account CA. Puoi trovare l'ARN di una CA privata chiamando l'azione ListCertificateAuthorities.

Le azioni API correlate includono PutPolicye 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

Il seguente esempio di Java mostra come utilizzare l'ImportCertificateAuthorityCertificateoperazione.

Questa operazione consente di importare il certificato emesso da una CA privata firmato in CA privata AWS. Prima di poter chiamare questa operazione, è necessario creare l'autorità di certificazione privata chiamando l'CreateCertificateAuthorityoperazione. È quindi necessario generare una richiesta di firma del certificato (CSR) chiamando l'GetCertificateAuthorityCsroperazione. Portare la CSR nella CA locale e firmarla utilizzando il certificato root o un certificato subordinato. Creare una catena di certificati e copiare il certificato firmato e la catena di certificati nella directory di lavoro.

```
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

Il seguente esempio di Java mostra come utilizzare l'IssueCertificateoperazione.

Questa operazione utilizza l'autorità di certificazione (CA) privata per emettere un certificato di entità finale. Questa operazione restituisce l'Amazon Resource Name (ARN) del certificato. È possibile recuperare il certificato chiamando GetCertificatee specificando l'ARN.

> **Note**
>
> L'IssueCertificateoperazione richiede di specificare un modello di certificato. Questo esempio utilizza il `EndEntityCertificate/V1` modello. Per informazioni su tutti i modelli disponibili, vedereInformazioni sui modelli di certificato.

```
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);
    }
}
```

L'output visualizzato dovrebbe essere simile al seguente:

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

ListCertificateAuthorities

Il seguente esempio di Java illustra come utilizzareListCertificateAuthoritiesOperazione

Questa operazione elenca le autorità di certificazione private (CA) che hai creato utilizzandoCreateCertificateAuthorityOperazione

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

Se sono presenti delle autorità di certificazione da elencare, l'output deve essere simile al seguente.

```
[{
 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

Il seguente esempio di Java mostra come utilizzare l'ListPermissionsoperazione.

Questa operazione elenca le eventuali autorizzazioni assegnate dalla CA privata. Le autorizzazioni, tra cui `IssueCertificateGetCertificate`, e`ListPermissions`, possono essere assegnate a un responsabile del AWS servizio con l'CreatePermissionoperazione e revocate con l'DeletePermissionsoperazione.

```
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);
   }
}
```

Se la CA privata designata ha assegnato autorizzazioni a un'entità servizio, l'output deve essere simile al seguente:

```
[{
       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

Il seguente esempio di Java mostra come utilizzare l'ListTagsoperazione.

Questa operazione elenca i tag, se presenti, che sono associati alla CA privata. I tag sono etichette che è possibile utilizzare per identificare e organizzare le CA. Ciascun tag è formato da una chiave e

da un valore facoltativo. Chiama l'TagCertificateAuthorityoperazione per aggiungere uno o più tag alla tua CA. Richiama l'UntagCertificateAuthorityoperazione per rimuovere i 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);
    }
}
```

Se sono presenti tag da elencare, l'output deve essere simile al seguente.

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

PutPolicy

Il seguente esempio di Java mostra come utilizzare l'PutPolicyoperazione.

L'operazione associa una policy basata sulle risorse a una CA privata, abilitando la condivisione tra account. Se autorizzato da una politica, un principale residente in un altro AWS account può emettere e rinnovare certificati privati di entità finale utilizzando una CA privata di cui non è titolare. Puoi trovare l'ARN di una CA privata chiamando l'azione ListCertificateAuthorities. Per esempi di politiche, consulta la CA privata AWS guida sulle politiche basate sulle risorse.

Una volta allegata una policy a una CA, è possibile esaminarla con l'GetPolicyazione o eliminarla con l'azione. 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.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

Il seguente esempio di Java mostra come utilizzare l'RestoreCertificateAuthorityoperazione. Una CA privata può essere ripristinata in qualsiasi momento durante il periodo di ripristino. Al momento, questo periodo può durare da 7 a 30 giorni a partire dalla data di eliminazione e può essere definito al momento dell'eliminazione della CA. Per ulteriori informazioni, consulta Ripristino di una CA. Vedi anche l'esempio 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

Il seguente esempio di Java mostra come utilizzare l'RevokeCertificateoperazione.

Questa operazione revoca un certificato emesso chiamando l'IssueCertificateoperazione. Se hai abilitato un elenco di revoca dei certificati (CRL) quando hai creato o aggiornato la tua CA privata, le informazioni sui certificati revocati sono incluse nel CRL. CA privata AWSscrive il CRL in un bucket Amazon S3 specificato. Per ulteriori informazioni, consulta la struttura. CrlConfiguration

```
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

Il seguente esempio di Java mostra come utilizzare l'TagCertificateAuthorityoperazione.

Questa funzione consente di aggiungere uno o più tag alla CA privata. I tag sono etichette che possono essere utilizzate per identificare e organizzare le risorse AWS. Ciascun tag è formato da una chiave e da un valore facoltativo. Quando chiami questa operazione, specifica la CA privata in base al suo Amazon Resource Name (ARN). Il tag deve essere specificato utilizzando una coppia chiave-valore. Per identificare una caratteristica specifica di tale CA, è possibile applicare un tag a una sola CA privata. In alternativa, per filtrare una relazione comune tra tali CA, è possibile applicare lo stesso tag a più CA private. Per rimuovere uno o più tag, utilizzare l'UntagCertificateAuthorityoperazione. Chiama l'ListTagsoperazione per vedere quali tag sono associati alla tua 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.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

Il seguente esempio di Java mostra come utilizzare l'UntagCertificateAuthorityoperazione.

Questa operazione consente di rimuovere uno o più tag dalla CA privata. Ciascun tag è costituito da una coppia chiave-valore. Se non si specifica la parte valore del tag quando si richiama questa operazione, il tag verrà rimosso indipendentemente dal valore. Se si specifica un valore, il tag viene rimosso solo se è associato al valore specificato. Per aggiungere tag a una CA privata, utilizzare l'TagCertificateAuthorityoperazione. Chiama l'ListTagsoperazione per vedere quali tag sono associati alla tua CA.

```
package com.amazonaws.samples;

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

import java.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

Il seguente esempio di Java mostra come utilizzare l'UpdateCertificateAuthorityoperazione.

L'operazione aggiorna lo stato o la configurazione di un'autorità di certificazione privata (CA). La CA privata deve essere nello stato `ACTIVE` o `DISABLED` prima di poterla aggiornare. È possibile disabilitare una CA privata che si trova nello stato `ACTIVE` o rendere di nuovo attiva una CA che si trova nello stato `DISABLED`.

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.UpdateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.model.CertificateAuthorityStatus;

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

```
import com.amazonaws.services.acmpca.model.ResourceNotFoundException;
import com.amazonaws.services.acmpca.model.InvalidArgsException;
import com.amazonaws.services.acmpca.model.InvalidArnException;
import com.amazonaws.services.acmpca.model.InvalidStateException;
import com.amazonaws.services.acmpca.model.InvalidPolicyException;
import com.amazonaws.services.acmpca.model.CrlConfiguration;
import com.amazonaws.services.acmpca.model.RevocationConfiguration;

public class UpdateCertificateAuthority {

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

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

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

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

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

      // Set the ARN of the private CA that you want to update.
      req.setCertificateAuthorityArn("arn:aws:acm-pca:us-
east-1:111122223333:certificate-authority/11223344-1234-1122-2233-112233445566");

      // Define the certificate revocation list configuration. If you do not want to
```

```
        // update the CRL configuration, leave the CrlConfiguration structure alone and
        // do not set it on your UpdateCertificateAuthorityRequest object.
        CrlConfiguration crlConfigure = new CrlConfiguration();
        crlConfigure.withEnabled(true);
        crlConfigure.withExpirationInDays(365);
        crlConfigure.withCustomCname("your-custom-name");
        crlConfigure.withS3BucketName("your-bucket-name");

        // Set the CRL configuration onto your UpdateCertificateAuthorityRequest object.
        // If you do not want to change your CRL configuration, do not use the
        // setCrlConfiguration method.
        RevocationConfiguration revokeConfig = new RevocationConfiguration();
        revokeConfig.setCrlConfiguration(crlConfigure);
        req.setRevocationConfiguration(revokeConfig);

        // Set the status.
        req.withStatus(CertificateAuthorityStatus.<<ACTIVE>>);

        // Create the result object.
        try {
            client.updateCertificateAuthority(req);
        } catch (ConcurrentModificationException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        } catch (InvalidArgsException ex) {
            throw ex;
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (InvalidStateException ex) {
            throw ex;
        } catch (InvalidPolicyException ex) {
            throw ex;
        }
    }
}
```

Crea CA e certificati con nomi di oggetto personalizzati

IlCustomAttributel'oggetto consente agli amministratori di passare identificatori di oggetti personalizzati (OID) a CA e certificati privati. Gli OID personalizzati possono essere utilizzati per creare gerarchie specializzate di nomi di argomenti che riflettano la struttura e le

esigenze dell'organizzazione. I certificati personalizzati devono essere creati utilizzando uno deiApiPassthroughmodelli. Per ulteriori informazioni sui modelli, consulta Varietà di modelli. Per ulteriori informazioni sull'utilizzo di attributi personalizzati, consultaEmissione di certificati privati per entità finalieProcedura per la creazione di una CA (CLI) .

Non puoi utilizzareStandardAttributesin collaborazione conCustomAttributes. Tuttavia, è possibile passare gli OID standard come parte diCustomAttributes. I nomi di oggetto di default (OID) sono elencati nella tabella seguente:

Nome di Subject	ID dell'oggetto
Paese	2.5.4.6
CommonName	2.5.4.3
DistinguishedNameQualifier	2.5.4.46
GenerationQualifier	2.5.4.44
GivenName	2.5.4.42
Initials	2.5.4.43
Locality	2.5.4.7
Organizzazione	2.5.4.10
OrganizationalUnit	2.5.4.11
Pseudonym	2.5.4.65
SerialNumber	2.5.4.5
Stato	2.5.4.8
Surname	2.5.4.4
Title	2.5.4.12

Argomenti

Crea CA con 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);
    }
}
```

Emetti un certificato con 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);
    }
```

```
}
```

Crea certificati con estensioni personalizzate

IlCustomExtensionl'oggetto consente agli amministratori di impostare estensioni X.509 personalizzate nei certificati privati. I certificati personalizzati devono essere creati utilizzando uno dei ApiPassthrough Modelli. Per ulteriori informazioni sui modelli, consulta Varietà di modelli. Per ulteriori informazioni sull'utilizzo di estensioni personalizzate, consultaEmissione di certificati privati per entità finali.

Argomenti

- Attiva una CA subordinata con il NameConstraints estensione
- Emetti un certificato con l'estensione della dichiarazione QC

Attiva una CA subordinata con il NameConstraints estensione

```
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);
    }
}
```

Emetti un certificato con l'estensione della dichiarazione 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);
```

```
    }
}
```

Utilizzo dell'CA privata AWSAPI per implementare lo standard Matter (esempi Java)

Puoi utilizzare l'AWS Private Certificate AuthorityAPI per creare certificati conformi allo standard di connettività Matter. Matter specifica configurazioni di certificati che migliorano la sicurezza e la coerenza dei dispositivi Internet of Things (IoT) su più piattaforme di progettazione. Per ulteriori informazioni su Matter, vedere buildwithmatter.com.

Gli esempi Java in questa sezione interagiscono con il servizio inviando richieste HTTP. Il servizio restituisce risposte HTTP. Per ulteriori informazioni, consulta AWS Private Certificate AuthorityAPI Reference.

Oltre all'API HTTP, puoi utilizzare gli SDK AWS e gli strumenti a riga di comando per interagire con CA privata AWS. Questa modalità è consigliata rispetto all'API HTTP. Per ulteriori informazioni, consulta Strumenti per Amazon Web Services. I seguenti argomenti illustrano come utilizzare AWS SDK for Java per programmare l'API CA privata AWS.

I GetCertificateAuthorityCsrcamerieri di supporto DescribeCertificateAuthorityAuditReportalle operazioni e alle operazioni. GetCertificate È possibile utilizzare i waiter per controllare la progressione del codice in base alla presenza o allo stato di determinate risorse. Per ulteriori informazioni, consulta i seguenti argomenti e la sezione dedicata ai camerieri AWS SDK for Java nel blog per sviluppatori. AWS

Matter 1.2, rilasciato nell'ottobre 2023, supporta la revoca del DAC utilizzando gli elenchi di revoca dei certificati (CRL). Per aiutarvi a conformarvi all'attuale standard Matter, quando abilitate la revoca CRL per le CA che emettono certificati Matter, nell'`CrlConfiguration`oggetto, nella struttura, impostate su. `CrlDistributionPointExtensionConfiguration` `OmitExtension` `true`

In genere, le CA incorporano il CRL Distribution Point (CDP) nei certificati che emettono in modo che le parti che effettuano la convalida della catena di certificati possano recuperare il CRL e verificare lo stato del certificato. In Matter, l'URI CDP non è scritto nei certificati. Invece, gli utenti recuperano i CDP dal Matter Distributed Compliance Ledger (DCL), l'affidabile archivio dati Matter. È necessario caricare l'URI CDP nel Matter DCL in modo che possa essere scoperto durante la convalida dei DAC. Per ulteriori informazioni sulla determinazione dell'URI CDP, vedere. Determinazione dell'URI del punto di distribuzione CRL (CDP) Per ulteriori informazioni su Matter, consulta la documentazione di Matter DCL.

Argomenti

- Attiva un'autorità di attestazione del prodotto (PAA)
- Attiva un PAI (Product Attestation Intermediate)
- Creare un certificato di attestazione del dispositivo (DAC)
- Attiva una CA principale per i certificati operativi dei nodi (NOC).
- Attivazione di una CA subordinata per i certificati operativi dei nodi (NOC)
- Creare un certificato operativo del nodo (NOC)

Attiva un'autorità di attestazione del prodotto (PAA)

Questo esempio di Java mostra come utilizzare il Definizione RootCACertificate_APIPassThrough/v1 modello per creare e installare un certificato Matter Root CA (PAA) per l'attestazione del prodotto. L'estensione AuthorityKeyIdentifier (AKI) è facoltativa per PaaS. Per impostare un AKI, è necessario generare un valore AKI con codifica Base64 e passarlo attraverso un. CustomExtension

L'esempio richiama le seguenti azioni API: CA privata AWS

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

In caso di problemi, consulta Utilizzo dello standard Matter la sezione Risoluzione dei problemi.

```
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);
    }
}
```

Attiva un PAI (Product Attestation Intermediate)

Questo esempio di Java mostra come utilizzare il BlankSubordinateDefinizione CACertificate_ PathLen 0_APIPassThrough/v1 modello per creare e installare un certificato Matter Subordinate CA (PAI) per l'attestazione del prodotto. È necessario generare un valore con codifica Base64 e passarlo tramite KeyUsage un. CustomExtension

L'esempio richiama le seguenti azioni API: CA privata AWS

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

- GetCertificateAuthorityCertificate

In caso di problemi, consulta Utilizzo dello standard Matter la sezione Risoluzione dei problemi.

```
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);
    }
}
```

Creare un certificato di attestazione del dispositivo (DAC)

Questo esempio di Java mostra come utilizzare il modello BlankEndEntityCertificate_ CriticalBasicConstraints _ApiPassthrough/v1 per creare un certificato di attestazione del dispositivo Matter. È necessario generare un valore con codifica Base64 e passarlo tramite un. KeyUsage CustomExtension

L'esempio richiama la seguente azione API: CA privata AWS

- IssueCertificate

In caso di problemi, consulta Utilizzo dello standard Matter la sezione Risoluzione dei problemi.

```
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);
    }
}
```

Attiva una CA principale per i certificati operativi dei nodi (NOC).

Questo esempio di Java mostra come utilizzare il Definizione RootCACertificate_APIPassThrough/v1 modello per creare e installare un certificato Matter Root CA per emettere NOC. L'estensione AuthorityKeyIdentifier (AKI) è facoltativa per i certificati NOC Root CA. Per impostare un AKI, è necessario generare un valore AKI con codifica Base64 e passarlo tramite un. CustomExtension

L'esempio richiama le seguenti azioni API: CA privata AWS

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

In caso di problemi, consulta Utilizzo dello standard Matter la sezione Risoluzione dei problemi.

```
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);
    }
}
```

Attivazione di una CA subordinata per i certificati operativi dei nodi (NOC)

Questo esempio di Java mostra come utilizzare il BlankSubordinateDefinizione CACertificate_ PathLen 0_APIPassThrough/v1 modello per emettere e installare un certificato CA Matter Subordinate per emettere NOC. È necessario generare un valore con codifica Base64 e KeyUsage passarlo tramite un. CustomExtension

L'esempio richiama le seguenti azioni API: CA privata AWS

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

Se si verificano problemi, Utilizzo dello standard Matter consulta la sezione Risoluzione dei problemi.

```
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);
    }
}
```

Creare un certificato operativo del nodo (NOC)

Questo esempio di Java mostra come utilizzare il modello BlankEndEntityCertificate_ CriticalBasicConstraints _ApiPassThrough/v1 per creare un certificato operativo Matter Node.
È necessario generare un valore con codifica KeyUsage Base64 e passarlo tramite un. CustomExtension

L'esempio richiama la seguente azione API: CA privata AWS

- IssueCertificate

In caso di problemi, consulta Utilizzo dello standard Matter la sezione Risoluzione dei problemi.

```
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);
   }
}
```

Utilizzo dell'CA privata AWSAPI per implementare lo standard della patente di guida mobile (mDL) (esempi Java)

È possibile utilizzare l'AWS Private Certificate AuthorityAPI per creare certificati conformi allo standard ISO/IEC per la patente di guida mobile (mDL). Questo standard stabilisce le specifiche di interfaccia per l'implementazione di una patente di guida in associazione a un dispositivo mobile, comprese le configurazioni dei certificati.

Gli esempi Java in questa sezione interagiscono con il servizio inviando richieste HTTP. Il servizio restituisce risposte HTTP. Per ulteriori informazioni, consulta la Documentazione di riferimento delle API di AWS Private Certificate Authority.

Oltre all'API HTTP, puoi anche utilizzare gli AWS SDK e AWS CLI gli strumenti per la gestioneCA privata AWS. Ti consigliamo di utilizzare l'SDK o l'AWS CLIAPI HTTP. Per ulteriori informazioni, consulta Strumenti per Amazon Web Services. I seguenti argomenti illustrano come utilizzare AWS SDK for Java per programmare l'API CA privata AWS.

I `GetCertificateAuthorityCsr`camerieri `GetCertificate`di supporto `DescribeCertificateAuthorityAuditReport`alle operazioni e alle operazioni. È possibile utilizzare i waiter per controllare la progressione del codice in base alla presenza o allo stato di determinate risorse. Per ulteriori informazioni, consultate i seguenti argomenti e Waiters in the AWS SDK for Java nel Developer Blog. AWS

Argomenti

- Attivare un certificato di autorità di certificazione dell'autorità emittente (IACA)
- Creare un certificato per il firmatario di un documento

Attivare un certificato di autorità di certificazione dell'autorità emittente (IACA)

Questo esempio di Java mostra come utilizzare il BlankRootDefinizione CACertificate_ PathLen 0_APIPassThrough/v1 modello per creare e installare un certificato IACA (Issuing Authority Certificate Authority) conforme allo standard ISO/IEC mDL. È necessario generare valori con codifica base64 per, e e trasmetterli. `KeyUsage` `IssuerAlternativeName` `CRLDistributionPoint` `CustomExtensions`

> **Note**
>
> Il certificato di collegamento IACA stabilisce un percorso di fiducia dal vecchio certificato radice IACA al nuovo certificato radice IACA. L'autorità emittente può generare e distribuire un certificato di collegamento IACA durante il processo di riassegnazione della chiave IACA. Non è possibile emettere un certificato di collegamento IACA utilizzando un certificato radice IACA con set. `pathLen=0`

L'esempio richiama le seguenti azioni CA privata AWS API:

- 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);
    }
}
```

Creare un certificato per il firmatario di un documento

Questo esempio di Java mostra come utilizzare il modello BlankEndEntityCertificate_ApiPassThrough/v1 per creare un certificato di firma di documenti conforme allo standard ISO/IEC mDL. È necessario `KeyUsage` generare `IssuerAlternativeName` valori con codifica in `CRLDistributionPoint` `CustomExtensions` base64 per, e trasmetterli.

L'esempio richiama la seguente azione API: CA privata AWS

- 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);
    }
}
```

Certificati CA privati firmati esternamente

Se la root di attendibilità della gerarchia CA privata deve essere una CA esterna a CA privata AWS, è possibile creare e autofirmare la propria CA root. In alternativa, è possibile ottenere un certificato emesso da una CA privato firmato da una CA privata esterna gestita dall'organizzazione. Qualunque sia la fonte, puoi utilizzare questa CA ottenuta esternamente per firmare un certificato CA privato subordinato che gestisca. CA privata AWS

> **Note**
>
> Le procedure per creare o ottenere un fornitore esterno di servizi fiduciari non rientrano nell'ambito di questa guida.

L'utilizzo di una CA principale esterna che CA privata AWS consente di applicare i vincoli relativi ai nomi CA, come definito nella sezione Name Constraints della RFC 5280. I vincoli di nome consentono agli amministratori della CA di limitare i nomi dei soggetti nei certificati.

Se si prevede di firmare un certificato CA privato subordinato con una CA esterna, ci sono tre attività da completare prima di avere una CA funzionante in: CA privata AWS

1. Genera una richiesta di firma del certificato (CSR).
2. Invia la CSR all'autorità di firma esterna e restituiscila con un certificato firmato e una catena di certificati.
3. Installa un certificato firmato inCA privata AWS.

Le procedure seguenti descrivono come completare queste attività utilizzando AWS Management Console oAWS CLI.

Per ottenere e installare un certificato CA con firma esterna (console)

1. (Facoltativo) Se non sei già nella pagina dei dettagli della CA, apri la CA privata AWS console all'indirizzo https://console.aws.amazon.com/acm-pca/home. Nella pagina Autorità di certificazione private, scegli una CA subordinata con lo stato Certificato in sospeso, Attivo, Disabilitato o Scaduto.
2. Scegli Azioni, Installa certificato CA per aprire la pagina Installa certificato CA subordinato.

3. Nella pagina Installa certificato CA subordinato, in Seleziona il tipo di CA, scegli CA privata esterna.
4. In CSR per questa CA, la console visualizza il testo ASCII con codifica Base64 della CSR. Puoi copiare il testo usando il pulsante Copia oppure puoi scegliere Esporta CSR in un file e salvarlo localmente.

   > **Note**
   >
   > Il formato esatto del testo CSR deve essere preservato durante il copia e incolla.

5. Se non è possibile eseguire immediatamente i passaggi offline per ottenere un certificato firmato dall'autorità di firma esterna, è possibile chiudere la pagina e tornare alla pagina dopo aver ottenuto un certificato firmato e una catena di certificati.

   Altrimenti, se sei pronto, esegui una delle seguenti operazioni:

   - Incolla il testo ASCII con codifica Base64 dell'ente del certificato e della catena di certificati nelle rispettive caselle di testo.
   - Scegliete Carica per caricare l'ente del certificato e la catena di certificati dai file locali nelle rispettive caselle di testo.
6. Scegli Conferma e installa.

Per ottenere e installare un certificato CA (CLI) con firma esterna

1. Utilizza il get-certificate-authority-csrcomando per recuperare la richiesta di firma del certificato (CSR) per la tua CA privata. Se si desidera inviare la CSR per la visualizzazione, utilizzare l'opzione `--output text` per eliminare i caratteri CR/LF dalla fine di ogni riga. Per inviare la CSR a un file, utilizzare l'opzione di reindirizzamento (**>**) seguita da un nome 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
   ```

   Dopo aver salvato una CSR come file locale, puoi ispezionarla utilizzando il seguente comando OpenSSL:

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

Il comando precedente genera un output simile al seguente. Tieni presente che l'estensione CA è TRUE, il che indica che la CSR è per un certificato emesso da una 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. Inviate la CSR all'autorità di firma esterna e ottenete i file contenenti il certificato firmato e la catena di certificati con codifica PEM Base64.
3. Utilizzate il import-certificate-authority-certificatecomando per importare il file di certificato CA privato e il file chain in. CA privata AWS

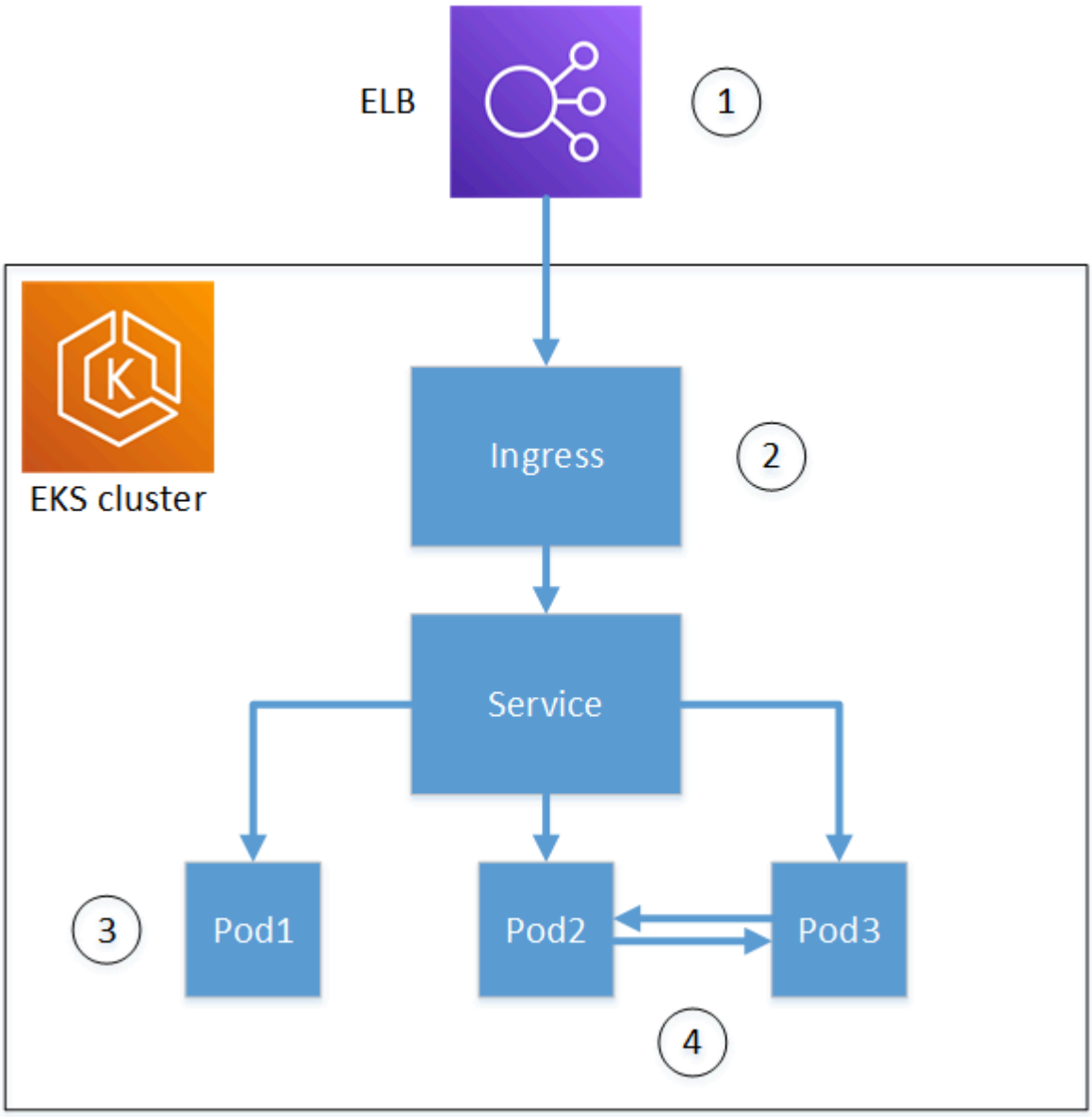
```
$ aws acm-pca import-certificate-authority-certificate \
--certificate-authority-arn arn:aws:acm-pca:region:account:\
certificate-authority/12345678-1234-1234-1234-123456789012 \
--certificate file://C:\example_ca_cert.pem \
--certificate-chain file://C:\example_ca_cert_chain.pem
```

Proteggere Kubernetes con CA privata AWS

I contenitori e le applicazioni Kubernetes utilizzano certificati digitali per fornire autenticazione e crittografia sicure tramite TLS. Una soluzione ampiamente adottata per la gestione del ciclo di vita dei certificati TLS in Kubernetes è cert-manager, un componente aggiuntivo di Kubernetes che richiede i certificati, li distribuisce nei contenitori Kubernetes e automatizza il rinnovo dei certificati.

CA privata AWSfornisce un plug-in open source per cert-manager, per gli utenti di cert-manager che desiderano configurare una CA senza archiviare chiavi private nel cluster. aws-privateca-issuer Gli utenti con requisiti normativi per il controllo dell'accesso e la verifica delle operazioni della CA possono utilizzare questa soluzione per migliorare la verificabilità e supportare la conformità. Puoi utilizzare il plug-in AWS Private CA Issuer con Amazon Elastic Kubernetes Service (Amazon EKS), un Kubernetes AWS autogestito su o in Kubernetes locale.

Il diagramma seguente mostra alcune delle opzioni disponibili per l'utilizzo di TLS in un cluster Amazon EKS. Questo cluster di esempio, contenente varie risorse, si trova dietro un sistema di bilanciamento del carico. I numeri identificano i possibili endpoint per le comunicazioni protette da TLS, tra cui il sistema di bilanciamento del carico esterno, il controller di ingresso, un singolo pod all'interno di un servizio e un paio di pod che comunicano in modo sicuro tra loro.



1. Interruzione presso il sistema di bilanciamento del carico.

   Elastic Load Balancing (ELB) è un servizio AWS Certificate Manager integrato, il che significa che puoi fornire ad ACM una CA privata, firmare un certificato con essa e installarlo utilizzando la console ELB. Questa soluzione fornisce comunicazioni crittografate tra un client remoto e il sistema di bilanciamento del carico. I dati vengono passati non crittografati al cluster EKS. In alternativa, è possibile fornire un certificato privato a un sistema diverso dal sistema di AWS bilanciamento del carico per terminare TLS.
2. Terminazione presso il controller di ingresso Kubernetes.

Il controller di ingresso risiede all'interno del cluster EKS come sistema di bilanciamento del carico e router nativi. Se hai installato sia cert-manager che e hai fornito al cluster una CA privata aws-privateca-issuer, Kubernetes può installare un certificato TLS firmato sul controller, consentendogli di fungere da endpoint del cluster per le comunicazioni esterne. Le comunicazioni tra il load balancer e il controller di ingresso sono crittografate e, dopo l'ingresso, i dati passano non crittografati alle risorse del cluster.

3. Terminazione presso un pod.

   Ogni pod è un gruppo di uno o più contenitori che condividono risorse di archiviazione e di rete. Se hai installato sia cert-manager che e hai fornito al cluster una CA privata aws-privateca-issuer, Kubernetes può installare certificati TLS firmati sui pod, se necessario. Per impostazione predefinita, una connessione TLS che termina su un pod non è disponibile per gli altri pod del cluster.

4. Comunicazioni sicure tra i pod.

   Puoi anche fornire a più pod certificati che consentano loro di comunicare tra loro. Gli scenari possibili sono i seguenti:

   - Fornitura con certificati autofirmati generati da Kubernetes. Ciò protegge le comunicazioni tra i pod, ma i certificati autofirmati non soddisfano i requisiti HIPAA o FIPS.
   - Fornitura con certificati firmati da una CA privata. Come nei numeri 2 e 3 precedenti, ciò richiede l'installazione sia di cert-manager che di una aws-privateca-issuerCA privata e il provisioning del cluster. Kubernetes può quindi installare certificati TLS firmati sui pod, se necessario.

Utilizzo del cert-manager su più account

Gli amministratori con accesso a una CA su più account possono utilizzare cert-manager per effettuare il provisioning di un cluster Kubernetes. Per ulteriori informazioni, consulta Procedure consigliate di sicurezza per l'accesso tra più account alle CA private.

> **Note**
>
> Solo alcuni modelli di CA privata AWS certificato possono essere utilizzati in scenari tra più account. the section called “Modelli di certificato supportati ”Per un elenco dei modelli disponibili, consulta.

Modelli di certificato supportati

La tabella seguente elenca i CA privata AWS modelli che possono essere utilizzati con cert-manager per effettuare il provisioning di un cluster Kubernetes.

Modelli supportati per Kubernetes	Support per l'utilizzo su più account
BlankEndEntityCertificateDefinizione _CSRPassthrough/v1	
CodeSigningCertificateDefinizione /V1	
EndEntityCertificateDefinizione /V1	✓
EndEntityClientAuthCertificateDefinizione /V1	✓
EndEntityServerAuthCertificateDefinizione /V1	✓
Definizione OCSP SigningCertificate /V1	

Soluzioni di esempio

Le seguenti soluzioni di integrazione mostrano come configurare l'accesso CA privata AWS a un cluster Amazon EKS.

- Cluster Kubernetes abilitati per TLS con e Amazon EKS CA privata AWS
- Configurazione della crittografia end-to-end TLS su Amazon EKS con il nuovo AWS Load Balancer Controller

CA privata AWS Connector per Active Directory

Cos'è AWS Private CA Connector for Active Directory

AWS Private CApuò emettere e gestire i certificati richiesti daAWS Managed Microsoft AD. Utilizzando CA privata AWS Connector for Active Directory (Connector for AD), puoi sostituire le CA aziendali o di terze parti locali con una CA privata gestita di tua proprietà, che fornisce la registrazione dei certificati a utenti, gruppi e computer gestiti da AD.

Puoi utilizzare Connector for AD per AWS Managed Microsoft AD eliminare l'infrastruttura locale migrando l'AD e l'infrastruttura a chiave pubblica nel cloud. Per i clienti che desiderano utilizzare AD in locale, questa funzionalità si integra anche AWS Private CA con Connector. AWS Managed Microsoft AD

Argomenti

- Sei un utente principiante di AD Connector?
- Accesso a Connector for AD
- Prezzi di Connector for AD

Sei un utente principiante di AD Connector?

Se sei un utente alle prime armi di Connector for AD, ti consigliamo di iniziare leggendo le seguenti sezioni:

- Cos’è CA privata AWS?
- Che cos'è AWS Directory Service?

Accesso a Connector for AD

Puoi accedere a Connector for AD tramite la AWS CLI console e le API. Puoi accedere al connettore nella console dalla console, dalla tua AWS Private CA AWS Directory Service console o cercando Connector for AD nella barra di AWS Management Console ricerca.

Prezzi di Connector for AD

Connector for AD è offerto come funzionalità senza costi aggiuntivi. CA privata AWS Paghi solo per le autorità di certificazione private e i certificati che emetti tramite loro.

Per le informazioni più recenti CA privata AWS sui prezzi, consulta AWS Private Certificate AuthorityPrezzi. Puoi anche utilizzare il calcolatore dei AWS prezzi per stimare i costi.

Guida introduttiva a AWS Private CA Connector for Active Directory

Con AWS Private CA Connector for Active Directory, puoi emettere certificati dalla tua CA privata agli oggetti Active Directory per l'autenticazione e la crittografia. Quando crei un connettore, AWS Private Certificate Authority crea per te un endpoint nel tuo VPC per gli oggetti della directory per richiedere certificati.

Per emettere certificati, create un connettore e modelli compatibili con AD per il connettore. Quando crei un modello, puoi impostare le autorizzazioni di registrazione per i tuoi gruppi AD.

Argomenti

Prerequisiti del connettore per AD

Per Connector for AD sono necessari i seguenti requisiti.

Per creare un Connector for AD, è necessario configurare una AWS Private Certificate Authority (CA) e una directory. È quindi necessario condividere la CA e la directory private con il servizio Connector for AD. È inoltre necessario impostare correttamente i gruppi di sicurezza e le politiche IAM per creare un connettore.

CA privata AWS

Imposta un CA privata AWS per l'emissione di certificati sugli oggetti della directory. Per ulteriori informazioni, consulta Amministrazione privata della CA.

CA privata AWSÈ necessario essere `Active` nello stato in cui è possibile creare un Connector for AD. Il nome del soggetto della CA privata deve includere un nome comune. La creazione del connettore avrà esito negativo se si tenta di creare un connettore utilizzando una CA privata senza un nome comune.

Active Directory

Oltre a unCA privata AWS, è necessario disporre di Active Directory in un cloud privato virtuale (VPC). Connector for AD supporta i seguenti tipi di directory offerti daAWS Directory Service:

- AWSMicrosoft Active Directory gestito: con AWS Directory Service è possibile eseguire Microsoft Active Directory (AD) come servizio gestito. AWS Directory Service for Microsoft Active Directorynoto anche comeAWS Managed Microsoft AD, è basato su Windows Server 2019. ConAWS Managed Microsoft AD, puoi eseguire carichi di lavoro compatibili con le directory in, Cloud AWS tra cui Microsoft Sharepoint e applicazioni personalizzate basate su .Net e SQL Server.
- Active Directory Connector: AD Connector è un gateway di directory in grado di reindirizzare le richieste di directory a Microsoft Active Directory locale, senza memorizzare nella cache alcuna informazione nel cloud. AD Connector supporta la connessione a un dominio ospitato su Amazon EC2

> **Note**
>
> La registrazione dei controller di dominio non è supportata quando si utilizza Connector for AD con. AWS Managed Microsoft AD

Account di servizio

Quando si utilizza il Directory Service AD Connector, è necessario delegare autorizzazioni aggiuntive all'account del servizio. Imposta l'elenco di controllo degli accessi (ACL) sull'account del servizio per consentire la possibilità di:

- Aggiungi e rimuovi un Service Principal Name (SPN) a se stesso
- Creare e aggiornare le autorità di certificazione nei seguenti container:

```
#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
```

- Crea e aggiorna un oggetto NT AuthCertificates Certification Authority (CA). Nota: se l'oggetto NT AuthCertificates CA esiste, è necessario delegare le relative autorizzazioni. Se l'oggetto non esiste, è necessario delegare la possibilità di creare oggetti secondari nel contenitore Public Key Services.

```
#objects
CN=NTAuthCertificates,CN=Public Key Services,CN=Services,CN=Configuration
```

> **Note**
>
> Se lo utilizziAWS Managed Microsoft AD, le autorizzazioni aggiuntive verranno delegate automaticamente quando autorizzerai il servizio Connector for AD con la tua directory. È possibile saltare questo passaggio preliminare.

È possibile utilizzare questo PowerShell script per delegare le autorizzazioni aggiuntive. Creerà l'oggetto dell'autorità di AuthCertifiates certificazione NT. Sostituisci «myconnectoraccount» con il nome dell'account del servizio.

```
$AccountName = 'myconnectoraccount'
# DO NOT modify anything below this comment.
# Getting Active Directory information.
Import-Module -Name 'ActiveDirectory'
$RootDSE = Get-ADRootDSE

# Getting AD Connector service account Information
$AccountProperties = Get-ADUser -Identity $AccountName
$AccountSid = New-Object -TypeName 'System.Security.Principal.SecurityIdentifier'
 $AccountProperties.SID.Value
[System.GUID]$ServicePrincipalNameGuid = (Get-ADObject -SearchBase
 $RootDse.SchemaNamingContext -Filter { lDAPDisplayName -eq 'servicePrincipalName' } -
Properties 'schemaIDGUID').schemaIDGUID
$AccountAclPath = $AccountProperties.DistinguishedName

# Getting ACL settings for AD Connector service account.
$AccountAcl = Get-ACL -Path "AD:\$AccountAclPath"
```

```
# Setting ACL allowing the AD Connector service account the ability to add and remove a
 Service Principal Name (SPN) to itself
$AccountAccessRule = New-Object -TypeName
 'System.DirectoryServices.ActiveDirectoryAccessRule' $AccountSid, 'WriteProperty',
 'Allow', $ServicePrincipalNameGuid, 'None'
$AccountAcl.AddAccessRule($AccountAccessRule)
Set-ACL -AclObject $AccountAcl -Path "AD:\$AccountAclPath"

# Add ACLs allowing AD Connector service account the ability to create certification
 authorities
[System.GUID]$CertificationAuthorityGuid = (Get-ADObject -SearchBase
 $RootDse.SchemaNamingContext -Filter { lDAPDisplayName -eq 'certificationAuthority' }
 -Properties 'schemaIDGUID').schemaIDGUID
$CAAccessRule = New-Object -TypeName
 'System.DirectoryServices.ActiveDirectoryAccessRule' $AccountSid,
 'ReadProperty,WriteProperty,CreateChild,DeleteChild', 'Allow',
 $CertificationAuthorityGuid, 'None'
$PKSDN = "CN=Public Key Services,CN=Services,CN=Configuration,
$($RootDSE.rootDomainNamingContext)"
$PKSACL = Get-ACL -Path "AD:\$PKSDN"
$PKSACL.AddAccessRule($CAAccessRule)
Set-ACL -AclObject $PKSACL -Path "AD:\$PKSDN"

$AIADN = "CN=AIA,CN=Public Key Services,CN=Services,CN=Configuration,
$($RootDSE.rootDomainNamingContext)"
$AIAACL = Get-ACL -Path "AD:\$AIADN"
$AIAACL.AddAccessRule($CAAccessRule)
Set-ACL -AclObject $AIAACL -Path "AD:\$AIADN"

$CertificationAuthoritiesDN = "CN=Certification Authorities,CN=Public Key
 Services,CN=Services,CN=Configuration,$($RootDSE.rootDomainNamingContext)"
$CertificationAuthoritiesACL = Get-ACL -Path "AD:\$CertificationAuthoritiesDN"
$CertificationAuthoritiesACL.AddAccessRule($CAAccessRule)
Set-ACL -AclObject $CertificationAuthoritiesACL -Path "AD:\$CertificationAuthoritiesDN"

$NTAuthCertificatesDN = "CN=NTAuthCertificates,CN=Public Key
 Services,CN=Services,CN=Configuration,$($RootDSE.rootDomainNamingContext)"
If (-Not (Test-Path -Path "AD:\$NTAuthCertificatesDN")) {
New-ADObject -Name 'NTAuthCertificates' -Type 'certificationAuthority' -OtherAttributes
 @{certificateRevocationList=[byte[]]'00';authorityRevocationList=[byte[]]'00';cACertificate=[b
 -Path "CN=Public Key Services,CN=Services,CN=Configuration,
$($RootDSE.rootDomainNamingContext)" }

$NTAuthCertificatesACL = Get-ACL -Path "AD:\$NTAuthCertificatesDN"
```

```
$NullGuid = [System.GUID]'00000000-0000-0000-0000-000000000000'
$NTAuthAccessRule = New-Object -TypeName
 'System.DirectoryServices.ActiveDirectoryAccessRule' $AccountSid,
 'ReadProperty,WriteProperty', 'Allow', $NullGuid, 'None'
$NTAuthCertificatesACL.AddAccessRule($NTAuthAccessRule)
Set-ACL -AclObject $NTAuthCertificatesACL -Path "AD:\$NTAuthCertificatesDN"
```

Politica IAM

Per creare un connettore per AD, hai bisogno di una policy IAM che ti consenta di creare risorse per i connettori, condividere la tua CA privata con il servizio Connector for AD e autorizzare il servizio Connector for AD con la tua directory.

Questo è un esempio di policy gestita dagli utenti:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "pca-connector-ad:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:ListCertificateAuthorities",
                "acm-pca:ListTags",
                "acm-pca:PutPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "acm-pca:IssueCertificate",
            "Resource": "*",
            "Condition": {
                "StringLike": {
```

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

Connector for AD richiede AWS RAM autorizzazioni aggiuntive, sia per l'utilizzo da console che da riga di comando.

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

Condividi CA privata AWS con Connector for AD

Dovrai condividere il tuo servizio AWS Private CA With the Connectors utilizzando la condivisione principale AWS Resource Access Manager del servizio.

Quando crei un connettore nella AWS console, la condivisione delle risorse viene creata automaticamente per te.

Quando crei una condivisione di risorse utilizzandoAWS CLI, utilizzerai il AWS RAM create-resource-share comando.

Il comando seguente crea una condivisione di risorse:

```
$  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
```

Il responsabile del servizio che chiama CreateConnector dispone delle autorizzazioni per l'emissione di certificati sul PCA. Per evitare che gli amministratori del servizio che utilizzano Connector for AD abbiano accesso generale alle tue CA privata AWS risorse, limita le loro autorizzazioni di utilizzo. `CalledVia`

Autorizza Connector for AD con la tua directory

Autorizzi il servizio Connector for AD con la tua directory in modo che il connettore possa comunicare con la tua directory. Per autorizzare il servizio Connector for AD, è necessario creare una registrazione alla directory. Per ulteriori informazioni sulla creazione di una registrazione di directory, vedere Gestione delle registrazioni degli elenchi

Gruppi di sicurezza

La comunicazione tra il tuo VPC e il connettore Connector for AD avviene tramiteAWS PrivateLink, il che richiede uno o più gruppi di sicurezza con regole in entrata che aprano le porte 443 TCP e UDP sul tuo VPC. Ti verrà richiesto questo gruppo di sicurezza quando crei un connettore. Puoi specificare la fonte come personalizzata e selezionare il blocco CIDR del tuo VPC. Puoi scegliere di limitarlo ulteriormente (ad esempio IP, CIDR e ID del gruppo di sicurezza).

Creazione di un connettore

Per istruzioni, consulta la procedura Creazione di un connettore

Configura le politiche AD

Connector for AD non è in grado di visualizzare o gestire la configurazione del Group Policy Object (GPO) del cliente. Il GPO controlla l'instradamento delle richieste AD verso il cliente CA privata AWS

o verso altri server di distribuzione di certificati o di autenticazione. Una configurazione GPO non valida può causare un instradamento errato delle richieste. Spetta ai clienti configurare e testare la configurazione di Connector for AD.

Le politiche di gruppo sono associate a un connettore e puoi scegliere di creare più connettori per un singolo AD. Spetta all'utente gestire il controllo dell'accesso a ciascun connettore se le configurazioni delle politiche di gruppo sono diverse.

La sicurezza delle chiamate sul piano dati dipende da Kerberos e dalla configurazione del VPC. Chiunque abbia accesso al VPC può effettuare chiamate sul piano dati purché sia autenticato nell'AD corrispondente. Ciò esiste al di fuori dei confini e la gestione dell'autorizzazione AWSAuth e dell'autenticazione spetta a te, il cliente.

In Active Directory, segui i passaggi seguenti per creare un GPO che punti all'URI generato quando hai creato un connettore. Questo passaggio è necessario per utilizzare Connector for AD dalla console o dalla riga di comando.

Configura i GPO.

1. Apri Server Manager sul DC
2. Vai a Strumenti e scegli Gestione delle politiche di gruppo nell'angolo in alto a destra della console.
3. Vai a Foresta > Domini. Seleziona il tuo nome di dominio e fai clic con il pulsante destro del mouse sul tuo dominio. Seleziona Crea un GPO in questo dominio, collegalo qui... e inserisci `PCA GP0` il nome.
4. Il GPO appena creato verrà ora elencato sotto il tuo nome di dominio.
5. Scegli PCA GPO e seleziona Modifica. Se si apre una finestra di dialogo con il messaggio di avviso Questo è un collegamento e le modifiche verranno propagate a livello globale, conferma il messaggio per continuare. Dovrebbe aprirsi il Group Policy Management Editor.
6. Nel Group Policy Management Editor, vai a Configurazione computer > Criteri > Impostazioni di Windows > Impostazioni di sicurezza > Politiche a chiave pubblica (scegli la cartella).
7. Vai al tipo di oggetto e scegli Certificate Services Client - Certificate Enrollment Policy
8. Nelle opzioni, modifica il modello di configurazione in Abilitato.
9. Conferma che la politica di registrazione di Active Directory sia selezionata e abilitata. Scegli Aggiungi.
10. La finestra Certificate Enrollment Policy Server dovrebbe aprirsi.

11. Immettere l'endpoint del server della politica di iscrizione del certificato generato al momento della creazione del connettore nel campo Enter enrollment server policy URI.
12. Lascia che il tipo di autenticazione sia integrato in Windows.
13. Scegli Convalida. Una volta completata la convalida, seleziona Aggiungi. La finestra di dialogo si chiude.
14. Torna a Certificate Services Client - Certificate Enrollment Policy e seleziona la casella accanto al connettore appena creato per assicurarti che il connettore sia la politica di registrazione predefinita
15. Scegli Active Directory Enrollment Policy e seleziona Rimuovi.
16. Nella finestra di dialogo di conferma, scegli Sì per eliminare l'autenticazione basata su LDAP.
17. Scegli Applica e OK nella finestra Certificate Services Client > Certificate Enrollment Policy e chiudila.
18. Vai alla cartella Public Key Policies e scegli Certificate Services Client - Auto-Enrollment.
19. Modificate l'opzione del modello di configurazione su Abilitato.
20. Conferma che le opzioni Rinnova certificati scaduti e Aggiorna certificati siano entrambe selezionate. Lascia le altre impostazioni così come sono.
21. Scegliete Applica, quindi OK e chiudete la finestra di dialogo.

Successivamente, configura le politiche a chiave pubblica per la configurazione dell'utente. Vai a Configurazione utente > Criteri > Impostazioni di Windows > Impostazioni di sicurezza > Politiche a chiave pubblica. Segui le procedure descritte dal passaggio 6 al passaggio 21 per configurare le politiche a chiave pubblica per la configurazione dell'utente.

Una volta terminata la configurazione dei GPO e delle politiche a chiave pubblica, gli oggetti del dominio richiederanno i certificati da CA privata AWS Connector for AD e otterranno i certificati emessi da. CA privata AWS

Crea un modello

Per istruzioni, consulta la procedura Creazione di un modello di connettore

Gestisci le autorizzazioni dei gruppi AD

Per istruzioni, consulta la procedura Gestione dei gruppi AD e delle autorizzazioni per i modelli

CA privata AWSConnettore per le procedure di Active Directory

Le procedure in questa sezione descrivono come creare connettori, configurare modelli e integrarsi con CA privata AWS Active Directory. È possibile eseguire queste operazioni dalla console CA privata AWS Connector for AD, utilizzando la sezione Connector for AD di o utilizzando l'API CA privata AWS Connector for AD. AWS CLI

> **Note**
>
> Sebbene CA privata AWS Connector for AD sia strettamente integrato conCA privata AWS, i due servizi dispongono di API separate. Per ulteriori informazioni, consulta l'AWS Private Certificate AuthorityAPI Reference e il CA privata AWSConnector for Active Directory API Reference.

Procedure

Creazione di un connettore

Utilizza le seguenti procedure per creare un connettore utilizzando la console, la riga di comando o l'API per AWS Private CA Connector for Active Directory.

Creazione di un connettore (console)

Completa le seguenti procedure per creare e configurare un connettore utilizzando la AWS console.

Attività

- Console aperta
- Apri il connettore Create
- Scegli o crea una directory
- Scegli una CA privata
- Configura l'etichettatura
- Rivedi e crea

Console aperta

Accedi al tuo AWS account e apri la console AWS Private CA Connector for Active Directory all'indirizzo**https://console.aws.amazon.com/pca-connector-ad/home**.

Apri il connettore Create

Nella pagina iniziale del servizio o nella pagina Connettori per Active Directory, scegli Crea connettore.

Scegli o crea una directory

Nella pagina Crea connettore CA privato per Active Directory, fornisci informazioni nella sezione Active Directory.

- In Seleziona il tipo di Active Directory, scegli uno dei due tipi disponibili:
  - AWS Directory Service for Microsoft Active Directory— Speciifica un Active Directory gestito daAWS Directory Service.
  - Active Directory locale con AWS AD Connector: utilizza AD Connector per accedere a un Active Directory ospitato in locale.
- In Seleziona la tua directory, scegli la tua directory dall'elenco.

  In alternativa, puoi scegliere Crea cartella, che apre la AWS Directory Service console in una nuova finestra. Al termine della creazione di una nuova directory, torna alla console di AWS Private CA Connector for Active Directory e aggiorna l'elenco delle directory. La nuova directory dovrebbe essere disponibile per la selezione.

> **Note**
>
> Quando crei una directory, tieni presente che Connector for AD supporta solo i seguenti tipi di directory disponibili nella AWS Directory Service console:
>
> - Microsoft AD gestito da AWS
> - AD Connector

- In Seleziona gruppi di sicurezza per endpoint VPC, scegli un gruppo di sicurezza dall'elenco.

  In alternativa, puoi scegliere Crea gruppo di sicurezza, che apre la console Amazon EC2 alla pagina Crea gruppo di sicurezza in una nuova finestra. Al termine della creazione di un gruppo di sicurezza, torna alla console AWS Private CA Connector for Active Directory e aggiorna l'elenco dei gruppi di sicurezza. Il nuovo gruppo di sicurezza dovrebbe essere disponibile per la selezione.

Scegli una CA privata

Nella sezione Autorità di certificazione privata, scegli una CA privata dall'elenco.

In alternativa, puoi scegliere Crea CA privata, che apre la CA privata AWS console alla pagina Autorità di certificazione private in una nuova finestra. Al termine della creazione di una CA, torna alla console AWS Private CA Connector for Active Directory e aggiorna l'elenco delle CA. La nuova CA dovrebbe essere disponibile per la selezione.

Configura l'etichettatura

Nel riquadro Tag, opzionale, puoi applicare e rimuovere i metadati sulla tua risorsa AD. I tag sono coppie di stringhe chiave-valore in cui la chiave deve essere unica per la risorsa e il valore è facoltativo. Il riquadro mostra tutti i tag esistenti per la risorsa in una tabella. Sono supportate le operazioni seguenti.

- Scegli Gestisci tag per aprire la pagina Gestisci tag.
- Scegli Aggiungi nuovo tag per creare un tag. Compila il campo Chiave e, facoltativamente, il campo Valore. Scegli Salva modifiche per applicare il tag.
- Scegli il pulsante Rimuovi accanto a un tag per contrassegnarlo per l'eliminazione e scegli Salva modifiche per confermare.

Rivedi e crea

Dopo aver fornito le informazioni richieste e aver esaminato le tue scelte, scegli Crea connettore. Viene visualizzata la pagina dei dettagli dei connettori per Active Directory in cui è possibile visualizzare lo stato di avanzamento del connettore durante la creazione.

Una volta completato il processo di creazione di un connettore, assegnagli un nome principale di servizio.

Crea un connettore per Active Directory () AWS CLI

Per creare un connettore per Active Directory con la CLI, utilizzare il comando create-connector nella sezione AWS Private CA Connettore per Active Directory del. AWS CLI

Crea un connettore per Active Directory (API)

Per creare un connettore per Active Directory con l'API, utilizza l' CreateConnectorazione nell'API AWS Private CA Connector for Active Directory.

Creazione di un modello di connettore

Creazione di un modello di connettore (console)

Completare le seguenti procedure per creare e configurare un modello di connettore utilizzandoAWSconsole.

Processi

- Console aperta
- Scegli il connettore
- Trova la sezione dei modelli
- Metodo di creazione del modello
- Impostazioni del modello
- Configura le impostazioni dei certificati
- Configura le impostazioni di gestione delle richieste e di registrazione
- Configura le estensioni di utilizzo delle chiavi
- Assegna politiche applicative
- Configura politiche applicative personalizzate

- Configurare le impostazioni di crittografia
- Configura gruppi e autorizzazioni
- Configura i modelli sostitutivi
- Configura l'etichettatura
- Rivedi e crea

Console aperta

Accedi al tuoAWSaccount e apri ilAWS Private CAConnettore per la console Active Directory all'indirizzo**https://console.aws.amazon.com/pca-connector-ad/home**.

Scegli il connettore

Scegliete un connettore traConnettori per Active Directoryelenca e poi scegliVisualizza i dettagli.

Trova la sezione dei modelli

Nella pagina dei dettagli del connettore, trovaModellisezione e poi scegliCrea modello.

Metodo di creazione del modello

SulCrea modellopagina, nellametodo di creazione del modellosezione, scegli una delle opzioni del metodo.

- Inizia da un modello predefinito(impostazione predefinita): scegli da un elenco di modelli predefiniti per le applicazioni AD:
  - Firma del codice
  - Computer
  - Autenticazione del controller di dominio
  - Agente di ripristino EFS
  - Agente di registrazione
  - Agente di registrazione (computer)
  - IPSec
  - Autenticazione Kerberos
  - Server RAS e IAS
  - Accesso con smartcard

- Firma dell'elenco di fiducia
- Firma dell'utente
- Autenticazione della workstation

- Inizia da un modello esistente che hai creato— Scegli da un elenco di modelli personalizzati che hai creato in precedenza.
- Inizia da un modello vuoto— Scegli questa opzione per iniziare a creare un modello completamente nuovo.

Impostazioni del modello

NelImpostazioni del modellosezione, fornisci le seguenti informazioni:

- nome del modello— Il nome del modello
- Versione dello schema del modello— La versione dello schema del modello. La versione dello schema influisce sulla disponibilità delle opzioni del modello nel modo seguente:

  Schema versione 2

  - Supporta la compatibilità client di Windows XP/Windows Server 2003 e versioni successive.
  - Supporta solo i provider di servizi crittografici precedenti.

  Schema versione 3

  - Supporta la compatibilità client di Windows Vista/Windows Server 2008 e versioni successive.
  - Supporta l'autorizzazione del richiedente al rinnovo con la chiave esistente.
  - Supporta solo i provider di archiviazione delle chiavi.

  Schema versione 4

  - Supporta la compatibilità client di Windows 8/Windows Server 2012 e versioni successive.
  - Supporta l'autorizzazione del richiedente al rinnovo con la chiave esistente.
  - Supporta i provider di servizi crittografici e i provider di archiviazione delle chiavi legacy.
- Compatibilità con i client— Il livello minimo del sistema operativo compatibile con il modello. Scegliete una delle opzioni elencate:
  - Windows XP/Windows Server 2003
  - Windows Vista/Windows Server 2008
  - Windows 7/Windows Server 2008 R2

- Windows 8 e versioni successive/Windows Server 2012
- Windows 8 e versioni successive/Windows Server 2012 R2
- Windows 8 e versioni successive/Windows Server 2016 e versioni successive

Configura le impostazioni dei certificati

NelImpostazioni del certificatosezione, definisci le seguenti impostazioni per i certificati basati su questo modello.

- Tipo di certificato— Specificare se creareUtenteoComputercertificati.
- Iscrizione automatica— Scegli se attivare la registrazione automatica per i certificati basati su questo modello.
- Periodo di validità— Specificare il periodo di validità del certificato come valore intero di ore, giorni, settimane, mesi o anni. Il valore minimo è 2 ore.
- periodo di rinnovo— Specificare un periodo di rinnovo del certificato come valore intero di ore, giorni, settimane, mesi o anni. Il periodo di rinnovo non deve superare il 75% del periodo di validità.
- Nome del soggetto— Scegli una o più opzioni da includere nel nome dell'oggetto in base alle informazioni contenute in Active Directory.

> **Note**
>
> È necessario specificare almeno un nome del soggetto o un'opzione di nome alternativo del soggetto.

  - Nome comune
  - DNS come nome comune
  - Percorso della directory
  - E-mail
- Nome alternativo del soggetto— Scegli una o più opzioni da includere nel nome alternativo dell'oggetto in base alle informazioni contenute in Active Directory.

> **Note**
>
> È necessario specificare almeno un nome del soggetto o un'opzione di nome alternativo del soggetto.

- GUID della directory
- Nome DNS
- DNS di dominio
- E-mail
- Nome principale del servizio (SPN)
- Nome principale dell'utente (UPN)

Configura le impostazioni di gestione delle richieste e di registrazione

NelOpzioni di gestione e registrazione delle richieste di certificatisezione, specifica lo scopo dei certificati in base al modello, scegliendo una delle seguenti opzioni.

- Signature (Firma)
- Encryption (Crittografia)
- Firma e crittografia
- Firma e accesso con smartcard

Quindi, scegli quale delle seguenti funzionalità attivare. Le opzioni variano a seconda dello scopo del certificato.

- Eliminare i certificati non validi (non archiviarli)
- Includi algoritmi simmetrici
- Chiave privata esportabile

Infine, scegli un'opzione di registrazione del certificato. Le opzioni variano a seconda dello scopo del certificato.

- Non è richiesto alcun input da parte dell'utente

- Chiedi conferma all'utente durante la registrazione
- Richiedi conferma all'utente durante la registrazione e richiedi l'input dell'utente

Configura le estensioni di utilizzo delle chiavi

NelImpostazioni delle estensioni di utilizzo chiavesezione, scegli l'opzione per l'utilizzo della firma e dell'utilizzo delle chiavi di crittografia.

Utilizzo della chiave di firma

- Firma digitale
- La firma è una prova dell'origine (non ripudio)

Utilizzo della chiave di crittografia

- Consenti lo scambio di chiavi senza crittografia delle chiavi (accordo chiave)
- Consenti lo scambio di chiavi solo con la crittografia delle chiavi (cifratura delle chiavi)
- Consenti la crittografia dei dati dell'utente (cifratura dei dati)

Puoi anche scegliere diRendi fondamentali le estensioni di utilizzo chiaveper entrambi i tipi di chiave.

Assegna politiche applicative

NelPolitiche applicativesezione, scegli tutte le politiche applicative applicabili. Le politiche disponibili sono elencate in diverse pagine. Alcune politiche potrebbero essere preselezionate a causa delle impostazioni precedenti.

Configura politiche applicative personalizzate

NelPolitiche applicative personalizzatesezione, è possibile aggiungere OID personalizzati al modello e specificare se le estensioni delle policy applicative sono fondamentali.

Configurare le impostazioni di crittografia

NelImpostazioni di crittografiasezione, scegli le seguenti categorie di impostazioni di crittografia per i certificati basati su questo modello.

1. Il contenuto nella parte superiore della sezione è determinato daMetodo di creazione del modello eImpostazioni del modello che hai scelto in precedenza.

- Se hai accettato l'impostazione predefinitaVersione del modello 2nelImpostazioni del modello , quindi qui vengono visualizzati i seguenti messaggi di stato:
  - categoria di provider di crittografia
  - Provider di servizi crittografici legacy

  In questo caso non ci sono impostazioni da configurare e puoi passare alla fase successiva.
- Se hai specificatoVersione del modello 3nelImpostazioni del modello , quindi qui vengono visualizzati i seguenti messaggi di stato:
  - categoria di provider di crittografia
  - Provider di archiviazione delle chiavi

  È inoltre necessario scegliere unAlgoritmo chiavedalle opzioni elencateECDH_P256,ECDH_P384,ECDH_P521, eRSA.
- Se hai specificatoVersione del modello 4nelImpostazioni del modello , allora devi scegliere tra unProvider di archiviazione delle chiavie unFornitore di servizi crittografici legacy. Se scegliL'archiviazione delle chiavi fornisce, quindi unAlgoritmo chiavedeve inoltre essere scelto tra le opzioni elencateECDH_P256,ECDH_P384,ECDH_P521, eRSA.

2. Dimensione minima della chiave (bit)— Specificare la dimensione minima della chiave. Questa impostazione influirà sui provider di crittografia disponibili.
3. Scegli quali provider crittografici possono essere utilizzati per le richieste— Scegli una delle due opzioni disponibili:

   - Le richieste possono utilizzare qualsiasi provider disponibile sul computer del soggetto
   - Le richieste devono utilizzare uno dei seguenti provider selezionati

     La scelta di questa opzione apre unProvider di crittografiaelenco. È possibile selezionare e assegnare priorità ai provider utilizzando i pulsanti presenti nelOrdinecolonna. Sono supportati i seguenti provider:
     - Microsoft Base Cryptographic Provider v1.0
     - Provider crittografico Microsoft Base DSS e Diffie-Hellman
     - Fornitore di servizi crittografici Microsoft Base Smart Card
     - Provider crittografico Microsoft DH SChannel
     - Microsoft Enhanced Cryptographic Provider v1.0
     - Provider crittografico Microsoft Enhanced DSS e Diffie-Hellman

- Provider crittografico Microsoft Enhanced RSA e AES
- Provider crittografico Microsoft RSA SChannel

Configura gruppi e autorizzazioni

NelGruppi e autorizzazioninella sezione, puoi visualizzare i modelli, i gruppi e le autorizzazioni esistenti per l'iscrizione, oppure puoi scegliere ilAggiungi nuovi gruppi e autorizzazionipulsante per aggiungerne di nuovi. Il pulsante apre un modulo che richiede le seguenti informazioni:

- Display name (Nome visualizzato)
- Identificatore di sicurezza(SID)
- Iscriversi, con le opzioni ALLOW | NEY | NOT SET
- Registrazione automatica, con le opzioni ALLOW | NEY | NOT SET

Configura i modelli sostitutivi

NelSostituisci i modellisezione, è possibile notificare ad Active Directory che il modello corrente sostituisce uno o più modelli creati in AD. Applica il modello sostitutivo scegliendoAggiungi modello da Active Directory da sostituiree specificando il nome comune del modello sostitutivo.

Configura l'etichettatura

NelTag: opzionaleriquadro, puoi applicare e rimuovere i metadati sulla tua risorsa AD. I tag sono coppie di stringhe chiave-valore in cui la chiave deve essere unica per la risorsa e il valore è facoltativo. Il riquadro mostra tutti i tag esistenti per la risorsa in una tabella. Sono supportate le operazioni seguenti.

- ScegliGestisci i tagper aprire ilGestire i tagpagina.
- Scegli Aggiungi nuovo tag per creare un tag. Compila ilChiavecampo e, facoltativamente, ilValorecampo. ScegliSalva le modificheper applicare il tag.
- Scegli ilRimuovipulsante accanto a un tag per contrassegnarlo per l'eliminazione e scegliSalva le modificheper confermare.

Rivedi e crea

Dopo aver fornito le informazioni richieste e aver esaminato le tue scelte, scegliCrea modello. Questo si apreDettagli del modello, dove puoi rivedere le impostazioni del nuovo modello, modificare o

eliminare il modello, gestire gruppi e autorizzazioni, gestire i modelli sostituiti, gestire i tag e impostare la nuova registrazione automatica per i titolari di certificati.

Creazione di un modello di connettore (CLI)

Usa ilcrea modellocomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Creazione di un modello di connettore (API)

Usa il CreateTemplateazione inAWS Private CAConnettore per l'API di Active Directory.

Elenco dei connettori per Active Directory

È possibile utilizzare ilAWS Private CAConnettore per la console Active Directory oAWS CLIper elencare i connettori di cui sei proprietario.

Per elencare i connettori utilizzando la console

1. Accedi al tuoAWSaccount e apri ilAWS Private CAConnettore per la console Active Directory all'indirizzo**`https://console.aws.amazon.com/pca-connector-ad/home`**.
2. Rivedi le informazioni contenute nelConnettori per Active Directoryelenco. È possibile navigare tra più pagine di connettori utilizzando i numeri di pagina in alto a destra. Per impostazione predefinita, ogni connettore occupa una riga che mostra le seguenti colonne di informazioni.

- ID del connettore— L'ID univoco del connettore.
- Nome della directory— La risorsa Active Directory associata al connettore.
- Stato del connettore— Stato del connettore. I valori possibili sono:Creare|Attivo|Eliminazione|Fallito.
- Stato del nome principale del servizio— Stato del nome principale del servizio (SPN) associato al connettore. I valori possibili sono:Creare|Attivo|Eliminazione|Fallito.
- Stato di registrazione all'elenco— Stato di registrazione del direttore associato. I valori possibili sono:Creare|Attivo|Eliminazione|Fallito.
- Creato in— Timestamp al momento della creazione del connettore.

Scegliendo l'icona a forma di ingranaggio nell'angolo in alto a destra della console, puoi personalizzare il numero di connettori mostrati su una pagina utilizzandoDimensioni della paginapreferenza.

Per elencare i connettori, utilizzare ilAWS CLI

Usa illist-connectorcomando per elencare i connettori.

Per elencare i connettori utilizzando l'API

Usa ilListConnectorsazione inAWS Private CAConnettore per l'API di Active Directory.

Elenco dei modelli di connettori

È possibile utilizzare ilAWS Private CAConnettore per la console Active Directory oAWS CLIper elencare i modelli per i connettori di cui sei proprietario. I modelli di connettore si basano suAWS Private CA BlankEndEntityCertificate_API Passthrough/v1modelli.

Per elencare i modelli utilizzando la console

1. Accedi al tuoAWSaccount e apri ilAWS Private CAConnettore per la console Active Directory all'indirizzo`https://console.aws.amazon.com/pca-connector-ad/home`.
2. Scegli un connettore traConnettori per Active Directoryelenca e poi scegliVisualizza i dettagli.
3. Nella pagina dei dettagli del connettore, esamina le informazioni contenute nelModellisezione. Puoi navigare tra più pagine di modelli utilizzando i numeri di pagina in alto a destra. Ogni modello occupa una riga che mostra le seguenti colonne di informazioni.

- Nome del modello— Il nome leggibile dall'uomo del modello.
- Stato del modello— Stato del modello. I valori possibili sono:Attivo|Eliminazione.
- ID del modello— L'identificatore univoco del modello.

Per elencare i modelli utilizzando ilAWS CLI

Usa ilmodelli di elencocomando per elencare i modelli per il connettore specificato.

Per elencare i modelli utilizzando l'API

Usa il ListTemplatesazione inAWS Private CAConnettore per l'API Active Directory per elencare i modelli per il connettore specificato.

Visualizza i dettagli del connettore

Utilizza le seguenti procedure per visualizzare i dettagli di configurazione di un connettore nella console, nella riga di comando o nell'API perAWS Private CAConnettore per Active Directory.

Visualizza connettore (console)

Per visualizzare i dettagli di un connettore (console)

1. Accedi al tuoAWSaccount e apri ilAWS Private CAConnettore per la console Active Directory all'indirizzo`https://console.aws.amazon.com/pca-connector-ad/home`.
2. Scegli un connettore traConnettori per Active Directoryelenca e poi scegliVisualizza i dettagli.
3. Nella pagina dei dettagli del connettore, esamina le informazioni nel riquadro Dettagli del connettore, che include quanto segue:
   - ID del connettore
   - Stato del connettore
   - Dettagli aggiuntivi sullo stato
   - Connettore ARN
   - Endpoint del server per la politica di registrazione dei certificati
   - Nome della directory
   - ID della directory
   - CA privata AWSsoggetto
   - CA privata AWSstato
   - endpoint e gruppi di sicurezza VPC
4. NelModelliriquadro, è possibile creare o gestire i modelli associati al connettore.
5. DalNome principale del servizio (SPN)riquadro, è possibile visualizzare il nome del principio di servizio associato al connettore.
6. DalRegistrazione alla directoryriquadro, è possibile visualizzare o modificare la registrazione della directory associata al connettore.
7. DalTag —opzionaleriquadro, è possibile creare o gestire i tag associati al connettore.

Visualizza connettore (CLI)

Usa ilget-connectorcomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Visualizza connettore (API)

Usa ilGetConnectorazione inAWS Private CAConnettore per l'API di Active Directory.

Visualizza i dettagli del modello di connettore

Utilizza le seguenti procedure per visualizzare i dettagli di configurazione di un modello di connettore utilizzando la console, la riga di comando o l'API perAWS Private CAConnettore per Active Directory

Visualizza modello (console)

Per visualizzare i dettagli di un modello di connettore (console)

1. Accedi al tuoAWSaccount e apri ilAWS Private CAConnettore per la console Active Directory all'indirizzo**`https://console.aws.amazon.com/pca-connector-ad/home`**.
2. Scegli un connettore traConnettori per Active Directoryelenca e poi scegliVisualizza i dettagli.
3. Nella pagina dei dettagli del connettore, esamina le informazioni contenute nelModellisezione e seleziona il modello che desideri ispezionare. Quindi scegliVisualizza i dettagli.
4. Nella pagina dei dettagli,Dettagli del modelloIl riquadro mostra le seguenti informazioni sul modello:

   - Nome del modello
   - ID del modello
   - Stato del modello
   - Versione dello schema del modello
   - Versione del modello
   - Modello ARN
   - Tipo di certificato
   - Registrazione automatica attivata
   - Periodo di validità
   - periodo di rinnovo

- Requisiti del nome del soggetto
- Requisiti relativi al nome alternativo dell'oggetto
- Impostazioni di richiesta e registrazione del certificato
- Categoria di provider di crittografia
- Algoritmo chiave
- Dimensione minima della chiave (bit)
- Algoritmo hash
- Fornitori di crittografia
- Impostazioni dell'estensione per l'utilizzo delle chiavi

Da questo riquadro è inoltre possibile eseguire le seguenti azioni utilizzandoModifica,Eliminare, eAzionibottoni.

- Modificare
- Elimina
- Gestisci gruppi e autorizzazioni— Per ulteriori informazioni, vedereConfigura gruppi e autorizzazioni.
- Gestisci i modelli sostituiti— Per ulteriori informazioni, vedereRivedi e crea.
- Gestisci i tag— Per ulteriori informazioni, vedereConnettore di etichettatura per risorse AD.
- Registra nuovamente tutti i titolari di certificati— Questa impostazione consente di aumentare automaticamente la versione principale di un modello. Tutti i membri dei gruppi di Active Directory autorizzati a registrarsi con un modello riceveranno un nuovo certificato rilasciato utilizzando tale modello. Per ulteriori informazioni, consulta l'API UpdateTemplate.

5. Il riquadro inferiore mostra una riga di schede che consentono di modificare la configurazione del modello.

- Gruppi e autorizzazioni— Visualizza e gestisci le autorizzazioni per i gruppi di Active Directory per registrare i certificati utilizzando questo modello. Per ulteriori informazioni, vedereConfigurare gruppi e autorizzazioni
- Politiche delle applicazioni— Visualizza e gestisci le politiche applicative dei modelli. Per ulteriori informazioni, vedereAssegnare politiche applicative.
- Modelli sostituiti— Visualizza e gestisci i modelli sostituiti. Per ulteriori informazioni, vedereRivedi e crea.

- Etichettaopzionale— Visualizza e gestisci i tag su questo modello. Per ulteriori informazioni, consulta Connettore di etichettatura per risorse AD.

Per visualizzare i dettagli di un modello di connettore (AWS CLI)

Visualizza modello (CLI)

Usa ilget-templatecomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Visualizza modello (API)

Per visualizzare i dettagli di un modello di connettore (API)

Usa il GetTemplateazione inAWS Private CAConnettore per l'API di Active Directory.

Gestione delle registrazioni degli elenchi

Per gestire le registrazioni nelle directory (console)

Le registrazioni delle directory per i connettori possono essere gestite dal livello superiore diAWS Private CAConnettore per la console Active Directory. Questo argomento illustra le opzioni di gestione disponibili.

1. Accedi al tuoAWSaccount e apri ilAWS Private CAConnettore per la console Active Directory all'indirizzo**`https://console.aws.amazon.com/pca-connector-ad/home`**.
2. Nell'area di navigazione a sinistra, scegliRegistrazioni agli elenchi.
3. LaRegistrazioni agli elenchila pagina mostra una tabella delle directory registrate con i seguenti campi:

   - ID della directory— L'ID univoco della directory
   - Nome della directory— Il nome del sito del dominio della directory
   - Tipo di directory
   - Registrato— Lo stato della registrazione. I valori supportati sono CREATING | ACTIVE | DELETING | FAILED.
   - Stato della directory— Lo stato della directory

   L'utente può utilizzareElenco dei registriper creare una nuova registrazione.

4. È possibile selezionare una delle registrazioni elencate per gestirla. Questo abilita ilVisualizza i dettagli della registrazioneeAnnullare la registrazione dell'elencopulsanti. LaVisualizza i dettagli della registrazioneil pulsante apre la pagina dei dettagli per la registrazione.
5. LaDettagli di registrazione all'elencoIl riquadro mostra le seguenti informazioni:

   - Nome del sito di dominio della directory
   - ID della directory— L'ID univoco della directory. Scegliendo il link si accede alAWS Directory Serviceconsole.
   - Tipo di directory
   - Stato— Stato della directory
   - ARN di registrazione all'elenco— Il nome della risorsa Amazon della registrazione della directory
   - Informazioni aggiuntive sullo stato
6. NelConnettori e nome principale del servizio (SPN)riquadro, è possibile gestire gli SPN per il connettore. Per ulteriori informazioni, vedereVisualizza i dettagli del connettore.
7. NelTag: opzionaleriquadro, puoi applicare e rimuovere i metadati sulla tua risorsa AD. I tag sono coppie di stringhe chiave-valore in cui la chiave deve essere unica per la risorsa e il valore è facoltativo. Il riquadro mostra tutti i tag esistenti per la risorsa in una tabella. Sono supportate le operazioni seguenti.

   - ScegliGestisci i tagper aprire ilGestire i tagpagina.
   - Scegli Aggiungi nuovo tag per creare un tag. Compila ilChiavecampo e, facoltativamente, ilValorecampo. ScegliSalva le modificheper applicare il tag.
   - Scegli ilRimuovipulsante accanto a un tag per contrassegnarlo per l'eliminazione e scegliSalva le modificheper confermare.

Per gestire le registrazioni delle directory (CLI)

Creare: Usa ilcreate-directory-registrationcomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Recuperare:get-directory-registrationcomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Elenco:list-directory-registrationscomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Eliminare:delete-directory-registrationcomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Per gestire le registrazioni nelle directory (API)

Creare:CreateDirectoryRegistrationazione nelAWS Private CAConnettore per l'API di Active Directory.

Recupera:GetDirectoryRegistrationazione nelAWS Private CAConnettore per l'API di Active Directory.

Elenco:ListDirectoryRegistrationsazione nelAWS Private CAConnettore per l'API di Active Directory.

Elimina:DeleteDirectoryRegistrationazione nelAWS Private CAConnettore per l'API di Active Directory.

Gestione dei gruppi AD e delle autorizzazioni per i modelli

Per gestire i gruppi di modelli e le autorizzazioni (console)

I gruppi e le autorizzazioni per un modello esistente possono essere gestiti dalla pagina dei dettagli del modello. Per ulteriori informazioni, consulta Visualizza i dettagli del modello di connettore.

Imposta le autorizzazioni in base alle quali i gruppi possono o non possono registrare certificati per il modello specifico. L'identificatore di sicurezza (SID) del gruppo viene fornito dall'utente. Quindi imposti le autorizzazioni di registrazione e registrazione automatica per il gruppo. Per l'iscrizione automatica, sia l'iscrizione che la registrazione automatica devono essere impostate su «Consenti».

Cerca l'identificatore di sicurezza del gruppo in Active Directory

È possibile utilizzare lo script seguente per cercare l'identificatore di sicurezza del gruppo in Active Directory.

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

Per gestire i gruppi di modelli e le autorizzazioni (CLI)

Crea il comando: create-template-group-access-control-entry nella sezione AWS Private CA Connector for Active Directory di. AWS CLI

Aggiornamento: comando update-template-group-access-control-entry nella sezione AWS Private CA Connector for Active Directory di. AWS CLI

Recupera il comando: get-template-group-access-control-entry nella sezione AWS Private CA Connector for Active Directory di. AWS CLI

Elenca: comando list-template-group-access-control-entries nella sezione AWS Private CA Connector for Active Directory di. AWS CLI

Elimina: comando delete-template-group-access-control-entries nella sezione AWS Private CA Connector for Active Directory di. AWS CLI

Per gestire i gruppi di modelli e le autorizzazioni (API)

Crea: CreateTemplateGroupAccessControlEntryazione nell'API AWS Private CA Connector for Active Directory.

Aggiornamento: UpdateTemplateGroupAccessControlEntryazione nell'API AWS Private CA Connector for Active Directory.

Recupera: GetTemplateGroupAccessControlEntryazione nell'API AWS Private CA Connector for Active Directory.

Elenco: ListTemplateGroupAccessControlEntriesazione nell'API AWS Private CA Connector for Active Directory.

Elimina: DeleteTemplateGroupAccessControlEntryazione nell'API AWS Private CA Connector for Active Directory.

Configurazione del nome principale del servizio

Per gestire i nomi principali dei servizi (console)

Il nome principale del servizio (SPN) di un connettore AD esistente può essere gestito dalla pagina dei dettagli del connettore. Per ulteriori informazioni, vedere Gestione della registrazione delle directoryVisualizza i dettagli del connettore

Per gestire i nomi principali dei servizi (CLI)

Creare: create-service-principal-namecomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Recuperare: get-service-principal-namecomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Elenco: list-service-principal-namescomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Eliminare: delete-service-principal-namecomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Per gestire i nomi principali dei servizi (API)

Creare: CreateServicePrincipalNameazione nelAWS Private CAConnettore per l'API di Active Directory.

Recupera: GetServicePrincipalNameazione nelAWS Private CAConnettore per l'API di Active Directory.

Elenco: ListServicePrincipalNamesazione nelAWS Private CAConnettore per l'API di Active Directory.

Elimina: DeleteServicePrincipalNameazione nelAWS Private CAConnettore per l'API di Active Directory.

Connettore di etichettatura per risorse AD

Puoi applicare tag ai connettori, ai modelli e alle registrazioni delle directory. L'etichettatura aggiunge metadati a una risorsa che può aiutare nell'organizzazione e nella gestione.

Per gestire l'etichettatura delle risorse (console)

L'etichettatura delle risorse esistenti viene gestita nella pagina dei dettagli della risorsa. Per ulteriori informazioni, consulta le procedure seguenti:

- Visualizza i dettagli del modello di connettore
- Gestione delle registrazioni degli elenchi

Per gestire la codifica delle risorse (CLI)

Etichetta:tag-risorsacomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Elenca i tag: list-tags-for-resourcecomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Untag:risorsa untagcomando inAWS Private CAConnettore per la sezione Active Directory delAWS CLI.

Per gestire l'etichettatura delle risorse (API)

Etichetta: TagResourceazione nelAWS Private CAConnettore per l'API di Active Directory.

Elenca i tag: ListTagsForResourceazione nelAWS Private CAConnettore per l'API di Active Directory.

Annulla tag: UntagResourceazione nelAWS Private CAConnettore per l'API di Active Directory.

Importante: è accettabile utilizzare tag per etichettare oggetti contenenti dati riservati. Tuttavia, i tag stessi non devono contenere informazioni di identificazione personale (PII), informazioni sensibili o riservate.

Risoluzione dei problemi

Consulta i seguenti argomenti in caso di problemi relativi all'utilizzo di CA privata AWS.

Argomenti

Creazione e firma di un certificato CA privato

Dopo aver creato la CA privata, è necessario recuperare la CSR e inviarla a una CA intermedia o root nell'infrastruttura X.509. La CA usa la CSR per creare il certificato CA privato e quindi firma il certificato prima di restituirlo all'utente.

Purtroppo, non è possibile fornire consigli specifici ai problemi relativi alla creazione e alla firme del certificato CA privato. I dettagli dell'infrastruttura X.509 e della gerarchia CA al suo interno non rientrano nell'ambito di questa documentazione. CA privata AWS

Latenza nelle risposte OCSP

La reattività OCSP può essere più lenta se il chiamante è geograficamente distante da una cache periferica regionale o dalla regione della CA emittente. Per ulteriori informazioni sulla disponibilità della cache edge regionale, vedere Global Edge Network. Consigliamo di emettere certificati in una regione vicina a quella in cui verranno utilizzati.

Configurazione di Amazon S3 per consentire la creazione di un bucket CRL

La tua CA privata potrebbe non riuscire a creare un bucket CRL se sul tuo account viene applicato l'accesso pubblico ad Amazon S3 Block (impostazioni del bucket). Controlla le impostazioni di Amazon S3 se ciò si verifica. Per ulteriori informazioni, consulta Utilizzo del blocco dell'accesso pubblico di Amazon S3.

Revoca di un certificato CA autofirmato

Non è possibile revocare un certificato emesso da una CA autofirmato. Invece, è necessario eliminare la CA.

Gestione delle eccezioni

Un CA privata AWS comando potrebbe non riuscire per diversi motivi. Per informazioni su ciascuna eccezione e suggerimenti per risolverle, consulta la tabella seguente.

CA privata AWS Eccezioni

Eccezione restituita da CA privata AWS	Descrizione	Correzione
`AccessDeniedException`	Le autorizzazioni necessarie per utilizzare il comando specificato non sono state delegate da una CA privata all'account chiamante.	Per informazioni sulla delega delle autorizzazioni in CA privata AWS, vedere. Assegna le autorizzazioni per il rinnovo dei certificati ad ACM
`InvalidArgsException`	È stata effettuata una richiesta di creazione o rinnovo del certificato con parametri non validi.	Controlla la documentazione individuale del comando per avere la certezza che i parametri di input siano validi. Se si sta creando un nuovo certificato, assicurati che l'algoritmo di firma richiesto

Eccezione restituita da CA privata AWS	Descrizione	Correzione
		possa essere utilizzato con il tipo di chiave della CA.
`InvalidStateException`	La CA privata associata non può rinnovare il certificato perché non è nello stato `ACTIVE`.	Tentativo di ripristino della CA privata. Se la CA privata non rientra nel periodo di ripristino, la CA non può essere ripristinata e il certificato non può essere rinnovato.
`LimitExceededException`	Ogni autorità di certificazione (CA) dispone di una quota di certificati che può rilasciare. La CA privata associata al certificato designato ha raggiunto la quota. Per ulteriori informazioni, vedere Service Quotas nella Riferimenti generali di AWS Guida.	Contatta il AWS Support Centro per richiedere un aumento della quota.
`MalformedCSRException`	La richiesta di firma del certificato (CSR) inviata a CA privata AWS non può essere verificata o convalidata.	Verifica che la CSR sia stata generata e configurata correttamente.
`OtherException`	Un errore interno ha causato l'esito negativo della richiesta.	Prova a eseguire nuovamente il comando. Se il problema persiste, contatta il AWS Support Centro.
`RequestFailedException`	Un problema di rete nell' AWS ambiente in uso ha causato il fallimento della richiesta.	Riprova la richiesta . Se l'errore persiste, controlla la configurazione di Amazon VPC (VPC).

Eccezione restituita da CA privata AWS	Descrizione	Correzione
`ResourceNotFoundException`	La CA privata che ha emesso il certificato è stata eliminata e non esiste più.	Richiedi un nuovo certificato da un'altra CA attiva.
`ThrottlingException`	Operazione API richiesta non riuscita perché ha superato una quota.	Verifica che non stai emettendo più chiamate di quelle consentite da. CA privata AWS È possibile che si verifichi un errore `ThrottlingException` anche perché si è verificata una condizione transitoria e non per una quota superata. Se si verifica l'errore e non hai effettuato chiamate in eccesso rispetto alla quota, ritenta la richiesta. Se hai superato la quota, puoi richiederne un aumento. Per ulteriori informazioni, vedere Service Quotas nella Riferimenti generali di AWS Guida.
`ValidationException`	I parametri di input della richiesta sono stati formattati in modo errato o il periodo di validità del certificato root termina prima del periodo di validità del certificato richiesto.	Controlla i requisiti di sintassi dei parametri di input del comando e il periodo di validità del certificato root della CA. Per informazioni sulla modifica del periodo di validità, consulta Aggiornamento della CA privata.

Utilizzo dello standard Matter

Lo standard di connettività Matter specifica le configurazioni dei certificati che migliorano la sicurezza e la coerenza dei dispositivi Internet of Things (IoT). Gli esempi Java per la creazione di certificati CA root, CA intermedi e certificati di entità finale conformi a Matter sono disponibili all'indirizzo. Utilizzo dell'CA privata AWSAPI per implementare lo standard Matter (esempi Java)

Per facilitare la risoluzione dei problemi, gli sviluppatori di Matter forniscono uno strumento di verifica dei certificati chiamato chip-cert. Gli errori segnalati dallo strumento sono elencati nella tabella seguente con le relative correzioni.

Codice di errore	Significato	Correzione
0x000003 5	`BasicConstraints` e le `KeyUsage` `Extension` `KeyUsage` estensioni devono essere contrasse gnate come critiche.	Assicurati di aver selezionato il modello corretto per il tuo caso d
0x000000 0	L'estensione dell'identificator e della chiave di autorità deve essere presente.	CA privata AWS non imposta l'estensione dell'identificatore dell chiave di autorità sui certificati root. È necessario generare un A KeyIdentifier valore con codifica Base64 utilizzando il CSR e qu passarlo tramite un. CustomExtension Per ulteriori informazioni, Attiva una CA principale per i certificati operativi dei nodi (NOC) un'autorità di attestazione del prodotto (PAA).
0x000000 E	Il certificato è scaduto.	Assicurati che il certificato che utilizzi non sia scaduto.
0x000000 4	Errore di convalida della catena di certificati.	Questo errore può verificarsi se si tenta di creare un certificato d finale conforme a Matter senza utilizzare gli esempi Java forniti, utilizzano l'API per passare un certificato correttamente configu privata AWS KeyUsage Per impostazione predefinita, CA privata AWS genera valori di KeyUsage estensione a nove bit, con il nono bit che genera un aggiuntivo. Matter ignora il byte aggiuntivo durante le conversio

Codice di errore	Significato	Correzione
		di formato, causando errori di convalida della catena. Tuttavia, u CustomExtensionnel `APIPassthrough` modello può essere u per impostare il numero esatto di byte nel valore. `KeyUsage` Pe un esempio, consulta Creare un certificato operativo del nodo (N Se si modifica il codice di esempio o si utilizza un'utilità X.509 a come OpenSSL, è necessario eseguire una verifica manuale pe errori di convalida della catena. Per verificare che le conversioni siano senza perdite 1. Usa openssl per verificare che un certificato (certificato di n (entità finale) contenga una catena valida. In questo esemp `rcac.pem` è il certificato CA principale, è il certificato CA in `icac.pem` ed è il certificato del nodo. `noc.pem` ``` openssl verify -verbose -CAfile <(cat rcac.pem icac noc.pem ``` 2. Usa chip-cert per convertire il certificato del nodo in formato formato TLV (tag, length, value) e viceversa. ``` ./chip-cert convert-cert noc.pem noc.chip -c ./chip-cert convert-cert noc.chip noc_converted.pem ``` I file `noc_converted.pem` dovrebbero essere esattamen `noc.pem` gli stessi confermati da uno strumento di confron stringhe.

Connettore per errori e guasti AD

Utilizzate queste informazioni per diagnosticare e correggere gli errori e gli errori di creazione quando utilizzate Connector for AD.

Argomenti

Errori

Connector for AD invia messaggi di errore per diversi motivi. Per informazioni su ogni errore e consigli su come risolverli, consulta la tabella seguente. Puoi ricevere questi errori iscrivendoti agli eventi di Amazon EventBridge Scheduler (fonte dell'evento:`aws.pca-connector-ad`) o utilizzando la registrazione manuale in Windows.

Codice di errore	Significato	Correzione
0x8FFFA000	Autenticazione Kerberos non riuscita.	Se utilizzi la registrazione automatica, correggi il responsabile del servizio di AWS risorse. Se utilizzi l'interfaccia utente di Active Directory per ottenere un certificato, esegui. `gpupdate /force`
0x8FFFA001	Il messaggio SOAP deve contenere un'intestazione di azione.	Aggiungi un'intestazione di azione.
0x8FFFA002	Il connettore non ha accesso alla CA privata a cui è connesso.	Condividi la tua CA privata con il connettore creando un AWS Resource Access Manager (RAM) da condividere tra la tua CA privata e il servizio Connector for AD.

Codice di errore	Significato	Correzione
0x8FFFA003	La CA privata per questo connettore non è attiva.	Sposta la CA privata allo stato Attivo. Se lo stato del certificato della CA privata è in sospeso, installa il certificato CA.
0x8FFFA004	La CA privata per questo connettore non esiste.	Sposta l'autorità di certificazione allo stato Attivo se si trova nello stato Eliminato. Se la tua CA privata viene eliminata definitivamente, crea un nuovo connettore con una CA diversa.
0x8FFFA005	Il modello ha specificato l'`directoryGuid` attributo per l'oggetto del certificato o il nome alternativo del soggetto, ma l'attribu to non è stato trovato nell'oggetto AD per il richiedente.	Active Directory non ha generato un file `directoryGuid` per la tua directory. Risolvi i problemi in Active Directory.
0x8FFFA006	Il modello ha specificato l'`dnsHostName` attributo per l'oggetto del certificato o il nome alternativo del soggetto, ma l'attribu to non è stato trovato nell'oggetto AD per il richiedente.	Aggiungi l'`dnsHostName` attributo al tuo oggetto AD.
0x8FFFA007	Il modello specificava l'attributo email da includere nell'oggetto del certifica to o nel nome alternativo dell'ogge tto, ma l'attributo non è stato trovato nell'oggetto AD del richiedente.	Aggiungi l'attributo email al tuo oggetto AD

Codice di errore	Significato	Correzione
0x8FFFA008	Il messaggio SOAP deve avere un'intestazione di azione uguale o. `http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy/IPolicy/GetPolicies http://schemas.microsoft.com/windows/pki/2009/01/enrollment/RST/wstep`	Aggiorna l'intestazione dell'azione per utilizzare uno dei valori specificati.
0x8FFFA009	Deve essere codificato in. BinarySecurityToken `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary`	Aggiorna il tipo di token di sicurezza binario.
0x8FFFA00A	Non è valido. BinarySecurityToken	Verificate che la CSR sia generata correttamente.
0x8FFFA00B	BinarySecurityToken Deve avere un tipo di valore pari o. `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#PKCS7 http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10`	Aggiornare il tipo di valore del token di sicurezza binario su un valore valido.

Codice di errore	Significato	Correzione
0x8FFFA00C	Il CMS contenuto non valido BinarySecurityToken .	Il Base64 è valido ma la sintassi dei messaggi crittografici (CMS) non è valida. Esamina la sintassi CMS.
0x8FFFA00D	Conteneva un CSR non valido. BinarySecurityToken	Verifica che la CSR sia stata generata correttamente.
0x8FFFA00E	La CA privata non è stata in grado di emettere un certificato utilizzando il modello specifico.	Rivedi l'eccezione di convalida di AWS Private CA. Puoi visualizzare l'eccezione di convalida in Amazon EventBridge o AWS CloudTrail.
0x8FFFA00F	Il messaggio SOAP deve avere un tipo di richiesta di. `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue`	Imposta il tipo di richiesta su`http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue` .
0x8FFFA010	Il messaggio SOAP deve avere un'intestazione to del campo del connettore o del `CertificateEnrollmentPolicyServerEndpoint` campo URI nella risposta XCEP.	Imposta l'intestazione del token di sicurezza della richiesta `CertificateEnrollmentPolicyServerEndpoint` sul campo o sul campo URI nella risposta XCEP.
0x8FFFA011	Il messaggio SOAP deve avere una sola intestazione di azione.	Esamina l'intestazione del messaggio SOAP del token di sicurezza della richiesta e imposta l'intestazione correttamente.

Codice di errore	Significato	Correzione
0x8FFFA012	Il messaggio SOAP deve avere una sola intestazione. `messageId`	Esamina l'intestazione del messaggio SOAP del token di sicurezza della richiesta e imposta l'intestazione correttamente.
0x8FFFA013	Il messaggio SOAP deve avere una sola intestazione.	Esamina l'intestazione del messaggio SOAP del token di sicurezza della richiesta e imposta l'intestazione correttamente.
0x8FFFA014	Il richiedente non ha accesso al modello richiesto.	Consenti al gruppo del richiedente di registrarsi utilizzando il modello richiesto creando un Access Control Entry.
0x8FFFA015	La `CertificateTemplateInformation` o l'`CertificateTemplateName` estensione devono essere presenti in. BinarySecurityToken	Aggiungi l'estensione di sicurezza alla tua CSR.
0x8FFFA016	Il modello richiesto non è stato trovato per il connettore specificato.	I modelli sono risorse secondarie per ogni connettore. Crea il modello per il connettore utilizzando`createTemplate` .
0x8FFFA017	La richiesta è stata negata a causa del throttling della richiesta.	Rallenta la frequenza delle richieste.
0x8FFFA018	Il messaggio SOAP deve contenere un'intestazione. `to`	Controlla l'intestazione del messaggio SOAP.

Codice di errore	Significato	Correzione
0x8FFFA019	Impossibile elaborare il messaggio SOAP a causa di un'intestazione non riconosciuta.	Controlla l'intestazione del messaggio SOAP.
0x8FFFA01A	Il modello specificava l'attributo UPN da includere nell'oggetto del certificato o nel nome alternativo del soggetto, ma l'attributo non è stato trovato nell'oggetto AD del richieden te.	Aggiungi un UPN all'oggetto Active Directory.

Errori di creazione del connettore

La creazione dei connettori può fallire per vari motivi. Quando la creazione del connettore non riesce, riceverai il motivo dell'errore nella risposta dell'API. Se utilizzi la console, il motivo dell'errore viene visualizzato nella pagina dei dettagli del connettore nel campo Ulteriori dettagli sullo stato all'interno del contenitore dei dettagli del connettore. La tabella seguente descrive i motivi dell'errore e i passaggi consigliati per la risoluzione.

Stato dell'errore	Descrizione	Correzione
`CA_CERTIFICATE_REG ISTRATION_FAILED`	Connector for AD non è in grado di importare i certificati CA nella tua directory.	Consulta la pagina Prerequis iti e verifica che il tuo account di servizio disponga delle autorizzazioni corrette. Dopo aver delegato le autorizza zioni corrette al tuo account di servizio, elimina il connettore guasto e creane uno nuovo. Per informazioni sulla delega delle autorizzazioni, consulta Delegare i privilegi all'accou

Stato dell'errore	Descrizione	Correzione
		nt di servizio nella Guida all'amministrazione.AWS Directory Service
`DIRECTORY_ACCESS_DENIED`	Connector for AD non è in grado di accedere alla tua directory.	Devi concedere a Connector for AD l'accesso alla tua directory. Consulta la Politica IAM sezione per assicurarti che la policy IAM associata al tuo AWS account ti consenta di accedere e descrivere le directory. Dopo aver concesso le autorizzazioni corrette al tuo AWS ruolo, elimina il connettor e guasto e creane uno nuovo.
`INTERNAL_FAILURE`	Connector for AD ha riscontra to un errore interno.	Riprova più tardi. Eliminare il connettore guasto e crearne uno nuovo.

Stato dell'errore	Descrizione	Correzione
`PRIVATECA_ACCESS_DENIED`	Connector for AD non è in grado di accedere alla tua CA privata.	Consulta la pagina Prerequisiti e verifica di disporre delle autorizzazioni necessarie per creare un connettore. Per informazioni, consulta Politica IAM. Se stai creando un connettore tramite AWS CLI una nostra API, consulta la pagina Prerequisiti e verifica di aver condiviso la CA privata con Connector for AD utilizzando. AWS Resource Access Manager Dopo aver verificato e corretto le autorizzazioni IAM e la condivisione AWS RAM delle risorse, elimina il connettore guasto e creane uno nuovo.
`PRIVATECA_RESOURCE_NOT_FOUND`	Connector for AD non riesce a trovare la CA privata specificata.	Assicurati di specificare il CA Amazon Resource Name (ARN) privato corretto, quindi elimina il connettore guasto e creane uno nuovo utilizzando il CA ARN privato desiderato.

Stato dell'errore	Descrizione	Correzione
`SECURITY_GROUP_NOT_IN_VPC`	Il gruppo di sicurezza non si trova nel VPC che ospita la tua directory.	Usa un gruppo di sicurezza che si trova nel VPC che ospita la tua directory. Per ulteriori informazioni, consulta Gruppi di sicurezza. Elimina il connettore guasto e creane uno nuovo con un gruppo di sicurezza che si trova nel VPC.
`VPC_ACCESS_DENIED`	Connector for AD non può accedere all'Amazon VPC che ospita la tua directory.	Controllare le autorizzazioni IAM. Elimina il connettore guasto e creane uno nuovo. Per un esempio di policy IAM che include le autorizzazioni di accesso, vedi Politica IAM
`VPC_ENDPOINT_LIMIT_EXCEEDED`	Connector for AD non può creare un endpoint nel tuo Amazon VPC. Hai raggiunto il limite di endpoint VPC che puoi creare per il tuo account.	Elimina gli endpoint Amazon VPC o richiedi un aumento del limite. Dopo aver eseguito uno dei due passaggi, elimina il connettore guasto e creane uno nuovo. Per informazioni sulle quote, consulta le quote di Amazon Virtual Private Cloud Service.

Stato dell'errore	Descrizione	Correzione
`VPC_RESOURCE_NOT_FOUND`	Connector for AD non riesce a trovare il VPC specificato.	Assicurati di aver specificato il VPC corretto e che il VPC esista. Quindi elimina il connettore guasto e creane uno nuovo utilizzando l'ID VPC corretto.

Errori di creazione SPN

La creazione del nome principale del servizio (SPN) può non riuscire per vari motivi. Quando la creazione di SPN fallisce, riceverai il motivo dell'errore nella risposta dell'API. Se utilizzi la console, il motivo dell'errore viene visualizzato nella pagina dei dettagli del connettore nel campo Ulteriori dettagli sullo stato all'interno del contenitore Service principal name (SPN). La tabella seguente descrive i motivi dell'errore e i passaggi consigliati per la risoluzione.

Stato dell'errore	Descrizione	Correzione
`DIRECTORY_ACCESS_DENIED`	Connector for AD non può accedere alla tua directory.	Concedi a Connector for AD l'accesso alla tua directory. Per un esempio di policy IAM che include le autorizzazioni che garantiscono l'accesso alle directory, consultaPolitica IAM.
`DIRECTORY_NOT_REACHABLE`	Connector for AD non può accedere alla tua directory.	Controlla la rete tra AWS e la tua directory e prova a creare nuovamente un SPN.
`DIRECTORY_RESOURCE_NOT_FOUND`	Connector for AD non riesce a trovare la directory specificata.	Assicurati di specificare l'ID di directory corretto, quindi elimina il connettore guasto e

Stato dell'errore	Descrizione	Correzione
		creane uno nuovo utilizzando l'ID di directory desiderato.
`INTERNAL_FAILURE`	Connector for AD ha riscontrato un errore interno.	Riprova più tardi.
`SPN_EXISTS_ON_DIFFERENT_AD_OBJECT`	Il nome principale del servizio (SPN) esiste su un oggetto Active Directory diverso.	Eliminare l'SPN dall'oggetto Active Directory e provare a creare nuovamente l'SPN.
`SPN_LIMIT_EXCEEDED`	Connector for AD non può creare l'SPN perché hai raggiunto il limite di SPN per directory. Il numero massimo di SPN per directory è 10.	Elimina uno o più SPN dal tuo account e prova a creare nuovamente l'SPN.

Errore di creazione del connettore per AD

La creazione del connettore per AD può fallire per vari motivi. Quando la creazione del connettore non riesce, riceverai il motivo dell'errore nella risposta dell'API. Se utilizzi la console, il motivo dell'errore viene visualizzato nella pagina dei dettagli del connettore nel campo Dettagli aggiuntivi sullo stato all'interno del contenitore dei dettagli del connettore. La tabella seguente descrive i motivi dell'errore e i passaggi consigliati per la risoluzione.

Termini e concetti

I seguenti termini e concetti possono aiutarti mentre lavori con AWS Private Certificate Authority (CA privata AWS).

Argomenti

Trust

Per poter considerare attendibile l'identità di un sito Web, un browser Web deve essere in grado di verificare il certificato di tale sito. I browser, tuttavia, considerano attendibili solo un ristretto numero di certificati noti come certificati root CA. Una terza parte attendibile, nota come autorità di certificazione (CA), convalida l'identità del sito Web ed emette un certificato digitale firmato all'operatore del sito Web. Il browser può quindi verificare la firma digitale per convalidare l'identità del sito Web. Se la convalida riesce, verrà visualizzata un'icona a forma di lucchetto nella barra degli indirizzi.

Certificati del server TLS

Le transazioni HTTPS richiedono certificati del server per autenticare un server. Un certificato del server è una struttura di dati X.509 v3 che associa la chiave pubblica nel certificato all'oggetto del

certificato. Un certificato TLS è firmato da un'autorità di certificazione (CA). Contiene il nome del server, il periodo di validità, la chiave pubblica, l'algoritmo di firma e altro ancora.

Firma del certificato

Una firma digitale è un hash crittografato su un certificato. Una firma viene utilizzata per confermare l'integrità dei dati del certificato. La CA privata crea una firma utilizzando una funzione hash crittografica come SHA256 sul contenuto del certificato di dimensioni variabili. Questa funzione hash produce una stringa di dati di dimensioni fisse in modo efficace imperdonabile. Questa stringa è chiamata hash. La CA effettua la crittografia del valore hash con la propria chiave privata e concatena gli hash crittografati con il certificato.

Autorità di certificazione

Un'autorità di certificazione (CA) emette e, se necessario, revoca i certificati digitali. Il tipo più comune di certificato si basa sullo standard ISO X.509. Un certificato X.509 conferma l'identità dell'oggetto del certificato e associa tale identità a una chiave pubblica. L'oggetto può essere un utente, un'applicazione, un computer o un altro dispositivo. La CA firma il certificato sottoponendo ad hashing i contenuti e crittografando l'hash con la chiave privata correlata alla chiave pubblica nel certificato. Un'applicazione client, ad esempio un browser Web che deve confermare l'identità di un oggetto, utilizza la chiave pubblica per decrittografare la firma del certificato. Esegue quindi l'hashing dei contenuti del certificato e confronta il valore sottoposto ad hashing con la firma decrittografata per stabilire se corrispondono. Per informazioni sulla firma dei certificati, consulta Firma del certificato.

È possibile utilizzare CA privata AWS per creare una CA privata e utilizzarla per l'emissione di certificati. La CA privata emette solo certificati SSL/TLS privati da utilizzare all'interno dell'organizzazione. Per ulteriori informazioni, consulta Certificato privato. Anche la CA privata richiede un certificato prima che sia possibile utilizzarla. Per ulteriori informazioni, consulta Certificato CA.

CA root

Una CA root è un elemento costitutivo di crittografia e la root di affidabilità sulla cui base possono essere emessi i certificati. È composta da una chiave privata per sottoscrivere (emettere) certificati e di un certificato root che identifica la CA root e vincola la chiave privata al nome della CA. Il certificato root viene distribuito negli archivi affidabili di ciascuna entità in un ambiente. Gli amministratori costruiscono archivi attendibili in modo da includere solo le CA attendibili. Gli amministratori

aggiornano o creano gli archivi attendibili nei sistemi operativi, nelle istanze e nelle immagini delle macchine host delle entità nel proprio ambiente. Quando le risorse tentano di connettersi tra loro, verificano i rispettivi certificati presentati. Un client controlla la validità dei certificati e se esiste una catena dal certificato a un certificato root installato nell'archivio attendibilità. Se tali condizioni sono soddisfatte, viene stabilita una «stretta di mano» tra le risorse. Questo handshake dimostra in modo crittografico l'identità di ciascuna entità rispetto all'altra e crea un canale di comunicazione crittografato (TLS/SSL) tra di loro.

Certificato CA

Un certificato di un'autorità di certificazione (CA) conferma l'identità della CA e la associa alla chiave pubblica contenuta nel certificato.

È possibile utilizzare CA privata AWS per creare una CA root privata o una CA subordinata privata, ciascuna supportata da un certificato emesso da una CA. I certificati emessi da una CA subordinata sono firmati da un altro certificato emesso da una CA superiore in una catena di attendibilità. Ma nel caso di una CA root, il certificato è firmato automaticamente. È inoltre possibile stabilire un'autorità root esterna (ospitata in locale, ad esempio). È quindi possibile utilizzare l'autorità principale per firmare un certificato emesso da una CA root subordinata ospitata da CA privata AWS.

L'esempio seguente mostra i campi tipici presenti in un certificato emesso da una CA X.509 CA privata AWS. Si noti che per un certificato CA il valore `CA:` del campo `Basic Constraints` è impostato su 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
```

Un certificato emesso da una CA root

Un'autorità di certificazione (CA) esiste in genere all'interno di una struttura gerarchica che contiene più altre CA con relazioni padre-figlio chiaramente definite tra di loro. Le CA figlio o subordinate sono certificate dalle CA padre, creando così una catena di certificati. La CA al livello più alto della gerarchia è detta CA root e il suo certificato viene denominato certificato root. Questo certificato generalmente è autofirmato.

Certificato di entità finale

Un certificato di entità finale identifica una risorsa, ad esempio un server, un'istanza, un contenitore o un dispositivo. A differenza dei certificati CA, i certificati di entità finale non possono essere utilizzati per emettere certificati. Altri termini comuni per il certificato di entità finale sono «client» o «leaf».

Certificati autofirmati

Un certificato firmato dall'emittente invece di una CA superiore. A differenza dei certificati emessi dalla root protetta mantenuta da una CA, i certificati autofirmati sono essi stessi la root; di conseguenza presentano limitazioni importanti, perché ad esempio possono essere utilizzati per fornire crittografia in transito ma non per verificare le identità; inoltre, non possono essere revocati. Sono inaccettabili dal punto di vista della sicurezza. Ma le organizzazioni li usano comunque perché sono facili da generare, non richiedono competenze o infrastrutture e molte applicazioni li accettano. Non sono disponibili controlli per l'emissione di certificati autofirmati. Le aziende che li usano vanno

incontro a rischi maggiori di interruzioni causate dalla scadenza dei dispositivi, perché non è possibile tenere sotto controllo le date di scadenza.

Certificato privato

CA privata AWSi certificati sono certificati SSL/TLS privati che è possibile utilizzare all'interno dell'organizzazione, ma non sono affidabili sulla rete Internet pubblica. Utilizzarli per identificare le risorse, ad esempio client, server, applicazioni, servizi, dispositivi e utenti. Quando si stabilisce un canale di comunicazione crittografato sicuro, ogni risorsa utilizza un certificato come il seguente e le tecniche crittografiche necessarie per provare la propria identità a un'altra risorsa. Gli endpoint API interni, i server Web, gli utenti VPN, i dispositivi IoT e molte altre applicazioni utilizzano i certificati privati per stabilire canali di comunicazione crittografati necessari per un funzionamento sicuro. Per impostazione predefinita, i certificati privati non sono pubblicamente attendibili. Un amministratore interno deve configurare esplicitamente le applicazioni affinché considerino attendibili i certificati privati e deve distribuire i certificati.

```
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
```

Percorso del certificato

Un client che si basa su un certificato verifica l'esistenza di un percorso dal certificato dell'entità finale, possibilmente attraverso una catena di certificati intermedi, a una radice attendibile. Il client verifica che ogni certificato lungo il percorso sia valido (non revocato). Verifica inoltre che il certificato dell'entità finale non sia scaduto, che abbia integrità (non sia stato manomesso o modificato) e che i vincoli del certificato siano applicati.

Vincolo di lunghezza del percorso

I vincoli di base pathLenConstraintper un certificato CA stabiliscono il numero di certificati CA subordinati che possono esistere nella catena sottostante. Ad esempio, un certificato CA con un vincolo di lunghezza del percorso pari a zero non può avere alcuna CA subordinata. Una CA con un vincolo di lunghezza del percorso di uno può avere al di sotto di essa fino a un livello di CA subordinate. RFC 5280 lo definisce come «il numero massimo di certificati non-self-issued intermedi che possono seguire questo certificato in un percorso di certificazione valido». Il valore della lunghezza del percorso esclude il certificato dell'entità finale, sebbene un linguaggio informale sulla «lunghezza» o la «profondità» di una catena di convalida possa includerlo... generando confusione.

Cronologia dei documenti

La tabella seguente descrive le modifiche importanti apportate a questa documentazione a partire da gennaio 2018. Oltre alle modifiche maggiori elencate qui, aggiorniamo la documentazione di frequente per migliorare le descrizioni e gli esempi e per dar spazio al feedback inviatoci. Per ricevere notifiche sulle modifiche rilevanti, utilizza il collegamento in alto a destra per iscriverti al feed RSS.

Modifica	Descrizione	Data
Nuova guida alla risoluzione dei problemi dei connettori	Sono state aggiunte due nuove sezioni sulla risoluzio ne dei problemi di creazione di connettori e SPN.	4 aprile 2024
Aggiungere l'estensione CDP per Matter	Aggiunge il supporto per l'estensione CDP (Certificate Revocation List Distribution Point) per Matter.	25 gennaio 2024
AWS Private CA Supporto API per mDL	È stato aggiunto il supporto API per la creazione di certificati conformi allo standard ISO/IEC per la patente di guida mobile (mDL).	16 gennaio 2024
AWS Private CA Connettore per Active Directory	Guida per l'utente, API e supporto CLI per Connector for AD. Per ulteriori informazi oni, consulta la documenta zione di Connector for AD.	24 agosto 2023
Modifica dei nomi delle politiche di sicurezza in modo che corrispondano al nuovo nome di servizio	Adozione di nuovi nomi per le policy IAM AWS gestite che specificano le autorizza zioni standard su CA privata AWS. Per ulteriori informazi	13 febbraio 2023

	oni, consulta le politiche AWS gestite.	
Aggiungere un tracker delle modifiche per le politiche AWS gestite	Documentazione aggiunta per tenere traccia delle modifiche alle politiche IAM AWS gestite che specificano le autorizza zioni standard su. CA privata AWS Per ulteriori informazioni, consulta Aggiornamenti alle politiche AWS gestite per CA privata AWS.	11 novembre 2022
Supporto API e CLI per CA che emettono certificati di breve durata	Con l'introduzione delle modalità di utilizzo della CA, una CA può essere configura ta per emettere certificati generici o esclusivamente di breve durata. Per ulteriori informazioni, vedere Modalità di autorità di certificazione.	24 ottobre 2022
Rebranding del servizio e aggiornamento della console	Il servizio viene rinominato in (). AWS Private Certifica te Authority CA privata AWS La CA privata AWS console presenta miglioramenti in termini di usabilità, tra cui pannelli di aiuto integrati che rimandano alla documenta zione completa.	27 settembre 2022

Supporto per certificati conformi a Matter	Tre nuovi modelli di certificato aggiungono il supporto per i certificati CA e per le entità finali conformi a Matter. Per ulteriori informazioni, consulta Comprendere i modelli di certificato.	20 luglio 2022
Supporto di una nuova regione	Endpoint aggiunto per l'Asia Pacifico (Giacarta). Per un elenco completo degli AWS Private CA endpoint, consulta ACM Private Certificate Authority Endpoints and Quotas.	4 maggio 2022
Support per attributi ed estensioni personalizzati	Utilizzate l'CustomAttributeoggetto per configurare CA e certificati personalizzati e l'CustomExtension oggetto per configurare certificati personalizzati.	16 marzo 2022
Support per Managed OCSP	Vedi Configurazione di un metodo di revoca dei certificati per le opzioni di revoca, incluso OCSP.	18 agosto 2021
Support per la funzionalità S3 Block Public Access per i CRL	Vedi Abilitazione della funzione S3 Block Public Access.	27 maggio 2021
Esempi di implementazione Java nuovi e aggiornati	Vedi Utilizzo dell'API CA privata ACM (esempi Java).	9 settembre 2020

Supporto di una nuova regione	Endpoint aggiunti per l'Africa (Città del Capo) e l'Europa (Milano). Per un elenco completo degli CA privata AWS endpoint, consulta AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	27 agosto 2020
È supportato l'accesso privato alla CA tra più account	AWS Certificate Manager gli utenti possono essere autorizzati a emettere certificati utilizzando CA private di cui non sono proprietari. Per ulteriori informazioni, consulta Accesso interaccount alle CA private.	17 agosto 2020
Supporto per endpoint VPC () PrivateLink	È stato aggiunto il supporto per l'uso degli endpoint VPC (AWS PrivateLink) per una maggiore sicurezza della rete. Per ulteriori informazioni, consulta ACM Private CA VPC AWS PrivateLink Endpoints ().	26 marzo 2020
È stata aggiunta una sezione dedicata alla sicurezza	La documentazione sulla sicurezza per è AWS stata consolidata in una sezione dedicata alla sicurezza. Per informazioni sulla sicurezza , consulta Security in AWS Certificate Manager Private Certificate Authority.	26 marzo 2020

Modello ARN aggiunto ai report di controllo.	Per ulteriori informazioni, vedere Creazione di un report di audit per la CA privata.	6 marzo 2020
CloudFormation supporto	Support aggiunto per AWS CloudFormation. Per ulteriori informazioni, consulta ACMPCA Riferimento dei tipi di risorse nella Guida per l'utente AWS CloudFormation .	22 gennaio 2020
CloudWatch Integrazione degli eventi	Integrazione con CloudWatch Events per eventi asincroni, tra cui creazione di CA, emissione di certificati e creazione di CRL. Per ulteriori informazioni, vedere Utilizzo degli eventi. CloudWatch	23 dicembre 2019
Endpoint FIPS	Aggiunti endpoint FIPS per AWS GovCloud (Stati Uniti orientali) e (Stati Uniti occidentali). AWS GovCloud Per un elenco completo degli CA privata AWS endpoint, consulta AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	13 dicembre 2019

Autorizzazioni basate su tag	Autorizzazioni basate su tag supportate utilizzando le nuove API `TagResource` , `UntagResource` e `ListTagsForResource` . Per informazioni generali sui controlli basati su tag, consulta Controllo dell'accesso a e per utenti e ruoli IAM mediante i tag delle risorse IAM.	5 novembre 2019
Applicazione dei vincoli relativi ai nomi	Aggiunto il supporto per l'applicazione dei vincoli del nome del soggetto sui certificati emessi da una CA importati . Per ulteriori informazioni, consulta Enforcing Name Constraints on a Private CA.	28 ottobre 2019
Nuovi modelli di certificato	Sono stati aggiunti nuovi modelli di certificato, inclusi modelli per la firma del codice con AWS Signer. Per ulteriori informazioni, consulta Utilizzo dei modelli.	1 ottobre 2019
Pianificazione della CA	Nuova sezione aggiunta sulla pianificazione della tua PKI utilizzando CA privata AWS. Per ulteriori informazioni, consulta Planning Your ACM Private CA Deployment.	30 settembre 2019

Aggiunta del supporto regionale	È stato aggiunto il supporto regionale per la regione AWS Asia Pacifico (Hong Kong). Per un elenco completo delle aree supportate, consulta AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	24 luglio 2019
È stato aggiunto il supporto completo per la gerarchia CA privata	Il supporto per la creazione e l'hosting di CA root elimina la necessità di un genitore esterno.	20 giugno 2019
Aggiunta del supporto regionale	È stato aggiunto il supporto regionale per le regioni AWS GovCloud (Stati Uniti occidentali e Stati Uniti orientali). Per un elenco completo delle aree supportate, consulta AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	8 maggio 2019
Aggiunta del supporto regionale	È stato aggiunto il supporto regionale per le regioni AWS Asia Pacifico (Mumbai e Seoul), Stati Uniti occidentali (California settentrionale) e UE (Parigi e Stoccolma). Per un elenco completo delle regioni supportate, consulta AWS Certificate Manager Private Certificate Authority Endpoints and Quotas.	4 aprile 2019

Test del flusso di lavoro per il rinnovo	I clienti ora possono testare manualmente la configura zione del loro flusso di rinnovo gestito da ACM. Per ulteriori informazioni, consulta Test della configurazione dei rinnovi gestiti di ACM.	14 marzo 2019
Aggiunta del supporto regionale	È stato aggiunto il supporto regionale per la regione AWS UE (Londra). Per un elenco completo delle aree supportat e, consulta AWS Certificate Manager Private Certifica te Authority Endpoints and Quotas.	1 agosto 2018
Ripristina le CA eliminate	Il ripristino della CA privata consente ai clienti di ripristin are le autorità di certificazione (CA) per un massimo di 30 giorni dopo la loro eliminazi one. Per ulteriori informazioni, consulta la sezione relativa al ripristino della CA privata.	20 giugno 2018

Aggiornamenti precedenti

La tabella seguente descrive la cronologia dei rilasci della documentazione AWS Private Certificate Authority precedenti a giugno 2018.

Modifica	Descrizione	Data
Nuova guida	Questa versione introduce AWS Private Certificate Authority.	04 Aprile 2018

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.