使用者指南

AWS Private Certificate Authority

版本 latest

AWS Private Certificate Authority: 使用者指南

Table of Contents

什麼是 AWS 私有 CA？ 1
什麼是最符合我需求的憑證服務？ 1
區域 2
整合服務 2
支援的演算法 3
配額 4
RFC 合規性 4
定價 6
安全 7
IAM 7
API 許可 8
AWS 受管理政策 13
客戶受管政策 18
內嵌政策 19
跨帳戶存取權 25
資源型政策 25
資料保護 29
AWS 私有 CA 私密金鑰的儲存與安全性合規 30
使用中目錄的 AWS Private CA 連接器中的資料加密 30
法規遵循驗證 30
建立稽核報告 31
基礎架構安全 38
VPC 端點 (AWS PrivateLink) 38
日誌記錄和監控 42
CloudWatch 度量 42
使用 CloudWatch 事件 43
使用 CloudTrail 49
規劃私有 CA 70
AWS 帳戶和 CLI 70
註冊一個 AWS 帳戶 71
建立具有管理權限的使用者 71
安裝 AWS Command Line Interface 72
設計 CA 階層 72
驗證最終實體憑證 74

規劃 CA 階層的結構 75
在憑證路徑上設定長度限制 78
管理 CA 生命週期 80
選擇有效期 80
管理 CA 繼任 81
撤銷 CA 82
撤銷 82
撤銷組態的一般需求 84
CRL 設定 84
OCSP 客製化 94
CA 模式 96
一般目的 (預設) 96
短期生活證書 97
恢復能力 97
備援與災難復原 97
最佳實務 99
記錄 CA 結構和政策 99
如果可能的話，盡量減少使用根 CA 99
為根 CA 提供自己的 AWS 帳戶 100
單獨的管理員和發行者角 100
實作憑證的管理撤銷 100
開啟 AWS CloudTrail 100
旋轉 CA 私密金鑰 101
刪除未使用的 CA 101
封鎖對 CRL 的公開存取 101
Amazon EKS 應用程式最佳實務 101
CA 管理 102
建立私有 CA 102
主控台程序 103
CLI 程序 108
使用 CloudFormation 121
安裝 CA 憑證 121
相容簽署演算法 121
安裝根 CA 憑證 123
安裝由主控的從屬 CA 憑證 AWS 私有 CA 130
安裝由外部父 CA 簽署的從屬 CA 憑證 132

控制存取 132
為 IAM 使用者建立單一帳戶許可 133
附加跨帳戶存取政策 135
列出私有 CA 137
檢視一個 CA 139
新增標籤 141
更新 CA 143
正在更新 CA 狀態 143
更新 CA（控制台） 146
更新一個 CA 149
刪除 CA 157
還原 CA 158
還原私有 CA (主控台) 158
還原私有 CA (AWS CLI) 159
憑證管理 161
簽發私人終端實體證書 161
簽發標準證書（AWS CLI） 162
使用 API 傳遞範本發行具有自訂主體名稱的憑證 164
使用 API 傳遞範本發行具有自訂擴充功能的憑證 167
擷取私人憑證 168
列出私有憑證 169
匯出憑證 174
撤銷私人憑證 174
已撤銷的憑證和 OCSP 175
CRL 中的已撤銷憑證 175
稽核報告中的已撤銷憑證 176
自動匯出 177
憑證範本 178
模板品種 178
操作模板順序 189
範本定義 190
使用 API (Java 範例) 229
以程式設計方式建立並啟動根 CA 230
以編程方式創建和激活下屬 CA 238
CreateCertificateAuthority 248
用 CreateCertificateAuthority來支援使用中目錄 252

CreateCertificateAuthorityAuditReport 260
CreatePermission 263
DeleteCertificateAuthority 265
DeletePermission 267
DeletePolicy 269
DescribeCertificateAuthority 272
DescribeCertificateAuthorityAuditReport 274
GetCertificate 277
GetCertificateAuthorityCertificate 279
GetCertificateAuthorityCsr 281
GetPolicy 284
ImportCertificateAuthorityCertificate 286
IssueCertificate 289
ListCertificateAuthorities 292
ListPermissions 296
ListTags 298
PutPolicy 300
RestoreCertificateAuthority 303
RevokeCertificate 304
TagCertificateAuthorities 306
UntagCertificateAuthority 309
UpdateCertificateAuthority 311
使用自訂主體名稱建立 CA 和憑證 313
使用建立 CA CustomAttribute 314
發行憑證 CustomAttribute 318
使用自訂擴充功能建立憑證 321
使用啟動下屬 CA NameConstraints 擴展 321
簽發具有 QC 語句擴展的證書 332
實現物質（Java 示例） 337
啟動產品驗證授權單位 (PAA) 338
啟用產品驗證中間 (PAI) 348
建立裝置驗證憑證 (DAC) 359
為節點操作憑證 (NOC) 啟動根 CA。 363
啟動節點作業憑證 (NOC) 的從屬 CA 373
建立節點作業憑證 (NOC) 383
實作 MDL (Java 範例) 388

啟動發行機構憑證授權單位 (IACA) 憑證 388
建立文件簽署者憑證 397
使用外部 CA 403
保護庫伯尼特 407
跨帳戶使用憑證管理員 409
支援的憑證範本 409
範例解決方 410
廣告用連接器 30
什麼是 AD 連接器？ 411
您是 AD 使用者的首次使用連接器嗎？ 411
存取 AD 的連接器 411
AD 連接器的定價 411
開始使用 412
必要條件 412
建立連接器 419
設定廣告 419
建立範本 420
管理 AD 群組權限 420
程序 420
建立連接器 421
建立範本 423
列出連接器 430
列表模板 431
檢視連接器 431
檢視範本 432
目錄註冊 434
群組和權限 436
服務委託人名稱 437
Tags (標籤) 438
連接器應用於 SCEP 439
什麼是適用於 SCEP 的連接器？ 439
用於 SCEP 的連接器的特點 439
如何開始使用適用於 SCEP 的連接器 440
相關服務 440
存取用於 SCEP 的連接器 440
適用於 SCEP 的連接器定價 440

概念 441
運作方式 442
一般用途 442
AWS Private Certificate Authority 適用於 Microsoft Intune 的 SCEP 連接器 443
考量與限制 444
考量事項 444
限制 445
設定 446
步驟 1：建立 AWS Identity and Access Management 策略 446
步驟 2：建立私有 CA 447
步驟 3：建立資源共用 448
開始使用 449
開始之前 449
步驟 1：建立連接器 449
步驟 2：將連接器詳細資料複製到 MDM 系統 451
MDM 系統 451
賈姆夫專業版 452
Microsoft 455
故障診斷 459
簽署 CSR 459
OCSP 回應中的延遲 459
Amazon S3 塊 CRL 存儲桶 459
撤銷自我簽署的 CA 憑證 460
處理例外狀況 460
使用物質標準 462
AD 錯誤和失敗的連接器 463
錯誤 464
連接器建立失敗 468
SPN 建立失敗 470
AD 連接器建立失敗錯誤的連接器 468
術語和概念 472
信任 472
TLS 伺服器憑證 472
憑證簽章 473
憑證授權單位 473
根 CA 473

加拿大證書 473
根 CA 憑證 474
終端實體證書 475
自我簽署的憑證 475
私人證書 475
憑證路徑 476
路徑長度約束 476
文件歷史記錄 477
舊版更新 482
..... cdlxxxiii

什麼是 AWS 私有 CA？

AWS 私有 CA 允許建立私有憑證授權單位 (CA) 階層，包括根 CA 和從屬 CA，而不需要操作內部部署 CA 的投資和維護成本。您的私有 CA 可以發行終端實體 X.509 憑證，在下列案例中很實用：

- 建立加密的 TLS 通訊通道
- 對使用者、電腦、API 端點和 IoT 裝置進行身分驗證
- 透過加密技術簽署程式碼
- 實作線上憑證狀態通訊協定 (OCSP) 以取得憑證撤銷狀態

AWS 私有 CA 作業可從存取 AWS Management Console、使用 AWS 私有 CA API 或使用 AWS CLI.

主題

- 什麼是最符合我需求的憑證服務？
- 區域
- 服務整合 AWS Private Certificate Authority
- 支援的加密演算法
- 配額
- RFC 合規性
- 定價

什麼是最符合我需求的憑證服務？

有兩種用於發行和部署 X.509 憑證的 AWS 服務。選擇最符合您需求的服務。考量事項包括您是否需要公開或私有的憑證、自訂憑證、要部署到其他 AWS 服務的憑證，或是自動化憑證管理和更新。

1. AWS 私有 CA — 此服務適用於在 AWS 雲端內建置公有金鑰基礎結構 (PKI)，並預計供組織內私人使用的企業客戶。使用時 AWS 私有 CA，您可以建立自己的 CA 階層並發行憑證，以驗證內部使用者、電腦、應用程式、服務、伺服器和其他裝置，以及簽署電腦程式碼。由私有 CA 發行的憑證，只會在您的組織內受信任，而不會在網際網路上受信任。

   建立私有 CA 之後，您可以直接發行憑證 (也就是不需要從第三方 CA 取得驗證)，並進行自訂以符合組織的內部需求。例如，建議您：

   - 使用任何主體名稱建立憑證。

- 使用任何過期日期建立憑證。
- 使用任何支援的私密金鑰演算法和金鑰長度。
- 使用任何支援的簽署演算法。
- 使用範本來控制憑證發行。

您正位於此服務的正確位置。若要開始使用，請登入 https://console.aws.amazon.com/acm-pca/ 主控台。

2. AWS Certificate Manager (ACM) — 此服務會為需要使用 TLS 的公開信任安全網站狀態的企業客戶管理憑證。您可以將 ACM 憑證部署到 E AWS lastic Load Balancing、Amazon CloudFront、Amazon API Gateway 和其他整合式服務。最常見的這類應用是具有龐大流量要求的安全公有網站。

   透過此服務，您可以使用 ACM 提供的公有憑證 (&ACM; 憑證) 或匯入 ACM 的憑證。如果您使 AWS 私有 CA 用建立 CA，ACM 可以管理來自該私有 CA 的憑證發行，並自動執行憑證續約。

   如需詳細資訊，請參閱 AWS Certificate Manager 使用者指南。

區域

與大多數 AWS 資源一樣，私有憑證授權單位 (CA) 是區域資源。若要在多個區域使用私有 CA，您必須在這些區域建立 CA。您無法在區域間複製私有 CA。請瀏覽AWS AWS 一般參考或AWS 區域表中的區域和端點，以查看的區域可用性 AWS 私有 CA。

> **Note**
>
> ACM 目前在某些不適用的區域 AWS 私有 CA 提供。

服務整合 AWS Private Certificate Authority

如果您使用 AWS Certificate Manager 要求私人憑證，您可以將該憑證與任何與 ACM 整合的服務產生關聯。這適用於鏈結至 AWS 私有 CA 根憑證以及鏈結至外部根的憑證。如需詳細資訊，請參閱 AWS Certificate Manager 使用指南中的整合式服務。

您也可以將私有 CA 整合至 Amazon Elastic Kubernetes Service，以便在 Kubernetes 叢集內提供憑證簽發。如需詳細資訊，請參閱 使用保護庫伯尼特 AWS 私有 CA。

> **Note**
>
> Amazon Elastic Kubernetes Service 不是 ACM 集成服務。

如果您使用 AWS 私有 CA API 或發 AWS CLI 行憑證或從 ACM 匯出私有憑證，您可以在任何您想要的位置安裝憑證。

支援的加密演算法

AWS 私有 CA 支援下列密碼編譯演算法，用於產生私密金鑰和憑證簽章。

支持的算法

私密金鑰演算法	簽署演算法
RSA_2048	沙 256 威塞德薩
RSA_4096	沙 384 与埃克萨
普里梅 256v1	沙 512 與艾薩
EC_SEP384R1 SM2 (僅限中國區域)	SHA256WITHRSA SHA384WITHRSA SHA512WITHRSA 小 3 與小 2

此清單僅適用於 AWS 私有 CA 透過其主控台、API 或命令列直接發行的憑證。使用 CA 發行憑證時 AWS Certificate Manager AWS 私有 CA，它支援部分 (但不是全部) 演算法。如需詳細資訊，請參閱 AWS Certificate Manager 使用者指南中的要求私人憑證。

> **Note**
>
> 在所有情況下，指定的簽署演算法系列 (RSA 或 ECDSA) 都必須符合 CA 私密金鑰的演算法系列。

配額

AWS 私有 CA 將配額指派給您允許的憑證和憑證授權單位數目。API 動作的要求率也會受到配額限制。 AWS 私有 CA 配額特定於 AWS 帳戶和區域。

AWS 私有 CA 根據 API 操作，以不同的速率限制 API 請求。節流表示 AWS 私有 CA 拒絕其他有效的要求，因為要求超過作業的每秒要求數量配額。當請求被限制時， AWS 私有 CA 返回一個ThrottlingException錯誤。 AWS 私有 CA 不保證 API 的最低請求率。

若要查看可以調整哪些配額，請參閱中的AWS 私有 CA 配額表AWS 一般參考。

您可以使用「 AWS Service Quotas」檢視目前的配額和要求增加的配額。

查看 AWS 私有 CA 配額 up-to-date 清單

1. 登錄到您的 AWS 帳戶。
2. 開啟 Service Quotas 主控台，網址為 https://console.aws.amazon.com/servicequotas/。
3. 在 [服務] 清單中，選擇 [Cer AWS tificate Manager 專用憑證授權單位 (ACM PCA)]。「服務配額」清單中的每個配額都會顯示您目前套用的配額值、預設配額值，以及配額是否可調整。選擇配額名稱以取得更多相關資訊。

請求提高配額

1. 在 [服務配額] 清單中，選擇可調整配額的圓鈕。
2. 選擇要求增加配額按鈕。
3. 填寫並提交申請配額增加表格。

AWS 私有 CA 與整合 AWS Certificate Manager。您可以使用 ACM 主控台 AWS CLI或 ACM API 從現有的私有 CA 要求私有憑證。這些由 ACM 管理的私有 PKI 憑證會受到 PCA 配額的約束，以及 ACM 在公用憑證和匯入憑證上設定的配額。如需 ACM 需求的詳細資訊，請參閱 AWS Certificate Manager 使用者指南中的要求私人憑證和配額。

RFC 合規性

AWS 私有 CA 不會強制執行 RFC 52 80 中定義的某些限制。相反的情況也是如此：適用於私有 CA 的某些額外限制會強制執行。

強制執行

- 不在日期後。為了符合 RFC 5280 的規範，在發行 CA 憑證的 `Not After` 日期後， AWS 私有 CA 會防止發行具有 `Not After` 的憑證。
- 基本限制。 AWS 私有 CA 在匯入的 CA 憑證中強制執行基本限制和路徑長度。

  基本限制條件會指出憑證所識別的資源是否為 CA 且可以發行憑證。匯入 AWS 私有 CA 的 CA 憑證必須包含基本限制延伸，且延伸必須標記為 `critical`。除了`critical`標誌之外，還`CA=true`必須設置。 AWS 私有 CA 由於下列原因，驗證例外狀況失敗，強制執行基本限制：

  - 延伸不包含在 CA 憑證中。
  - 延伸未標記為 `critical`。

  路徑長度 (pathLenConstraint) 決定從匯入的 CA 憑證下游可能存在多少從屬 CA。 AWS 私有 CA 由於以下原因，驗證異常失敗來強制執行路徑長度：

  - 匯入 CA 憑證會違反 CA 憑證中 (或鏈結中的任何 CA 憑證中) 的路徑長度限制條件。
  - 發行憑證將違反路徑長度限制條件。
- 名稱限制表示名稱空間，必須在其中找到憑證路徑中後續憑證中的所有主體名稱。限制適用於主體辨別名稱和主體替代名稱。

未強制執行

- 憑證原則。憑證原則會規範 CA 發行憑證的條件。
- 禁止任何政策。用於發行給 CA 的憑證。
- 發行人替代名稱。允許其他身分與 CA 憑證的簽發者相關聯。
- 策略限制。這些限制條件會限制 CA 發行次級 CA 憑證的容量。
- 策略對應。用於 CA 憑證。列出一或多對 OID；每對包括 issuerDomainPolicy 和。subjectDomainPolicy
- 主旨目錄屬性。用於傳達主體的識別屬性。
- 主旨資訊存取。如何存取出現延伸功能之憑證主旨的資訊和服務。
- 主體金鑰識別符 (SKI) 和授權機構金鑰識別符 (AKI)。RFC 需要 CA 憑證，才能包含 SKI 延伸。CA 核發的憑證必須包含符合 CA 憑證 SKI 的 AKI 副檔名。 AWS 不會強制執行這些要求。如果您的 CA 憑證不包含 SKI，則發行的終端實體或次級 CA 憑證 AKI，將會改為發行者公有金鑰的 SHA-1 雜湊。
- SubjectPublicKeyInfo和主旨別名 (SAN)。發行憑證時，從提供的 CSR AWS 私有 CA 複製 SubjectPublicKeyInfo 和 SAN 延伸，而不執行驗證。

定價

從您帳戶的建立時間開始，會針對每個私有 CA 每月向您收取費用。也會向您收取您發行的每個憑證費用。此費用包括您從 ACM 匯出的憑證，以及您從 AWS 私有 CA API 或 AWS 私有 CA CLI 建立的憑證。私有 CA 刪除後，您不需付費。不過，如果您還原私有 CA，您需支付刪除到還原這段期間的費用。您無權存取其私密金鑰的私有憑證是免費的。其中包括與整合式服務 (例如 Elastic Load Balancing) 和 API Gateway 搭配使用的憑證。 CloudFront

如需最新 AWS 私有 CA 定價資訊，請參閱AWS Private Certificate Authority 定價。您也可以使用定AWS 價計算器來估算成本。

中的安全性 AWS Private Certificate Authority

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，這些架構是為了滿足對安全性最敏感的組織的需求而建置的。

安全是 AWS 與您之間共同的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。若要瞭解適用於的規範遵循方案 AWS Private Certificate Authority，請參閱AWS 服務 遵循規範計劃。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您瞭解如何在使用時套用共同責任模型 AWS 私有 CA。下列主題說明如何設定 AWS 私有 CA 以符合安全性與合規性目標。您還將學習如何使用其 AWS 服務 他幫助您監控和保護 AWS 私有 CA 資源的其他方法。

主題

- 適用於的 Identity and Access Management (IAM) AWS Private Certificate Authority
- 跨帳戶存取私有 CA 的安全性最佳做法
- 資料保護 AWS Private Certificate Authority
- AWS Private Certificate Authority的法規遵循驗證
- 基礎結構安全 AWS Private Certificate Authority
- AWS Private Certificate Authority的記錄和監控

適用於的 Identity and Access Management (IAM) AWS Private Certificate Authority

存取權 AWS 私有 CA 需要 AWS 可用來驗證您的請求的憑證。以下主題提供如何使用 AWS Identity and Access Management (IAM) 的詳細資訊，藉由控制可存取的人員，協助確保私有憑證授權機構 (CA) 的安全。

在中 AWS 私有 CA，您使用的主要資源是憑證授權單位 (CA)。您擁有或控制的每個私有 CA 皆是以 Amazon Resource Name (ARN) 識別，其格式如下。

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

資源擁有者是在其中建立 AWS 資源之 AWS 帳號的主參與者實體。下列範例說明其如何運作。

- 如果您使用的認證 AWS 帳戶根使用者 來建立私有 CA，則您的 AWS 帳戶擁有該 CA。

> **Important**
>
> - 我們不建議使用 AWS 帳戶根使用者 建立 CA。
> - 我們強烈建議您隨時使用多重要素驗證 (MFA)。 AWS 私有 CA

- 如果您在 AWS 帳戶中建立 IAM 使用者，您可以授與該使用者建立私有 CA 的權限。不過，該 CA 歸使用者所屬的帳戶所有。
- 如果您在 AWS 帳戶中建立 IAM 角色，並授予其建立私有 CA 的權限，則任何可以擔任該角色的人都可以建立 CA。不過，該私有 CA 歸角色所屬的帳戶所有。

許可政策描述誰可以存取哪些資源。以下討論會說明可用來建立許可政策的選項。

> **Note**
>
> 本文件討論在的內容中使用 IAM AWS 私有 CA。它不提供 IAM 服務的詳細資訊。如需完整的 IAM 文件，請參閱 IAM 使用者指南。如需 IAM 政策語法和說明的詳細資訊，請參閱 AWS IAM 政策參考資料。

AWS 私有 CA API 操作和權限

當您設定計劃附加至 IAM 身分 (身分型政策) 的存取控制和許可政策時，請使用下表作為參考。表格中的第一欄會列出每個 AWS 私有 CA API 作業。您可以在政策的 `Action` 元素中指定動作。其餘欄位提供其他資訊。

AWS 私有 CA API 作業	所需的許可	資源
CreateCertificateAuthority	`acm-pca:CreateCertificateAuthority`	`arn:aws:acm-pca: us-east-1 :111122223333 :certificate-`

AWS 私有 CA API 作業	所需的許可	資源
	acm-pca:TagCertificateAuthority (只有在建立含標籤的 CA 時才需要)。	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/ *11223344-1234-1122-2233-112233445566*
DeleteCertificateAuthority	acm-pca:DeleteCertificateAuthority	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
DeletePermission	acm-pca:DeletePermission	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*

AWS 私有 CA API 作業	所需的許可	資源
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*
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*

AWS 私有 CA API 作業	所需的許可	資源
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*
ImportCertificateAuthorityCertificate	acm-pca:ImportCertificateAuthorityCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
IssueCertificate	acm-pca:IssueCertificate	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListCertificateAuthorities	acm-pca:ListCertificateAuthorities	N/A

AWS 私有 CA API 作業	所需的許可	資源
ListPermissions	acm-pca:ListPermissions	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
ListTags	acm-pca:ListTags	N/A
PutPolicy	acm-pca:PutPolicy	arn:*aws*:acm-pca: *us-east-1* :*111122223333* :certificate-authority/ *11223344-1234-1122-2233-112233445566*
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*

AWS 私有 CA API 作業	所需的許可	資源
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*

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 使用者和群組位於 AWS IAM Identity Center：

  建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 建立權限合集 說明進行操作。

- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。請按照 IAM 使用者指南 的 為第三方身分提供者 (聯合) 建立角色 中的指示進行操作。

- IAM 使用者：
  - 建立您的使用者可擔任的角色。請按照 IAM 使用者指南 的 為 IAM 使用者建立角色 中的指示進行操作。
  - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南的新增許可到使用者 (主控台)中的指示。

AWS 受管理政策

AWS 私有 CA 包含一組針對 AWS 管理 AWS 私有 CA 員、使用者和稽核者預先定義的受管理原則。了解這些政策有助於您實作 客戶受管政策。

選擇下面列出的任何策略以查看詳細信息和示例策略代碼。

AWSPrivateCAFullAccess

授予不受限制的管理控制。

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

AWSPrivateCAReadOnly

授予僅限於唯讀 API 操作的訪問權限。

```
{
   "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

授予發行和撤銷 CA 憑證的能力。這項政策沒有其他管理功能，且無法發行終端實體憑證。其許可與 User 政策互斥。

```
{
   "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

授予發行和撤銷終端實體憑證的能力。這項政策沒有管理功能，且無法發行 CA 憑證。權限與PrivilegedUser原則互斥。

```
{
   "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

授與唯讀 API 作業的存取權，以及產生 CA 稽核報告的權限。

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

AWS 受管理策略的更新 AWS 私有 CA

在下表中，檢視有關自服務開始追蹤這些變更以 AWS 私有 CA 來 AWS 受管理原則的更新詳細資訊。如需有關所有變更的自動警示 AWS 私有 CA，請訂閱文件歷史記錄頁面上的 RSS 摘要。

受管理原則變更

變更	描述	日期
新策略名稱： • AWSPrivateCAFullAccess • AWSPrivateCAReadOnly • AWSPrivateCAPrivilegedUser • AWSPrivateCAAuditor • AWSPrivateCAUser	策略名稱前綴已從變更AWSCertificateManagerPrivateCA 為AWSPrivateCA 。 功能保持不變。	2023 年 2 月 13 日

客戶受管政策

作為最佳實踐，不要使用您與之 AWS 帳戶根使用者 進行交互 AWS，包括 AWS 私有 CA. 而是使用 AWS Identity and Access Management (IAM) 建立 IAM 使用者、IAM 角色或聯合身分使用者。建立管理員群組，並將自己新增至該群組。然後，以管理員身分登入。視需要將其他使用者新增至該群組。

另一個最佳做法是建立可指派給使用者的客戶受管 IAM 政策。客戶受管政策是您建立的獨立身分識別型政策，您可以將政策連接到 AWS 帳戶中的多個使用者、群組或角色。這種策略會限制使用者只能執行您指定的 AWS 私有 CA 動作。

以下範例客戶受管政策會允許使用者建立 CA 稽核報告。以下僅為範例。您可以選擇所需的任何 AWS 私有 CA 操作。如需更多範例，請參閱內嵌政策。

若要建立客戶受管政策

1. 使用 AWS 管理員的登入資料登入 IAM 主控台。
2. 在主控台的導覽窗格中，選擇 Policies (政策)。
3. 選擇 Create policy (建立政策)。
4. 請選擇 JSON 標籤。
5. 請複製以下政策，然後在編輯器中貼上。

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

6. 選擇檢閱政策。
7. 在 Name (名稱) 輸入 `PcaListPolicy`。
8. (選用) 輸入描述。
9. 選擇建立政策。

管理員可以將政策附加到任何 IAM 使用者，以限制使用者可以執行的 AWS 私有 CA 動作。如需套用許可政策的方法，請參閱 IAM 使用者指南中的變更 IAM 使用者的許可。

內嵌政策

內嵌政策是您建立及管理的政策，且直接內嵌至單一使用者、群組或角色。下列原則範例顯示如何指派執行 AWS 私有 CA 動作的權限。如需有關內嵌政策的一般資訊，請參閱 IAM 使用者指南中的使用內

嵌政策。您可以使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 IAM API 建立和嵌入內嵌政策。

> **Important**
>
> 我們強烈建議您隨時使用多重要素驗證 (MFA)。 AWS 私有 CA

主題

- 列出私有 CA
- 擷取私有 CA 憑證
- 匯入私有 CA 憑證
- 刪除私有 CA
- Tag-on-create：建立時將標籤附加至 CA
- Tag-on-create：限制標記
- 使用標記控制私有 CA 的存取
- 唯讀存取權 AWS 私有 CA
- 完全存取 AWS 私有 CA
- 以管理員身分存取所有 AWS 資源

列出私有 CA

以下政策可讓使用者列出帳戶中的所有私有 CA。

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

擷取私有 CA 憑證

以下政策可讓使用者擷取特定私有 CA 憑證。

```
{
   "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"
   }
}
```

匯入私有 CA 憑證

以下政策可讓使用者匯入私有 CA 憑證。

```
{
   "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"
   }
}
```

刪除私有 CA

以下政策可讓使用者刪除特定私有 CA。

```
{
   "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：建立時將標籤附加至 CA

下列原則可讓使用者在 CA 建立期間套用標記。

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

Tag-on-create：限制標記

下列 tag-on-create 原則會防止在 CA 建立期間使用金鑰值組環境 =Prod。允許使用其他鍵值對進行標記。

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

```
      }
   ]
}
```

使用標記控制私有 CA 的存取

下列原則只允許存取具有索引鍵值組環境 PreProd = 的 CA。它還要求新的 CA 包含此標籤。

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

唯讀存取權 AWS 私有 CA

以下政策可讓使用者描述及列出私有憑證授權機構，並擷取私有 CA 憑證和憑證鏈。

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

完全存取 AWS 私有 CA

下列原則可讓使用者執行任何 AWS 私有 CA 動作。

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

以管理員身分存取所有 AWS 資源

以下策略允許使用者對任何 AWS 資源執行任何動作。

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

跨帳戶存取私有 CA 的安全性最佳做法

AWS 私有 CA 管理員可以與其他 AWS 帳戶中的主參與者 (使用者、角色等) 共用 CA。當已收到並接受共用時，主體可以使用 CA 使用 AWS 私有 CA 或 AWS Certificate Manager 資源來發行最終實體憑證。主體可以使用 CA 來發行從屬 CA 憑證。 AWS 私有 CA

> **Important**
>
> 與跨帳戶案例中發行的憑證相關的費用，會向發行憑證的 AWS 帳戶收取費用。

若要共用 CA 的存取權， AWS 私有 CA 管理員可以選擇下列其中一種方法：

- 使用 AWS Resource Access Manager (RAM) 與另一個帳號中的主參與者共用 CA 做為資源，或與其他帳號中的主參與者共用 AWS Organizations。RAM 是跨帳戶共用資 AWS 源的標準方法。如需有關 RAM 的詳細資訊，請參閱使AWS RAM 用者指南。若要取得有關的更多資訊 AWS Organizations，請參閱AWS Organizations 使用者指南。
- 使用 AWS 私有 CA API 或 CLI 將以資源為基礎的政策附加到 CA，從而授與另一個帳戶中主體的存取權。如需詳細資訊，請參閱 資源型政策。

本指南控制對私有 CA 的存取章節提供在單一帳戶和跨帳戶案例中授與 CA 存取權的工作流程。

資源型政策

以資源為基礎的策略是您建立並手動附加至資源 (在此情況下為私有 CA) 的權限原則，而非使用者身分識別或角色。或者，您可以將 AWS 受管政策用於，而不是建立自己的原則 AWS Private CA。使用套用 AWS RAM 以資源為基礎的策略， AWS 私有 CA 管理員可以直接或透過 AWS Organizations與不同 AWS 帳戶中的使用者共用 CA 的存取權。或者， AWS 私有 CA 系統管理員也可以使用 PCA API PutPolicyGetPolicy、和或對應的 AWS CLI 命令放入原則DeletePolicy、取得原則和刪除原則，套用和管理以資源為基礎的原則。

如需以資源為基礎的原則的一般資訊，請參閱以身分識別為基礎的原則和以資源為基礎的原則和使用原則控

若要檢視以資源為基礎的 AWS 受管理策略清單 AWS Private CA，請導覽至 AWS Resource Access Manager 主控台中的受管理權限程式庫，然後搜尋CertificateAuthority。如同任何原則，我們建議您在套用原則之前，先在測試環境中套用原則，以確保其符合您的需求。

AWS Certificate Manager (ACM) 對私有 CA 具有跨帳戶共用存取權的使用者可以發行 CA 簽署的受管理憑證。跨帳戶發行者受到以資源為基礎的政策的限制，並且只能存取下列最終實體憑證範本：

- EndEntityCertificate/1
- EndEntityClientAuthCertificate/1
- EndEntityServerAuthCertificate/1
- BlankEndEntityCertificate_API通過/V1
- BlankEndEntityCertificate_APICSR通過/V1
- 次級PathLen緩存證書 _ 0/V1

政策範例

本節提供各種需求的跨帳戶策略範例。在所有情況下，都會使用下列命令模式來套用原則：

```
$ 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
```

除了指定 CA 的 ARN 之外，系統管理員還提供 AWS 帳戶識別碼或將授與 CA 存取權的 AWS Organizations 識別碼。下列每個原則的 JSON 格式化為可讀性的檔案，但也可以作為內嵌 CLI 引數提供。

> **Note**
>
> 下面顯示的 JSON 資源為基礎的政策的結構必須精確遵循。客戶只能設定主參與者的 ID 欄位 (AWS 帳號或 Organ AWS izations 識別碼) 和 CA ARN。

1. 文件：政策1.json-與不同帳戶中的用戶共享對 CA 的訪問權限

   將 *5555555555* 取代為共用 CA 的 AWS 帳戶識別碼。

   對於資源 ARN，請用您自己的值替換以下內容：

   - *aws*- AWS 分區 例如，awsaws-us-gov、aws-cn、等。
   - *us-east-1*-資源可用的 AWS 區域，例如us-west-1。

- *111122223333*-資源擁有者的 AWS 帳號 ID。
- *11223344-1234-1122-2233-112233445566*-憑證授權單位的資源 ID。

```
{
  "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. 文件：政策 2.json-通過共享對 CA 的訪問 AWS Organizations

   用識別碼取代 *o-a1b2* c3d4z5。 AWS Organizations

   對於資源 ARN，請用您自己的值替換以下內容：

   - *aws*- AWS 分區 例如，awsaws-us-gov、aws-cn、等。
   - *us-east-1*-資源可用的 AWS 區域，例如us-west-1。
   - *111122223333*-資源擁有者的 AWS 帳號 ID。
   - *11223344-1234-1122-2233-112233445566*-憑證授權單位的資源 ID。

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

資料保護 AWS Private Certificate Authority

AWS 共用責任模型適用於中的資料保護 AWS Private Certificate Authority。如此模型中所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用主控台、API AWS 私有 CA 或 AWS SDK 時 AWS 服務 使用或其他使用時。 AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

AWS 私有 CA 私密金鑰的儲存與安全性合規

私密 CA 的私密金鑰會儲存在 AWS 受管理的硬體安全性模組 (HSM) 中。HSM 符合 FIPS PUB 140-2 密碼編譯模組的第 3 級安全性需求。

使用中目錄的 AWS Private CA 連接器中的資料加密

AWS Private CA AD 的連接器會儲存有關連接器、範本、目錄註冊、服務主要名稱和範本群組存取控制項目的客戶組態資料。此資料在傳輸過程中和靜態時都會加密。您可以使用 AWS Private CA API 中的動作探索透過 AD 連接器發GetCertificate行之憑證的相關資訊。不會儲存有關已發行憑證的資訊，或與要求憑證的用戶端或機器相關的資訊 AWS。

AWS Private Certificate Authority的法規遵循驗證

協力廠商稽核人員會評估其安全性與合規性， AWS Private Certificate Authority 做為多個 AWS 合規計畫的一部分。這些計劃包括 SOC、PCI、FedRAMP、HIPAA 等等。

如需特定法規遵循方案範圍內的 AWS 服務清單，請參閱合規性計劃AWS 服務 範圍內的AWS 服務 (依法規)。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 私有 CA 於您資料的敏感性、公司的合規目標以及適用的法律和法規。 AWS 提供下列資源以協助遵循法規：

- 對於需要加密 Amazon S3 儲存貯體的組織，下列主題說明如何設定加密以容納 AWS 私有 CA 資產：
  - 加密您的稽核報告
  - 加密您的 CRL
- 安全性與合規快速入門指南 — 這些部署指南討論架構考量，並提供在上部署以安全性和法規遵循為重點的基準環境的步驟。 AWS
- 建構 HIPAA 安全性與合規性白皮書 — 本白皮書說明公司如何使用建立符合 HIPAA 標準的應用 AWS 程式。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub— 此 AWS 服務提供安全狀態的全面檢視，協助您檢查您 AWS 是否符合安全性產業標準和最佳做法。

將稽核報告與您的私有 CA 搭配使用

您可以建立稽核報告，以列出私有 CA 已發行或撤銷的所有憑證。報告會儲存在新的或輸入時指定的現有 S3 儲存貯體。

如需為您稽核報告新增加密保護的資訊，請參閱 加密您的稽核報告 。

稽核報告檔案具有以下路徑和檔案名稱。Amazon S3 存儲桶的 ARN 是的價值。bucket-name CA_ID是發行 CA 的唯一識別碼。 UUID是稽核報告的唯一識別碼。

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

每 30 分鐘可從您的儲存貯體產生及下載新的報告。以下範例顯示以 CSV 分隔的報告。

```
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
```

JSON 格式的報告如下列範例所示。

```
[
   {
      "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

更 AWS Certificate Manager 新憑證時，私有 CA 稽核報告會填入`requestedByServicePrincipal`欄位。`acm.amazonaws.com`這表示 AWS Certificate Manager 服務代表客戶呼叫 AWS 私有 CA API 的`IssueCertificate`動作，以更新憑證。

準備用於稽核報告的 Amazon S3 儲存貯體

Important

AWS Private CA 不支援使用 Amazon S3 物件鎖定。如果您在值區上啟用物件鎖定，就 AWS Private CA 無法將稽核報告寫入值區。

若要存放稽核報告，您需要準備 Amazon S3 儲存貯體。如需詳細資訊，請參閱如何建立 S3 儲存貯體？

您的 S3 儲存貯體必須由連接的許可政策保護。授權的使用者和服務主體需 AWS 私有 CA 要Put允許將物件放置在值區中的權Get限，以及擷取物件的權限。我們建議您套用下面顯示的政策，這會限制對私有 CA 的 AWS 帳戶和 ARN 的存取。如需詳細資訊，請參閱使用 Amazon S3 主控台新增儲存貯體政策。

> **Note**
>
> 在建立稽核報告的主控台程序期間，您可以選擇允許 AWS 私有 CA 建立新值區並套用預設權限原則。預設原則不會對 CA 套用任何SourceArn限制，因此比建議的原則更寬鬆。如果您選擇預設值，稍後可以隨時修改它。

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

```
}
```

建立稽核報告

您可以從主控台或建立稽核報告 AWS CLI。

建立稽核報告 (主控台)

1. 登入您的 AWS 帳戶，然後在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台。
2. 在 [私人憑證授權] 頁面上，從清單中選擇您的私有 CA。
3. 從 Actions (動作) 選單中，選擇 Generate audit report (產生稽核報告)。
4. 在稽核報告目標下，建立新的 S3 儲存貯體？ ，選擇「是」 並輸入唯一的值區名稱，或選擇「否」，然後從清單中選擇現有的時段。

   如果您選擇「是」，則 AWS 私有 CA 會建立預設政策並將其附加至值區。如果您選擇「否」，則必須在儲存貯體中附加策略，然後才能產生稽核報告。使用中所述的策略模式準備用於稽核報告的 Amazon S3 儲存貯體。如需附加政策的相關資訊，請參閱使用 Amazon S3 主控台新增儲存貯體政策
5. 在 [輸出格式] 下，選擇 [JSON] 做為 [JavaScript 物件標記法] 或 [CSV] 做為逗號分隔
6. 選擇 Generate audit report (產生稽核報告)。

建立稽核報告 (AWS CLI)

1. 如果您還沒有要使用的 S3 儲存貯體，請建立一個。
2. 將政策附加到您的值區。使用中所述的策略模式準備用於稽核報告的 Amazon S3 儲存貯體。如需附加政策的相關資訊，請參閱使用 Amazon S3 主控台新增儲存貯體政策
3. 使用 create-certificate-authority-audit-report 命令建立稽核報告，並將其放置在準備好的 S3 儲存貯體中。

```
$ 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
```

擷取稽核報告

若要擷取稽核報告以進行檢查，請使用 Amazon S3 主控台、API、CLI 或開發套件。如需詳細資訊，請參閱 Amazon 簡單儲存服務使用者指南中的下載物件。

加密您的稽核報告

您可以選擇在包含稽核報告的 Amazon S3 儲存貯體上設定加密。 AWS 私有 CA 支援 S3 中資產的兩種加密模式：

- 使用亞馬遜 S3 代管的 AES-256 金鑰進行自動伺服器端加密。
- 使用客戶管理的加密， AWS Key Management Service 並根據您的規格進行 AWS KMS key 配置。

> **Note**
>
> AWS 私有 CA 不支援使用 S3 自動產生的預設 KMS 金鑰。

下列程序說明如何設定每個加密選項。

設定自動加密

完成以下步驟以啟用 S3 伺服器端加密。

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在「時段」表格中，選擇要保留 AWS 私有 CA 資產的值區。
3. 在您的儲存貯體頁面上，選擇屬性索引標籤。
4. 選擇預設加密卡片。
5. 選擇 啟用 。
6. 選擇 Amazon S3 密鑰（ SSE-S3 ）。
7. 選擇 Save Changes (儲存變更)。

設定自訂加密

完成下列步驟以啟用使用自訂金鑰加密。

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在「時段」表格中，選擇要保留 AWS 私有 CA 資產的值區。
3. 在您的儲存貯體頁面上，選擇屬性索引標籤。
4. 選擇預設加密卡片。
5. 選擇 啟用 。
6. 選擇金AWS Key Management Service 鑰 (SSE-KMS)。
7. 選擇「從您的 AWS KMS 金鑰中選擇」或「輸入 AWS KMS key ARN」。
8. 選擇 Save Changes (儲存變更)。
9. (選擇性) 如果您尚未擁有 KMS 金鑰，請使用下列建立金鑰指令建立一個 KMS AWS CLI 金鑰：

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

輸出包含 KMS 金鑰的金鑰識別碼和 Amazon 資源名稱 (ARN)。下面是一個示例輸出：

```
{
    "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. 使用下列步驟，您可以授與 AWS 私有 CA 服務主體使用 KMS 金鑰的權限。根據預設，所有 KMS 金鑰都是私密的；只有資源擁有者可以使用 KMS 金鑰來加密和解密資料。然而，資源擁有者可以授與其他使用者和資源存取 KMS 金鑰的許可。服務主體必須與存放 KMS 金鑰所在的區域相同。

    a. 首先，使`policy.json`用下列get-key-policy命令將 KMS 金鑰的預設原則儲存為：

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

b. 在文字編輯器中開啟 policy.json 檔案。選取下列其中一個原則陳述式，然後將其新增至現有的策略。

如果您的 Amazon S3 儲存貯體金鑰已啟用，請使用下列陳述式：

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

如果您的 Amazon S3 儲存貯體金鑰已停用，請使用下列陳述式：

```
{
   "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. 最後，使用下列put-key-policy命令套用更新的原則：

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

基礎結構安全 AWS Private Certificate Authority

作為託管服務， AWS Private Certificate Authority 受到 AWS 全球網絡安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎結構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的 API 呼叫透 AWS Private CA 過網路進行存取。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

AWS 私有 CA VPC 端端點 ()AWS PrivateLink

您可以在 VPC 之間建立私人連線，並 AWS 私有 CA 透過設定介面 VPC 端點來建立私人連線。接口端點由AWS PrivateLink一種用於私有訪問 AWS 私有 CA API 操作的技術提供支持。 AWS PrivateLink AWS 私有 CA 透過您的 VPC 和 Amazon 網路來路由所有網路流量，避免暴露在開放的網際網路上。每個 VPC 端點皆會由一個或多個具私有 IP 地址彈性網路界面來表示，而該界面位於 VPC 子網路中。

介面 VPC 端點可直接將您的 VPC 連線到 AWS 私有 CA 沒有網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線的情況下。VPC 中的執行個體不需要公有 IP 位址即可與 AWS 私有 CA API 通訊。

若要 AWS 私有 CA 透過 VPC 使用，您必須從 VPC 內的執行個體進行連線。或者，您可以使用 AWS Virtual Private Network (AWS VPN) 或 AWS Direct Connect將私人網路連線到 VPC。如需相關資訊 AWS VPN，請參閱 Amazon VPC 使用者指南中的 VPN 連線。若要取得有關資訊 AWS Direct Connect，請參閱《使用指南》中的AWS Direct Connect 〈建立連接〉。

AWS 私有 CA 不需要使用 AWS PrivateLink，但我們建議您將其作為額外的安全層。如需 AWS PrivateLink 和 VPC 端點的詳細資訊，請參閱透過 AWS PrivateLink存取服務。

AWS 私有 CA VPC 端點的考量

在設定的介面 VPC 端點之前 AWS 私有 CA，請注意下列考量事項：

- AWS 私有 CA 可能不支援某些可用區域中的 VPC 端點。建立 VPC 端點時，請先在管理主控台中檢查支援。不支援的可用性區域會標示為「此可用區域不支援服務」。
- VPC 端點不支援跨區域請求。請確實在計劃發出 AWS 私有 CA API 呼叫的相同區域中建立端點。
- VPC 端點僅支援 Amazon 透過 Amazon 路線 53 提供的 DNS。如果您想要使用自己的 DNS，您可以使用條件式 DNS 轉送。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 DHCP 選項集。
- 連接到 VPC 端點的安全群組，必須允許從 VPC 的私有子網路，透過 443 埠傳入的連線。
- AWS Certificate Manager 不支援 VPC 端點。
- FIPS 端點 (及其區域) 不支援 VPC 端點。

AWS 私有 CA API 目前支援下列各項的 VPC 端點： AWS 區域

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- 亞太區域 (香港)
- 亞太區域 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太區域 (雪梨)

- 亞太區域 (東京)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- 歐洲 (米蘭)
- 以色列 (特拉維夫)
- Middle East (Bahrain)
- 南美洲 (聖保羅)

為下列項目建立 VPC 端點 AWS 私有 CA

您可以使用 VPC 私人 VPC 主控台 https://console.aws.amazon.com/vpc/ 或. AWS 私有 CA AWS Command Line Interface如需詳細資訊，請參閱 Amazon VPC 使用者指南中的建立介面端點程序。AWS 私有 CA 支援在 VPC 內呼叫其所有 API 作業。

如果您已為端點啟用私人 DNS 主機名稱，則預設 AWS 私有 CA 端點現在會解析為您的 VPC 端點。如需預設服務端點的完整清單，請參閱服務端點和配額。

如果您尚未啟用私人 DNS 主機名稱，Amazon VPC 會提供 DNS 端點名稱，您可以使用下列格式：

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

> **Note**
>
> 值##代表支援之區域的 AWS 地區識別碼 AWS 私有 CA，us-east-2例如美國東部 (俄亥俄) 區域。如需的清單 AWS 私有 CA，請參閱 Cer AWS tificate Manager 私人憑證授權單位端點和配額。

如需詳細資訊，請參閱 Amazon AWS 私有 CA VPC 使用者指南中的 VPC 端點 (AWS PrivateLink)。

建立 AWS 私有 CA的 VPC 端點政策

您可以為的 Amazon VPC 端點建立政策，以指 AWS 私有 CA 定下列項目：

- 可執行動作的委託人
- 可執行的動作
- 可在其中執行動作的資源

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的使用 VPC 端點控制服務的存取。

範例 — 用於動作的 VPC 端點原則 AWS 私有 CA

連接至端點時，下列策略會授與所有主參與者對動 AWS 私有 CA 作IssueCertificate、、DescribeCertificateAuthorityGetCertificateGetCertificateAut和ListTags的存取權。每一節中的資源都是私有 CA。第一節會授權使用指定的私有 CA 和憑證範本建立終端實體憑證。如果您不想要控制使用的範本，則不需要 Condition 區段。但是，移除此區段會允許所有委託人建立 CA 憑證及終端實體憑證。

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

AWS Private Certificate Authority的記錄和監控

監控是維持 AWS 解決方案的可靠性、可用性和效能的 AWS Private Certificate Authority 重要組成部分。您應該從 AWS 解決方案的所有部分收集監視資料，以便在發生多點失敗時更輕鬆地偵錯。

下列主題說明可搭配 AWS 私有 CA使用的 AWS 雲端監控工具。

主題

- 支援的 CloudWatch 指標
- 使用 CloudWatch 事件
- 使用 CloudTrail

支援的 CloudWatch 指標

Amazon CloudWatch 是 AWS 資源的監控服務。您可以用 CloudWatch 來收集和追蹤指標、設定警示，以及自動回應 AWS 資源中的變更。 CloudWatch 量度至少會發佈一次。

AWS 私有 CA 支援下列 CloudWatch 指標。

指標	描述
CRLGenerated	已產生憑證撤銷清單 (CRL)。此指標只適用於私有 CA。
MisconfiguredCRLBucket	未正確設定為 CRL 指定的 S3 儲存貯體。請檢查儲存貯體政策。此指標只適用於私有 CA。

指標	描述
Time	發出要求與發行完成 (或失敗) 之間的時間 (以毫秒為單位)。此測量結果僅適用於IssueCertificate作業。
Success	憑證已成功發行。此測量結果僅適用於IssueCertificate作業。
Failure	操作失敗。此測量結果僅適用於IssueCertificate作業。

如需 CloudWatch 測量結果的相關資訊，請參閱下列主題：

- 使用 Amazon CloudWatch 指標
- 創建 Amazon CloudWatch 警報

使用 CloudWatch 事件

您可以使用 Amazon E CloudWatch vents 自動化 AWS 服務，並自動回應系統事件，例如應用程式可用性問題或資源變更。來自 AWS 服務的事件會以近乎即時的方式傳遞至「 CloudWatch 事件」。您可以撰寫簡單的規則來指出您感興趣的事件，以及當事件符合規則時要採取的自動化動作。 CloudWatch 活動至少發佈一次。如需詳細資訊，請參閱建立在 CloudWatch 事件上觸發的事件規則。

CloudWatch 事件被變成使用 Amazon 的行動 EventBridge。使用時 EventBridge，您可以使用事件觸發目標，包括 AWS Lambda 功能、任 AWS Batch 務、Amazon SNS 主題以及許多其他目標。有關更多信息，請參閱什麼是 Amazon EventBridge？

建立私有 CA 時成功或失敗

這些事件由CreateCertificateAuthority操作觸發。

Success (成功)

成功時，操作會傳回新 CA 的 ARN。

```
{
   "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"
   }
}
```

失敗

失敗時，操作會傳回 CA 的 ARN。使用 ARN，您可以調用DescribeCertificateAuthority以確定 CA 的狀態。

```
{
   "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"
   }
}
```

簽發憑證時成功或失敗

這些事件由IssueCertificate操作觸發。

Success (成功)

成功時，操作會傳回 CA 和新憑證的 ARN。

```
{
   "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"
   }
}
```

失敗

失敗時，操作會傳回憑證 ARN 和 CA 的 ARN。使用憑證 ARN，您可以呼叫GetCertificate以檢視失敗的原因。

```
{
   "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"
   }
}
```

撤銷憑證時成功

此事件由RevokeCertificate操作觸發。

如果撤銷失敗或憑證已遭撤銷，則不會傳送任何事件。

Success

成功時，操作會傳回 CA 和撤銷憑證的 ARN。

```
{
   "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"
   }
}
```

生成 CRL 時的成功或失敗

這些事件是由RevokeCertificate作業觸發，這會導致建立憑證撤銷清單 (CRL)。

Success (成功)

成功時，操作會傳回與 CRL 相關聯 CA 的 ARN。

```
{
   "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"
   }
}
```

故障 1 — 由於權限錯誤，CRL 無法儲存到 Amazon S3

如果發生此錯誤，請檢查您的 Amazon S3 儲存貯體許可。

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

故障 2 — 由於內部錯誤，CRL 無法儲存到 Amazon S3

如果發生此錯誤，請重試操作。

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

失敗 3 — AWS 私有 CA 無法建立 CRL

若要疑難排解此錯誤，請檢查您的CloudWatch 指標。

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

建立 CA 稽核報告時成功或失敗

這些事件由CreateCertificateAuthorityAuditReport操作觸發。

Success (成功)

成功時，操作會傳回 CA 的 ARN 和稽核報告的 ID。

```
{
   "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"
   }
}
```

失敗

如果 Amazon S3 儲存貯體 AWS 私有 CA 缺少PUT許可、在儲存貯體上啟用加密或其他原因，稽核報告可能會失敗。

```
{
   "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"
   }
}
```

使用 CloudTrail

您可AWS CloudTrail以使用來記錄由 AWS Private Certificate Authority. 如需詳細資訊，請參閱下列主題。

主題

- 建立政策
- 擷取政策
- 刪除政策
- 建立憑證授權單位
- 生成中心
- 生成 OCS 響應
- 建立稽核報告
- 刪除憑證授權單位
- 還原憑證授權單位
- 描述憑證授權單位
- 擷取憑證授權單位憑證
- 擷取憑證授權單位簽署要求
- 擷取憑證
- 匯入憑證授權單位憑證
- 頒發證書
- 列出憑證授權單
- 列出標籤
- 撤銷憑證
- 標記私有憑證授權單
- 從私人憑證授權單位移除標籤
- 更新憑證授權單位

建立政策

下列 CloudTrail 範例顯示呼叫作PutPolicy業的結果。

```
{
   "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"
}
```


擷取政策

下列 CloudTrail 範例顯示呼叫作GetPolicy業的結果。


```
{
   "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"
}
```

刪除政策

下列 CloudTrail 範例顯示呼叫作DeletePolicy業的結果。


```
{
   "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"
}
```

建立憑證授權單位

下列 CloudTrail 範例顯示呼叫作CreateCertificateAuthority業的結果。

```
{
   "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"
}
```

生成中心

下列 CloudTrail 範例顯示產生 CRL 事件的記錄。

```
{
    "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"
}
```

生成 OCS 響應

下列 CloudTrail 範例顯示產生 OCS PResponse 事件的記錄。

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

建立稽核報告

下列 CloudTrail 範例顯示呼叫作CreateCertificateAuthorityAuditReport業的結果。

```
{
   "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"
}
```

刪除憑證授權單位

下列 CloudTrail 範例顯示呼叫作DeleteCertificateAuthority業的結果。在此範例中，無法刪除憑證授權單位，因為其位於 ACTIVE 狀態。

```
{
   "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"
}
```

還原憑證授權單位

下列 CloudTrail 範例顯示呼叫作RestoreCertificateAuthority業的結果。在此範例中，無法還原憑證授權單位，因為它不是位於 DELETED 狀態。

```
{
   "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"
}
```

描述憑證授權單位

下列 CloudTrail 範例顯示呼叫作DescribeCertificateAuthority業的結果。

```
{
   "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"
}
```

擷取憑證授權單位憑證

下列 CloudTrail 範例顯示呼叫作GetCertificateAuthorityCertificate業的結果。

```
{
   "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"
}
```

擷取憑證授權單位簽署要求

下列 CloudTrail 範例顯示呼叫作GetCertificateAuthorityCsr業的結果。

```
{
   "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"
}
```

擷取憑證

下列 CloudTrail 範例顯示呼叫作GetCertificate業的結果。

```
{
   "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"
}
```

匯入憑證授權單位憑證

下列 CloudTrail 範例顯示呼叫作ImportCertificateAuthorityCertificate業的結果。

```
{
   "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"
}
```

頒發證書

下列 CloudTrail 範例顯示呼叫作IssueCertificate業的結果。

```
{
   "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"
}
```

列出憑證授權單

下列 CloudTrail 範例顯示呼叫作ListCertificateAuthorities業的結果。

```
{
   "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"
}
```

列出標籤

下列 CloudTrail 範例顯示呼叫作ListTags業的結果。

```
{
   "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"
}
```

撤銷憑證

下列 CloudTrail 範例顯示呼叫作RevokeCertificate業的結果。

```
{
   "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"
}
```

標記私有憑證授權單

下列 CloudTrail 範例顯示呼叫作TagCertificateAuthority業的結果。

```
{
   "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"
}
```

從私人憑證授權單位移除標籤

下列 CloudTrail 範例顯示呼叫作UntagCertificateAuthority業的結果。

```
{
   "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"
}
```

更新憑證授權單位

下列 CloudTrail 範例顯示呼叫作UpdateCertificateAuthority業的結果。

```
{
   "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"
}
```

規劃您的 AWS 私有 CA 部署

AWS 私有 CA 為您提供對組織私有 PKI (公開金鑰基礎結構) 的完整雲端控制，從根憑證授權單位 (CA) 延伸至從屬 CA 延伸到最終實體憑證。對於安全、可維護、可擴展且符合組織需求的 PKI 而言，詳盡的規劃非常重要。本節提供設計 CA 階層、管理私有 CA 和私有終端實體憑證生命週期，及套用安全最佳實務的指導。

本節說明如何在建立私有憑證授權單位 (CA) 之前準備 AWS 私有 CA 使用。它也說明透過線上憑證狀態通訊協定 (OCSP) 或憑證撤銷清單 (CRL) 新增撤銷支援的選項。

此外，您應該判斷組織是否偏好在內部部署 (而非使 AWS用) 託管其私人根 CA 認證。在這種情況下，您需要在使用之前設置並保護自我管理的私有 PKI。 AWS 私有 CA在這個案例中，您可以在以外的 AWS 私有 CA父項 CA AWS 私有 CA 支援建立從屬 CA。如需詳細資訊，請參閱安裝由外部父 CA 簽署的從屬 CA 憑證。

主題

- 設置您的 AWS 帳戶和 AWS CLI
- 設計 CA 階層
- 管理私有 CA 生命週期
- 設定憑證撤銷方法
- 憑證機構模式
- 規劃復原力

設置您的 AWS 帳戶和 AWS CLI

如果您還不是 Amazon Web Services（AWS）客戶，則必須註冊才能使用 AWS 私有 CA。您的帳戶會自動具備存取所有可用服務的權限，但是您只需要針對所使用的服務付費。

> **Note**
>
> AWS 私有 CA AWS 免費方案不提供。

主題

- 註冊一個 AWS 帳戶

- 建立具有管理權限的使用者
- 安裝 AWS Command Line Interface

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊時 AWS 帳戶，會建立AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM 身分中心使用者登入的說明，請參閱使用AWS 登入 者指南中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

安裝 AWS Command Line Interface

如果您尚未安裝， AWS CLI 但想要使用它，請遵循中的指示AWS Command Line Interface。在本指南中，我們假設您已經設定了端點、Region 和驗證詳細資料，而我們會從範例命令中省略這些參數。

設計 CA 階層

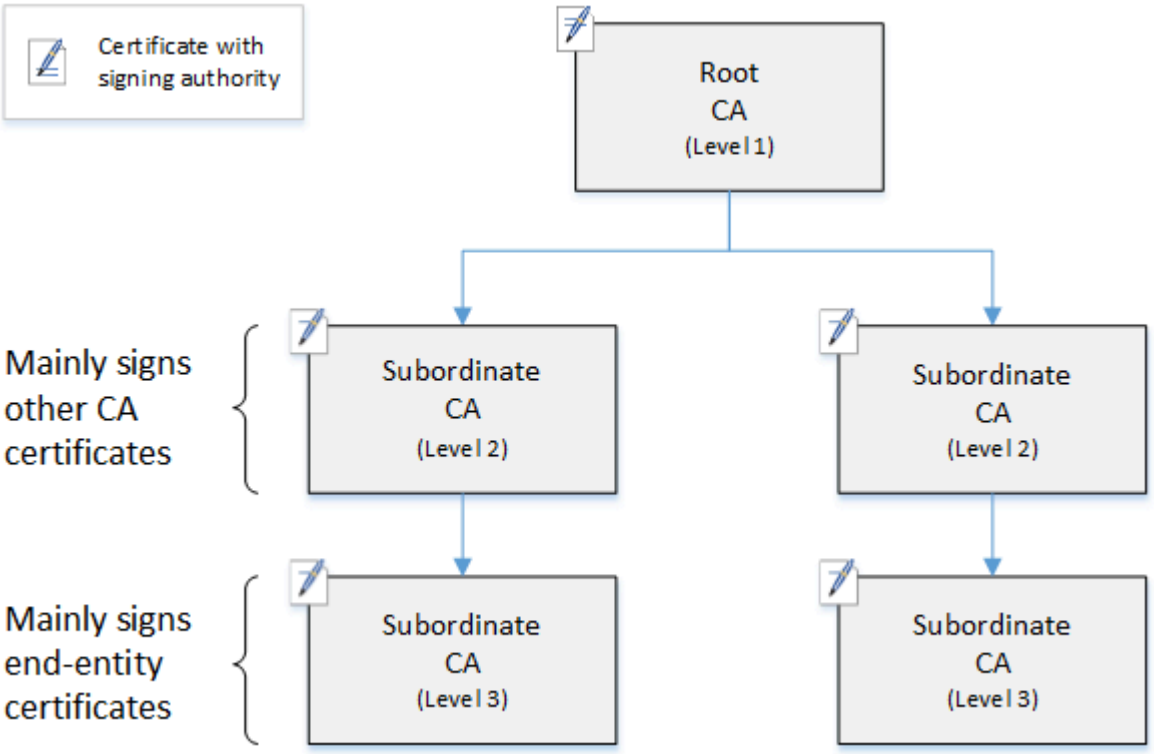
使用時 AWS 私有 CA，您可以建立最多五個層級的憑證授權單位階層。位於階層樹狀結構頂端的根 CA，可以有任何數量的分支。根 CA 在每個分支上最多可以有四個層級的次級 CA。您也可以建立多個階層，每個階層都具有自己的根。

設計完善的 CA 階層具有下列優點：

- 適用於每個 CA 的精細安全控制

- 具有更佳負載平衡與安全的管理任務分工
- 使用有限且可撤銷信任的 CA 進行日常操作
- 有效期間和憑證路徑限制

下圖說明簡單的三層 CA 階層。



樹狀結構中的每個 CA 都由具有簽署授權單位的 X.509 v3 憑證進行支援 (以圖示表 pen-and-paper 示)。這表示其做為 CA 可以簽署其他從屬於其下方的憑證。當 CA 簽署較低層級的 CA 憑證時，會對已簽署的憑證授予有限且可撤銷的授權。層級 1 中的根 CA，會簽署層級 2 中的高階次級 CA 憑證。這些 CA 會依次簽署層級 3 中 CA 的憑證，而這些 CA 則是由管理終端實體憑證的 PKI (公有金鑰基礎設施) 管理員所使用。

CA 階層中的安全，應在樹狀目錄的頂端設為最強。此設定可保護根 CA 憑證及其私有金鑰。根 CA 會錨定所有次級 CA 及其下方終端實體憑證的信任。雖然對終端實體憑證造成的危害會導致局部損害，但對根造成的危害會破壞整個 PKI 中的信任。根和高階從屬 CA 只會很少使用 (通常用來簽署其他 CA 憑證)。因此，這些 CA 會受到嚴格控制與稽核，以確保降低危害風險。階層中的較低層級，對安全方面的限制則較少。這種方法可讓您執行例行管理任務，例如發行及撤銷使用者、電腦主機和軟體服務的終端實體憑證。

> **Note**
>
> 使用根 CA 簽署次級憑證是罕見事件，只發生在少數情況下：
>
> - 建立 PKI 時
> - 需要更換高階憑證授權機構時
> - 需要設定憑證撤銷清單 (CRL) 或線上憑證狀態通訊協定 (OCSP) 回應程式時
>
> 根和其他高階 CA，需要高度安全的操作程序和存取控制通訊協定。

主題

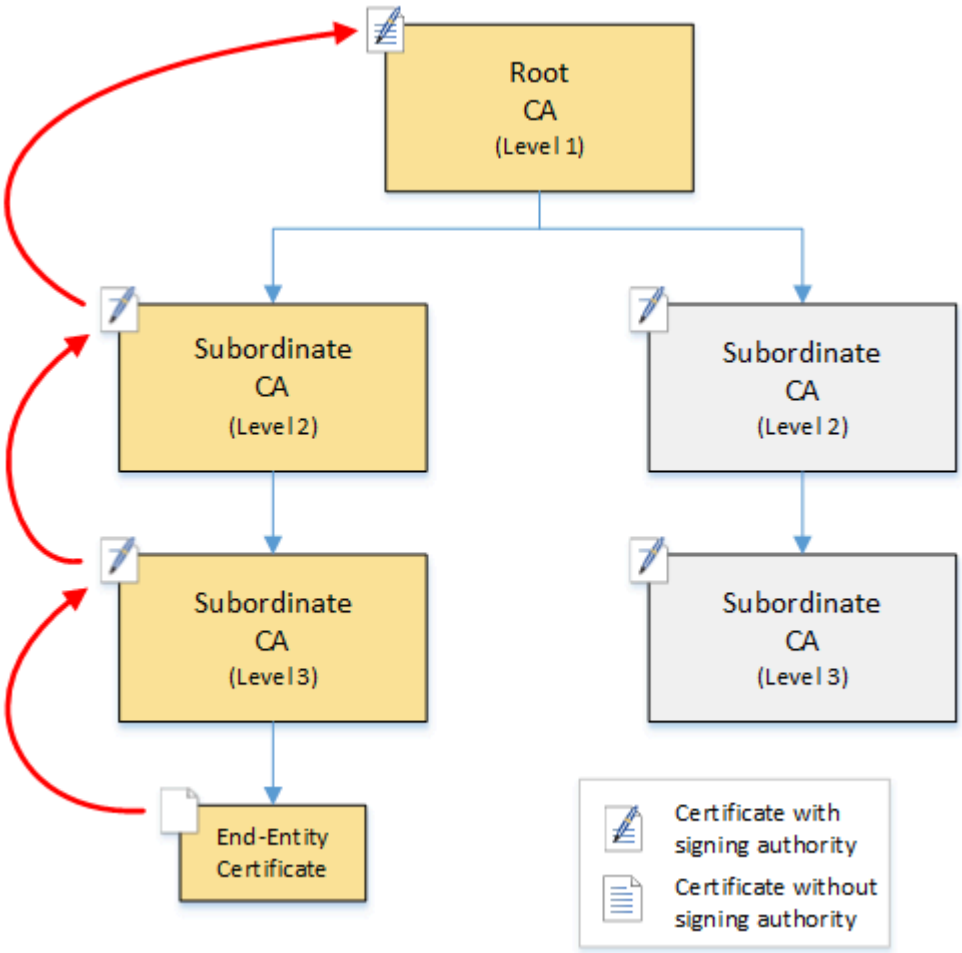
- 驗證最終實體憑證
- 規劃 CA 階層的結構
- 在憑證路徑上設定長度限制

驗證最終實體憑證

終端實體憑證會從透過次級 CA 回到根 CA 的認證路徑取得信任。當 Web 瀏覽器或其他用戶端提供終端實體憑證時，其會嘗試建構信任鏈。例如，其可能會檢查憑證的「發行者辨別名稱」和「主體辨別名稱」是否與發行 CA 憑證的對應欄位相符。接著會繼續在階層的每個連續層級往上繼續比對，直到用戶端到達其信任存放區中所包含的信任根目錄為止。

信任存放區是瀏覽器或作業系統所包含的信任 CA 資源庫。針對私有 PKI，您組織的 IT 必須確保每個瀏覽器或系統先前已將私有根 CA 新增至其信任存放區。否則會無法驗證認證路徑，而導致用戶端錯誤。

下圖顯示在提供終端實體 X.509 憑證時，瀏覽器會遵循的驗證路徑。請注意，終端實體憑證缺少簽署授權機構，並且僅可用於驗證擁有該憑證的實體。



瀏覽器檢查終端實體憑證。瀏覽器發現憑證提供次級 CA (層級 3) 的簽章做為其信任登入資料。次級 CA 的憑證必須包含在相同 PEM 檔案中。或者，也可以位於包含組成信任鏈結之憑證的個別檔案中。找到這些資訊後，瀏覽器檢查次級 CA (層級 3) 的憑證，並發現其提供次級 CA (層級 2) 的簽章。反過來，次級 CA (層級 2) 會提供來自根 CA (層級 1) 的簽章做為其信任登入資料。如果瀏覽器發現預先安裝在其信任存放區中的私有根 CA 憑證複本，則會將終端實體憑證驗證為受信任。

一般來說，瀏覽器也會根據憑證撤銷清單 (CRL) 來檢查每個憑證。已過期、已撤銷或設定錯誤的憑證會遭拒絕，且驗證會失敗。

規劃 CA 階層的結構

一般而言，您的 CA 階層應該要反映出組織的結構。瞄準路徑長度 (也就是 CA 層級數目)，不得超過委派系統管理和資訊安全角色所需的路徑長度。將 CA 新增至階層，表示增加認證路徑中的憑證數量，而這會增加驗證時間。將路徑長度保持在最小值還可以減少驗證最終實體憑證時從伺服器傳送至用戶端的憑證數目。

理論上，沒有pathLenConstraint參數的根 CA 可以授權無限層級的從屬 CA。從屬 CA 可以擁有其內部組態所允許的子系從屬 CA 數量。 AWS 私有 CA 受管理階層支援最多五個層級的 CA 憑證路徑。

設計完善的 CA 結構有幾項優點：

- 不同組織單位的個別管理控制
- 將存取權委派給次級 CA 的能力
- 可透過額外安全控制來保護較高層級 CA 的階層式結構

有兩種常見的 CA 結構可以達成上述所有項目：

- 兩個 CA 層級：根 CA 和次級 CA

  這是最簡單的 CA 結構，可允許根 CA 和次級 CA 的個別系統管理、控制和安全政策。您可以在為根 CA 維護嚴格控制和政策的同時，給予次級 CA 較寬鬆的存取權。後者用於大量發行終端實體憑證。

- 三個 CA 層級：根 CA 和兩個次級 CA 層級

  與上述情況類似，此結構會增加一層額外的 CA，以進一步將根 CA 與低層 CA 操作分開。中間層 CA 僅用於簽署執行終端實體憑證發行的次級 CA。

較不常見的 CA 結構包括下列項目：

- 四個或更多 CA 層級

  雖然比三層階層較少見，但具有四層或更多層級的 CA 階層仍可能存在，且可能需要允許管理委派。

- 一個 CA 層級：僅限根 CA

  此結構通常用於進行開發及測試，而不需要完整的信任鏈時。在生產環境中使用並不常見。此外，其違反了針對根 CA 及發行終端實體憑證的 CA 維護個別安全政策的最佳實務。

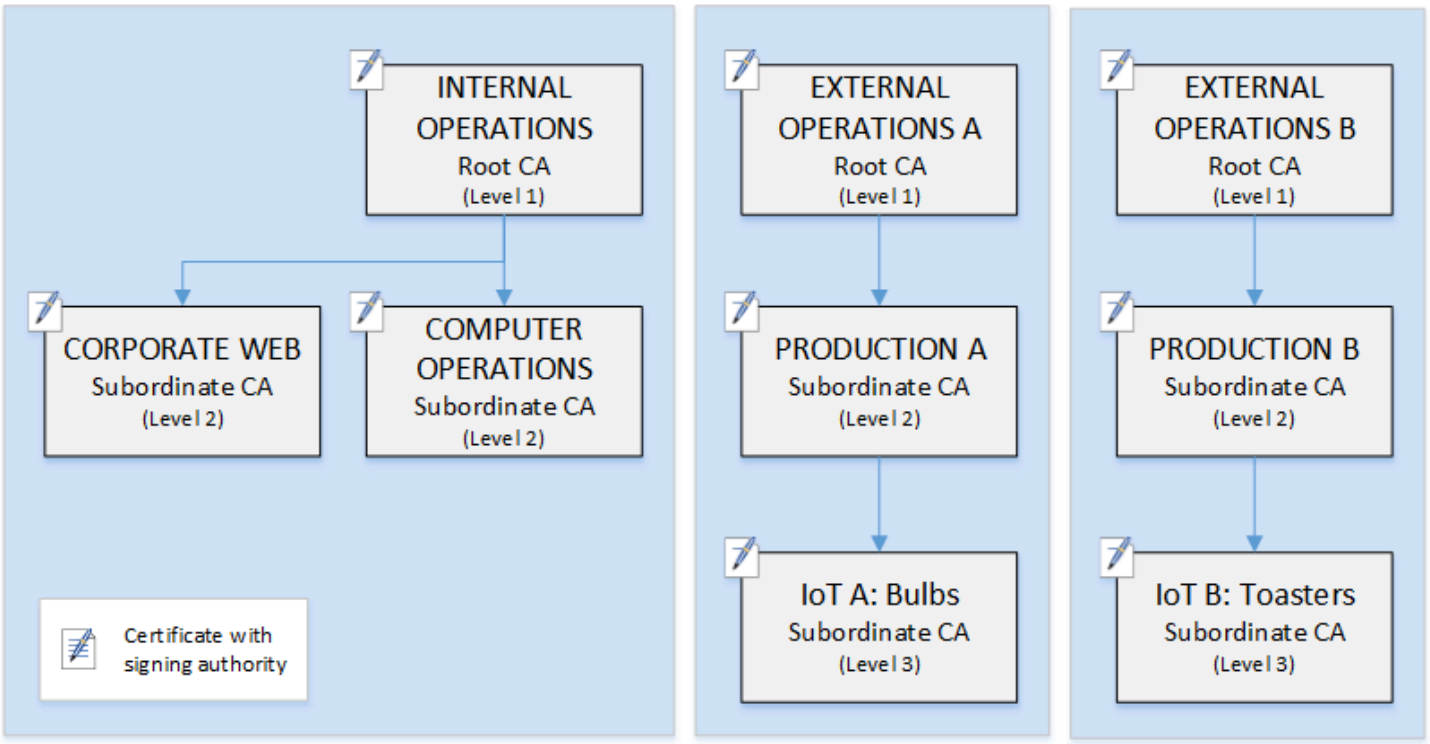
  不過，如果您已經直接從根 CA 發行憑證，則可以移轉至 AWS 私有 CA. 與使用透過 OpenSSL 或其他軟體管理的根 CA 相比，這樣做可提供安全性和控制優勢。

製造商的私人 PKI 範例

在此範例中，一家虛構的科技公司生產兩種物聯網產品：智慧型燈泡和智慧型烤麵包機。在生產過程中，會將終端實體憑證發行給每部裝置，以便裝置能夠透過網際網路與製造商安全通訊。公司 PKI 也

會確保其電腦基礎設施的安全，包括內部網站和各種自我裝載電腦服務，這些服務負責執行財務和業務營運。

因此，CA 階層會密切為業務的這些管理和操作層面建立模型。



此階層包含三個根目錄，一個用於內部操作，另外兩個用於外部操作 (每個產品系列都有一個根 CA)。它也說明了多個憑證路徑長度，其中包含兩個層級的 CA 供內部作業使用，以及三個層級的「外部作業」。

在外部作業端使用分離的根 CA 和其他從屬 CA 層是一項服務業務和安全性需求的設計決策。具備多個 CA 樹狀結構時，PKI 就可以適應未來的企業重新組織、撤資或併購。在發生變更時，整個根 CA 階層都可以隨著其保護的部門而整潔地移動。由於具備了兩個層級的次級 CA，根 CA 擁有高度隔離的效果，可與負責大量簽署數千或數百萬個製造項目之憑證的層級 3 CA 區隔。

在內部方面，企業 Web 和內部電腦操作可完成兩個層級的階層。這些層級可讓 Web 管理員和操作工程師獨立管理其工作領域的憑證發行。將 PKI 區分為不同的功能領域是安全最佳實務，可保護每個領域免受可能影響另一個網域的危害。Web 管理員會發行終端實體憑證供整個公司的 Web 瀏覽器使用，以驗證及加密內部網站上的通訊。操作工程師會發行終端實體憑證，以互相驗證資料中心主機和電腦服務。此系統會藉由在 LAN 上加密資料以協助保護敏感資料的安全。

在憑證路徑上設定長度限制

CA 階層架構的結構，由每個憑證所包含的「基本限制條件延伸」來定義及強制執行。延伸會定義兩個限制條件：

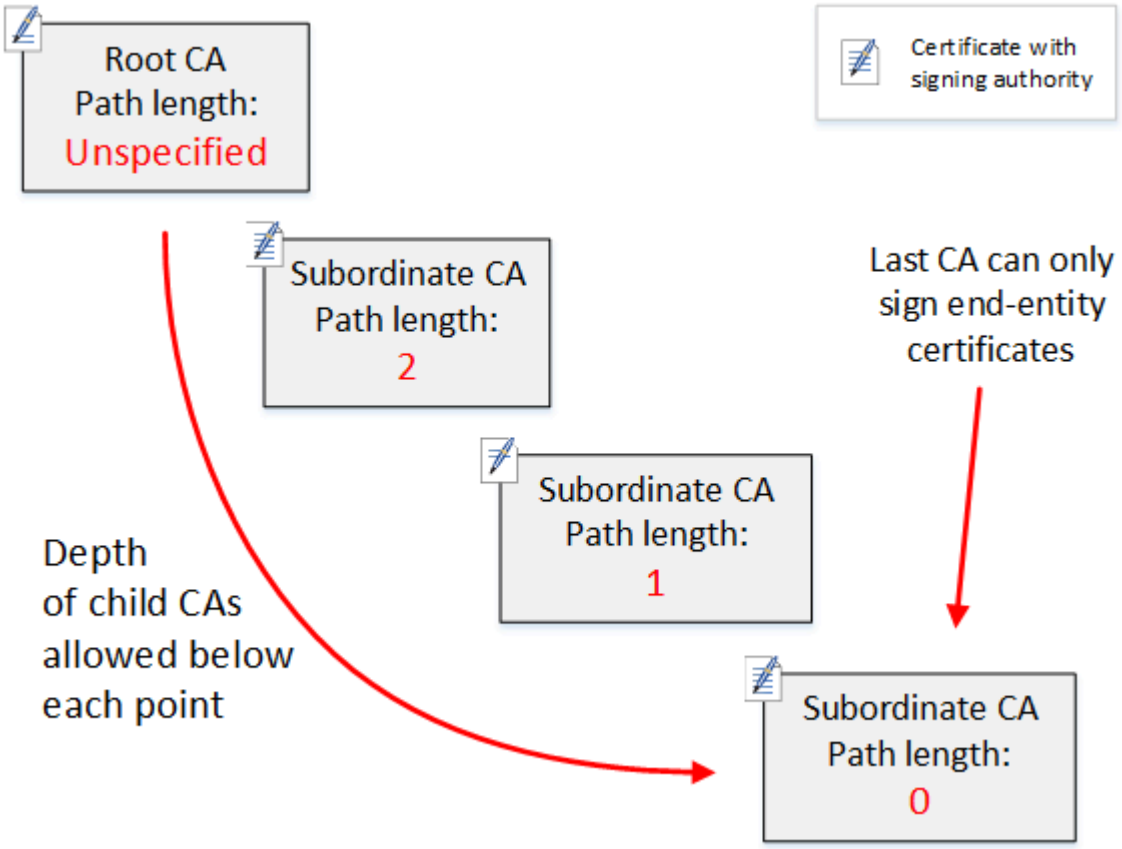
- `cA`— 憑證是否定義 CA。如果此值為 False (預設值)，則憑證為終端實體憑證。
- `pathLenConstraint`— 可存在於有效信任鏈中的較低層級從屬 CA 的最大數目。終端實體憑證不會計入，因為它不是 CA 憑證。

根 CA 憑證需要具備最大的彈性，且不包含路徑長度限制條件。這允許根定義任何長度的認證路徑。

> **Note**
>
> AWS 私有 CA 將認證路徑限制為五個層級。

次級 CA 的 `pathLenConstraint` 值會等於或大於零，取決於階層放置中的位置與所需功能。例如，在具有三個 CA 的階層中，不會對根 CA 指定任何路徑限制條件。第一個次級 CA 的路徑長度為 1，因此可以簽署子 CA。每個子 CA 都必須具有值為零的 `pathLenConstraint`。這表示其可以簽署終端實體憑證，但無法發行其他 CA 憑證。限制建立新 CA 的能力，是一項重要的安全控制。

下圖說明了階層中受限授權能力的這種傳播方式。



在四層階層中，根目錄是不受限制的 (如同往常)。但第一個次級 CA 的 `pathLenConstraint` 值為 2，這會限制其子 CA 深入超過兩個層級。因此，針對有效的認證路徑，限制條件值必須在接下來的兩個層級遞減為零。如果 Web 瀏覽器遇到來自此分支的終端實體憑證，且路徑長度大於 4，則驗證就會失敗。這類憑證可能是意外建立的 CA、設定錯誤的 CA 或未經授權的發行所造成。

使用範本管理路徑長度

AWS 私有 CA 提供用於發行根憑證、從屬憑證和終端實體憑證的範本。這些範本納入了基本限制條件值 (包括路徑長度) 的最佳實務。範本包括下列項目：

- RootCACertificate/V1
- 次級PathLen緩存證書 _ 0/V1
- 次級PathLen緩存證書 _ 1/V1
- 次級PathLen緩存證書 _ 2/V1
- 下屬緩PathLen存證書 _ 3/V1
- EndEntityCertificate/1

如果您嘗試建立路徑長度大於或等於其發行 CA 憑證路徑長度的 CA，則 `IssueCertificate` API 將傳回錯誤。

如需憑證範本的詳細資訊，請參閱瞭解憑證範本。

使用自動化 CA 階層設定 AWS CloudFormation

當您完成 CA 階層的設計時，您可以對其進行測試，並使用 AWS CloudFormation 範本將其投入生產環境。如需此類範本的範例，請參閱《AWS CloudFormation 使用指南》中的〈宣告私有 CA 階層〉。

管理私有 CA 生命週期

CA 憑證有固定的生命週期或有效期間。當 CA 憑證過期時，在 CA 階層中由次級 CA 直接或間接發行的所有憑證都會失效。您可以預先進行規劃，以避免 CA 憑證過期。

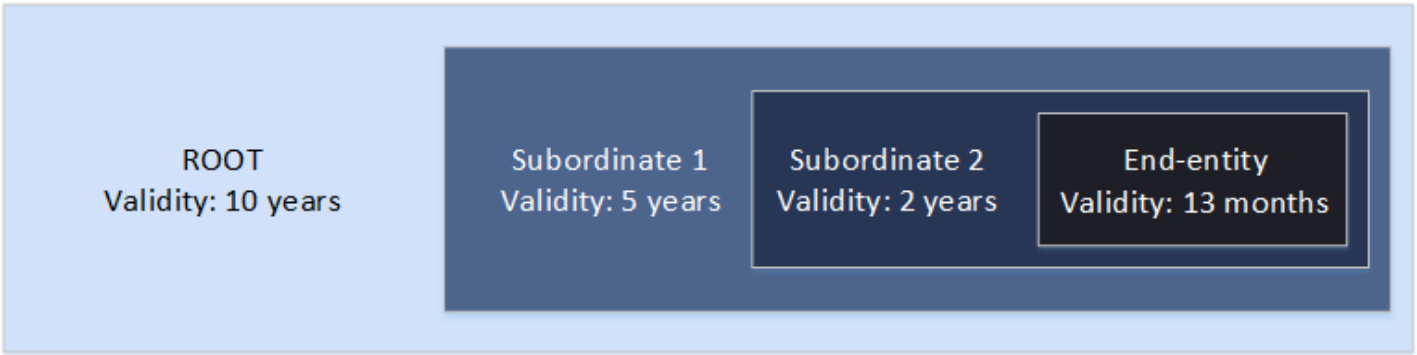
選擇有效期

X.509 憑證的有效期間，是必要的基本憑證欄位。其會決定發行 CA 認證憑證可以受信任的時間範圍 (除了撤銷之外)。(根憑證是自我簽署的，會認證其本身的有效期間。)

AWS 私有 CA 並 AWS Certificate Manager 協助設定受到下列限制的憑證有效期限：

- 由管理的憑證的有效期 AWS 私有 CA 必須小於或等於發行憑證之 CA 的有效期。換句話說，子 CA 和終端實體憑證的有效期間不能超過其父憑證。嘗試使用 `IssueCertificate` API 來發行有效期間長於或等於父 CA 的 CA 憑證會失敗。
- 由 AWS Certificate Manager (ACM 產生私密金鑰的憑證) 所發行及管理的憑證有效期為 13 個月 (395 天)。ACM 會管理這些憑證的更新程序。如果您使用直 AWS 私有 CA 接發行憑證，您可以選擇任何有效期。

下圖顯示巢狀有效期間的一般組態。根憑證的有效期間最長，終端實體憑證相對較短，而次級 CA 的有效期間範圍則位於這兩者之間。



在您規劃 CA 階層時，請判斷 CA 憑證的最佳生命週期。從想要發行之終端實體憑證的所需生命週期向後推算。

終端實體憑證

終端實體憑證應具有適用於使用案例的有效期。短暫的生命週期，可在憑證私有金鑰遺失或遭竊時，盡量減少洩漏憑證的機會。然而，短暫的生命周期也代表必須頻繁續約。若無法續約即將到期的憑證，就可能會導致停機。

如果存在安全性漏洞，以分發方式使用終端實體憑證也可能會造成後勤問題。您的規劃應該考量更新和分配憑證、撤銷遭危害的憑證，以及撤銷傳播至依賴憑證之用戶端的速度。

透過 ACM 發行的終端實體憑證的預設有效期為 13 個月 (395 天)。在中 AWS 私有 CA，您可以使用 `IssueCertificate` API 套用任何有效期間，只要該期間小於發行 CA 的有效期間。

次級 CA 憑證

次級 CA 憑證的有效期間，應該比其發行的憑證要長得多。CA 憑證有效性的良好範圍，是其發行之任何子 CA 憑證或終端實體憑證的二到五倍。例如，假設您有兩個層級的 CA 階層 (根 CA 和一個次級 CA)。如果您想要發行生命週期為一年的終端實體憑證，您可以將次級發行 CA 生命週期設為三年。這是中附屬 CA 憑證的預設有效期間。 AWS 私有 CA次級 CA 憑證可以在不取代根 CA 憑證的情況下進行變更。

根憑證

根 CA 憑證的變更會影響整個 PKI (公有金鑰基礎設施)，並要求您更新所有相依的用戶端作業系統和瀏覽器信任存放區。若要將操作影響降至最低，您應該為根憑證選擇較長的有效期間。根憑證的 AWS 私有 CA 預設值為十年。

管理 CA 繼任

您有兩種方式可以管理 CA 繼承：取代舊的 CA，或使用新的有效期間重新發行 CA。

取代舊的 CA

若要取代舊 CA，您可以建立新的 CA，並將其鏈結至相同的父 CA。之後，您就可以從新的 CA 發出憑證。

從新 CA 發行的憑證，具有新 CA 鏈結。建立新的 CA 後，您就可以停用舊 CA，以防止其發行新的憑證。停用時，舊 CA 支援撤銷從 CA 發行的舊憑證，如果設定這樣做，它會繼續透過 OCSP 和/或憑證撤銷清單 (CRL) 驗證憑證。從舊 CA 發出的最後一個憑證到期時，您就可以刪除舊的 CA。您可以為 CA 發行的所有憑證產生稽核報告，以確認所有已發行的憑證都已過期。如果舊 CA 具有次級 CA，您

也必須取代這些次級 CA，因為次級 CA 會與其父 CA 同時或在之前過期。首先，請取代階層中需要取代的最高層級 CA。然後在每個後續的較低層級中，建立替換用的新次級 CA。

AWS 建議您視需要在 CA 的名稱中加入 CA 產生識別碼。例如，假設您將第一代 CA 命名為「公司根 CA」。當您建立第二代 CA 時，請將其命名為「公司根 CA G2」。這種簡單的命名慣例，有助於避免在兩個 CA 都未過期時產生混淆。

這是較佳的 CA 繼承方法，因為會輪替 CA 的私有金鑰。輪替私有金鑰是 CA 金鑰的最佳實務。輪替的頻率應該與金鑰使用頻率成正比：發行較多憑證的 CA，應該更頻繁地進行輪替。

> **Note**
>
> 如果您取代 CA，則無法更新透過 ACM 發行的私有憑證。如果您使用 ACM 進行發行和續約，則必須重新發行 CA 憑證，以延長 CA 的存留期。

重新發行舊 CA

當 CA 即將到期時，延長其壽命的另一種方法是以新的到期日重新發行 CA 憑證。重新發行會保留所有 CA 中繼資料，並保留現有的私密金鑰和公開金鑰。在這個案例中，CA 所發行的現有憑證鏈結和未過期的終端實體憑證會保持有效，直到到期為止。新的憑證發行也可以繼續不中斷。若要使用重新發行的憑證更新 CA，請遵循中建立和安裝 CA 憑證所述的一般安裝程序。

> **Note**
>
> 我們建議您更換即將到期的 CA，而不是重新發行憑證，因為輪換為新 key pair 所帶來的安全性優勢。

撤銷 CA

您可藉由撤銷 CA 的基礎憑證來撤銷 CA。這也有效地撤銷 CA 頒發的所有證書。撤銷資訊是透過 OCSP 或 CRL 發佈給用戶端。只有當您想要撤銷 CA 憑證的所有已核發最終實體和下屬 CA 憑證時，才應撤銷 CA 憑證。

設定憑證撤銷方法

當您規劃私有 PKI 時 AWS 私有 CA，您應該考慮如何處理您不再希望端點信任已發行憑證的情況，例如端點的私密金鑰暴露時。這個問題的常見方法是使用短期憑證或設定憑證撤銷。短期憑證會在這麼短

的時間內過期 (以小時或數天為單位)，撤銷是沒有意義的，因為憑證在通知終端節點撤銷所花費的時間大約相同的時間內失效。本節說明 AWS 私有 CA 客戶的撤銷選項，包括組態和最佳做法。

尋找撤銷方法的客戶可以選擇「線上憑證狀態通訊協定」(OCSP)、憑證撤銷清單 (CRL)，或兩者皆選。

> **Note**
>
> 如果您在未設定撤銷的情況下建立 CA，您隨時可以稍後進行設定。如需詳細資訊，請參閱 更新您的私有 CA。

- 線上憑證狀態通訊協定 (OCSP)

  AWS 私有 CA 提供完全受控的 OCSP 解決方案，可通知端點憑證已撤銷，而不需要客戶自行操作基礎結構。客戶可以使用 AWS 私有 CA 主控台、API、CLI 或透 AWS CloudFormation過單一作業，在新的或現有的 CA 上啟用 OCSP。CRL 會在端點上儲存和處理，而且可能會過時，OCSP 儲存和處理需求會在回應程式後端同步處理。

  當您針對 CA 啟用 OCSP 時，會在每個發行的新憑證的授權單位資訊存取 (AIA) 延伸中 AWS 私有 CA 包含 OCSP 回應者的 URL。此擴充功能可讓用戶端 (例如網頁瀏覽器) 查詢回應者，並判斷最終實體或下屬 CA 憑證是否可信任。回應者會傳回已加密簽署的狀態訊息，以確保其真實性。

  AWS 私有 CA OCSP 回應程式符合 R FC 5019 規範。

  OCSP 考量

  - OCSP 狀態訊息會使用與發行 CA 設定使用的相同簽署演算法來簽署。根據預設，在 AWS 私有 CA 主控台中建立的 CA 會使用 SHA256WITHRSA 簽章演算法。其他支援的演算法可以在 CertificateAuthorityConfigurationAPI 文件中找到。
  - 如果已啟用 O@@ CSP 回應程式，APIPPASS 和企業社會責任傳遞憑證範本將無法與 AIA 擴充套件搭配使用。
  - 受管理 OCSP 服務的端點可在公用網際網路上存取。想要 OCSP 但不想擁有公用端點的客戶將需要操作自己的 OCSP 基礎結構。
- 憑證撤銷清單 (CRL)

  CRL 包含已撤銷的憑證清單。當您設定 CA 以產生 CRL 時，請在每個發行的新憑證中 AWS 私有 CA 包含 CRL 發佈點延伸模組。此擴充功能會提供 CRL 的 URL。此擴充功能可讓用戶端 (例如網頁瀏覽器) 查詢 CRL，並判斷最終實體或附屬 CA 憑證是否可信任。

因為用戶端必須下載 CRL 並在本機處理它們，因此它們的使用比 OCSP 更耗用記憶體。CRL 可能會消耗較少的網路頻寬，因為 CRL 清單已下載並快取，而 OCSP 會檢查每次新連線嘗試的撤銷狀態。

> **Note**
>
> OCSP 和 CRL 都會在撤銷和狀態變更的可用性之間出現一些延遲。
>
> - 當您撤銷憑證時，OCSP 回應最多可能需要 60 分鐘才能反映新狀態。一般而言，OCSP 傾向於支援更快速的撤銷資訊散發，因為與用戶端可快取數天的 CRL 不同，OCSP 回應通常不會由用戶端快取。
> - CRL 通常在憑證撤銷後約 30 分鐘更新。如果 CRL 更新因任何原因失敗，則每 15 分鐘 AWS 私有 CA 會進一步嘗試一次。

撤銷組態的一般需求

下列需求適用於所有撤銷組態。

- 停用 CRL 或 OCSP 的組態必須只包含 `Enabled=False` 參數，如果包含其他參數 (例如 `CustomCname` 或 `ExpirationInDays`)，則會失敗。
- 在 CRL 組態中，`S3BucketName`參數必須符合 Amazon 簡單儲存服務儲存貯體命名規則。
- 包含 CRL 或 OCSP 之自訂標準名稱 (CNAME) 參數的模型組態必須符合 RFC7230 限制，在 CNAME 中使用特殊字元。
- 在 CRL 或 OCSP 組態中，CNAME 參數的值不得包含通訊協定字首，例如 "http://" 或 "https://"。

主題

- 規劃憑證撤銷清單 (CRL)
- 設定 AWS 私有 CA OCSP 的自訂網址

規劃憑證撤銷清單 (CRL)

在您可以將 CRL 設定為 CA 建立程序的一部分之前，可能需要先進行一些設定。本節說明在建立附加 CRL 的 CA 之前，您應該瞭解的先決條件和選項。

如需有關使用線上憑證狀態通訊協定 (OCSP) 作為 CRL 的替代方案或補充的資訊，請參閱憑證撤銷選項和。設定 AWS 私有 CA OCSP 的自訂網址

主題

- CRL 結構
- Amazon S3 中 CRL 的訪問政策
- 啟用 S3 封鎖公用存取 (BPA) CloudFront
- 加密您的 CRL
- 決定 CRL 發佈點 (CDP) URI

CRL 結構

每個 CRL 皆為 DER 編碼檔案。若要下載檔案並使用 OpenSSL 來檢視檔案，請使用類似下列的命令：

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

CRL 具有下列格式：

```
Certificate Revocation List (CRL):
          Version 2 (0x1)
      Signature Algorithm: sha256WithRSAEncryption
          Issuer: /C=US/ST=WA/L=Seattle/O=Example Company CA/OU=Corporate/
CN=www.example.com
          Last Update: Feb 26 19:28:25 2018 GMT
          Next Update: Feb 26 20:28:25 2019 GMT
          CRL extensions:
              X509v3 Authority Key Identifier:
                  keyid:AA:6E:C1:8A:EC:2F:8F:21:BC:BE:80:3D:C5:65:93:79:99:E7:71:65

              X509v3 CRL Number:
                  1519676905984
  Revoked Certificates:
      Serial Number: E8CBD2BEDB122329F97706BCFEC990F8
          Revocation Date: Feb 26 20:00:36 2018 GMT
          CRL entry extensions:
              X509v3 CRL Reason Code:
                  Key Compromise
      Serial Number: F7D7A3FD88B82C6776483467BBF0B38C
          Revocation Date: Jan 30 21:21:31 2018 GMT
          CRL entry extensions:
              X509v3 CRL Reason Code:
                  Key Compromise
```

```
    Signature Algorithm: sha256WithRSAEncryption
         82:9a:40:76:86:a5:f5:4e:1e:43:e2:ea:83:ac:89:07:49:bf:
         c2:fd:45:7d:15:d0:76:fe:64:ce:7b:3d:bb:4c:a0:6c:4b:4f:
         9e:1d:27:f8:69:5e:d1:93:5b:95:da:78:50:6d:a8:59:bb:6f:
         49:9b:04:fa:38:f2:fc:4c:0d:97:ac:02:51:26:7d:3e:fe:a6:
         c6:83:34:b4:84:0b:5d:b1:c4:25:2f:66:0a:2e:30:f6:52:88:
         e8:d2:05:78:84:09:01:e8:9d:c2:9e:b5:83:bd:8a:3a:e4:94:
         62:ed:92:e0:be:ea:d2:59:5b:c7:c3:61:35:dc:a9:98:9d:80:
         1c:2a:f7:23:9b:fe:ad:6f:16:7e:22:09:9a:79:8f:44:69:89:
         2a:78:ae:92:a4:32:46:8d:76:ee:68:25:63:5c:bd:41:a5:5a:
         57:18:d7:71:35:85:5c:cd:20:28:c6:d5:59:88:47:c9:36:44:
         53:55:28:4d:6b:f8:6a:00:eb:b4:62:de:15:56:c8:9c:45:d7:
         83:83:07:21:84:b4:eb:0b:23:f2:61:dd:95:03:02:df:0d:0f:
         97:32:e0:9d:38:de:7c:15:e4:36:66:7a:18:da:ce:a3:34:94:
         58:a6:5d:5c:04:90:35:f1:8b:55:a9:3c:dd:72:a2:d7:5f:73:
         5a:2c:88:85
```

> **Note**
>
> CRL 只有在發出指向它的憑證之後，才會存放在 Amazon S3 中。在此之前，Amazon S3 儲存貯體中只會顯示一個`acm-pca-permission-test-key`檔案。

Amazon S3 中 CRL 的訪問政策

如果您打算建立 CRL，則需要準備一個 Amazon S3 儲存貯體以存放該儲存貯體。 AWS 私有 CA 自動將 CRL 存入您指定的 Amazon S3 儲存貯體，並定期更新該儲存貯體。如需詳細資訊，請參閱建立儲存貯體。

您的 S3 儲存貯體必須由附加的 IAM 許可政策保護。授權的使用者和服務主體需 AWS 私有 CA 要`Put`允許將物件放置在值區中的權`Get`限，以及擷取物件的權限。在建立 CA 的主控台程序期間，您可以選擇允許 AWS 私有 CA 建立新值區並套用預設權限原則。

> **Note**
>
> IAM 政策組態取決於 AWS 區域 涉及的項目。區域分為兩類：
>
> - 預設啟用的區域 — 預設為所有人啟用的區域。 AWS 帳戶
> - 預設停用區域 — 依預設為停用，但客戶可以手動啟用的區域。

如需詳細資訊與預設停用區域的清單，請參閱管理。 AWS 區域如需 IAM 內容中服務主體的討論，請參閱選擇加入區域中的AWS 服務主體。
當您將 CRL 設定為憑證撤銷方法時， AWS 私有 CA 會建立 CRL 並將其發佈到 S3 儲存貯體。S3 儲存貯體需要允許 AWS 私有 CA 服務主體寫入儲存貯體的 IAM 政策。服務主體的名稱會根據使用的區域而有所不同，並且並非支援所有可能性。

PCA	S3	服務主體
兩者都在同一地區		acm-pca.amazonaws.com
已啟用	已啟用	acm-pca.amazonaws.com
已停用	已啟用	acm-pca.*Region*.amazonaws.com
已啟用	已停用	不支援

預設原則不會對 CA 套用任何SourceArn限制。我們建議您手動套用下面顯示的較低權限政策，這會限制對特定 AWS 帳戶和特定私有 CA 的存取。如需詳細資訊，請參閱使用 Amazon S3 主控台新增儲存貯體政策。

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

如果您選擇允許預設策略，您可以隨時在以後修改它。

啟用 S3 封鎖公用存取 (BPA) CloudFront

新的 Amazon S3 儲存貯體預設會設定為啟用封鎖公用存取 (BPA) 功能。BPA 包含在 Amazon S3 安全最佳實務中，是一組存取控制，客戶可用來微調 S3 儲存貯體中物件和整個儲存貯體的存取。當 BPA 處於作用中且正確設定時，只有經過授權和驗證的 AWS 使用者才能存取值區及其內容。

AWS 建議在所有 S3 存儲桶上使用 BPA，以避免敏感信息暴露給潛在的對手。不過，如果您的 PKI 用戶端透過公用網際網路 (也就是說，未登入 AWS 帳戶) 擷取 CRL，則需要額外的規劃。本節說明如何使用 Amazon (內容交付網路 (CDN) 設定私有 PKI 解決方案 CloudFront，以提供 CRL，而不需經過驗證的用戶端存取 S3 儲存貯體。

> **Note**
>
> 使用 CloudFront 會對您的 AWS 帳戶產生額外費用。如需詳細資訊，請參閱 Amazon CloudFront 定價。
> 如果您選擇將 CRL 存放在已啟用 BPA 的 S3 儲存貯體中，但未使用 CloudFront，則必須建置另一個 CDN 解決方案，以確保 PKI 用戶端能夠存取 CRL。

使用雙酚 A 設置 Amazon S3

在 S3 中，像往常一樣為 CRL 創建一個新存儲桶，然後在其上啟用 BPA。

設定封鎖 CRL 公開存取的 Amazon S3 儲存貯體

1. 使用建立儲存貯體中的程序建立新的 S3 儲存貯體。在此程序期間，選取 [封鎖所有公用存取] 選項。

   如需詳細資訊，請參閱封鎖 Amazon S3 儲存的公開存取。

2. 建立值區後，請從清單中選擇其名稱，瀏覽至「權限」標籤，在「物件擁有權」區段中選擇「編輯」，然後選取「偏好的值區擁有者」。

3. 同樣在 [權限] 索引標籤上，將 IAM 政策新增至儲存貯體，如中所述Amazon S3 中 CRL 的訪問政策。

設定 CloudFront 雙酚 A

建立可存取私有 S3 儲存貯體的 CloudFront 發行版，並可將 CRL 提供給未經驗證的用戶端。

若要設定 CRL 的 CloudFront 發佈

1. 使用 Amazon CloudFront 開發人員指南中建立分發中的程序建立新 CloudFront 的分發。

   完成程序時，套用下列設定：

   - 在原始網域名稱中，選擇您的 S3 儲存貯體。
   - 針對限制值區存取，選擇「是」。
   - 選擇為原始存取身分建立新的身分識別。
   - 在「授與值區讀取權限」下選擇「是，更新儲存貯體政策」。

   > **Note**
   >
   > 在此程序中， CloudFront 修改值區政策以允許其存取值區物件。請考慮編輯此原則，只允許存取`crl`資料夾下的物件。

2. 在發行版初始化之後，請在 CloudFront 主控台中找到其網域名稱，並將其儲存以供下一個程序使用。

> **Note**
>
> 如果您的 S3 儲存貯體是在 us-east-1 以外的區域中新建立的，則當您透過存取已發佈的應用程式時，可能會收到 HTTP 307 暫時重新導向錯誤。 CloudFront存儲桶的地址可能需要幾個小時才能傳播。

設定適用雙酚 A 的 CA

設定新 CA 時，請將別名納入您的 CloudFront發行版本。

若要使用 CNAME 設定您的 CA，請執行下列項目 CloudFront

- 使用建立您的 CA 建立 CA (CLI) 的程序 。

  當您執行此程序時，撤銷檔案`revoke_config.txt`應包含下列幾行，以指定非公開 CRL 物件，並在中提供發佈端點的 URL： CloudFront

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

  之後，當您使用此 CA 發行證書時，它們將包含如下所示的塊：

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

> **Note**
>
> 如果您擁有由此 CA 發行的舊憑證，他們將無法存取 CRL。

加密您的 CRL

您可以選擇在包含 CRL 的 Amazon S3 儲存貯體上設定加密。 AWS 私有 CA 針對 Amazon S3 中的資產支援兩種加密模式：

- 使用亞馬遜 S3 代管的 AES-256 金鑰進行自動伺服器端加密。
- 使用客戶管理的加密， AWS Key Management Service 並根據您的規格進行 AWS KMS key 配置。

> **Note**
>
> AWS 私有 CA 不支援使用 S3 自動產生的預設 KMS 金鑰。

下列程序說明如何設定每個加密選項。

設定自動加密

完成以下步驟以啟用 S3 伺服器端加密。

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在「時段」表格中，選擇要保留 AWS 私有 CA 資產的值區。
3. 在您的儲存貯體頁面上，選擇屬性索引標籤。
4. 選擇預設加密卡片。
5. 選擇 啟用 。
6. 選擇 Amazon S3 密鑰（ SSE-S3 ）。
7. 選擇 Save Changes (儲存變更)。

設定自訂加密

完成下列步驟以啟用使用自訂金鑰加密。

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在「時段」表格中，選擇要保留 AWS 私有 CA 資產的值區。
3. 在您的儲存貯體頁面上，選擇屬性索引標籤。
4. 選擇預設加密卡片。
5. 選擇 啟用 。
6. 選擇金AWS Key Management Service 鑰 (SSE-KMS)。

7. 選擇「從您的 AWS KMS 金鑰中選擇」或「輸入 AWS KMS key ARN」。
8. 選擇 Save Changes (儲存變更)。
9. (選擇性) 如果您尚未擁有 KMS 金鑰，請使用下列建立金鑰指令建立一個 KMS AWS CLI 金鑰：

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

輸出包含 KMS 金鑰的金鑰識別碼和 Amazon 資源名稱 (ARN)。下面是一個示例輸出：

```
{
    "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. 使用下列步驟，您可以授與 AWS 私有 CA 服務主體使用 KMS 金鑰的權限。根據預設，所有 KMS 金鑰都是私密的；只有資源擁有者可以使用 KMS 金鑰來加密和解密資料。然而，資源擁有者可以授與其他使用者和資源存取 KMS 金鑰的許可。服務主體必須與存放 KMS 金鑰所在的區域相同。

    a. 首先，使`policy.json`用下列get-key-policy命令將 KMS 金鑰的預設原則儲存為：

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

    b. 在文字編輯器中開啟 `policy.json` 檔案。選取下列其中一個原則陳述式，然後將其新增至現有的策略。

    如果您的 Amazon S3 儲存貯體金鑰已啟用，請使用下列陳述式：

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

如果您的 Amazon S3 儲存貯體金鑰已停用，請使用下列陳述式：

```
{
   "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. 最後，使用下列put-key-policy命令套用更新的原則：

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

決定 CRL 發佈點 (CDP) URI

如果您使用 S3 儲存貯體做為 CA 的 CDP，CDP URI 可以採用下列其中一種格式。

- `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`

如果您已使用自訂 CNAME 來設定 CA，則 CDP URI 將包含 CNAME，例如 `http://alternative.example.com/crl/CA-ID.crl`

設定 AWS 私有 CA OCSP 的自訂網址

> **Note**
>
> 本主題適用於想要為品牌或其他目的自訂 OCSP 回應程式端點公開 URL 的客戶。如果您打算使用 AWS 私有 CA Managed OCSP 的預設組態，您可以略過此主題，並遵循設定撤銷中的組態指示進行。

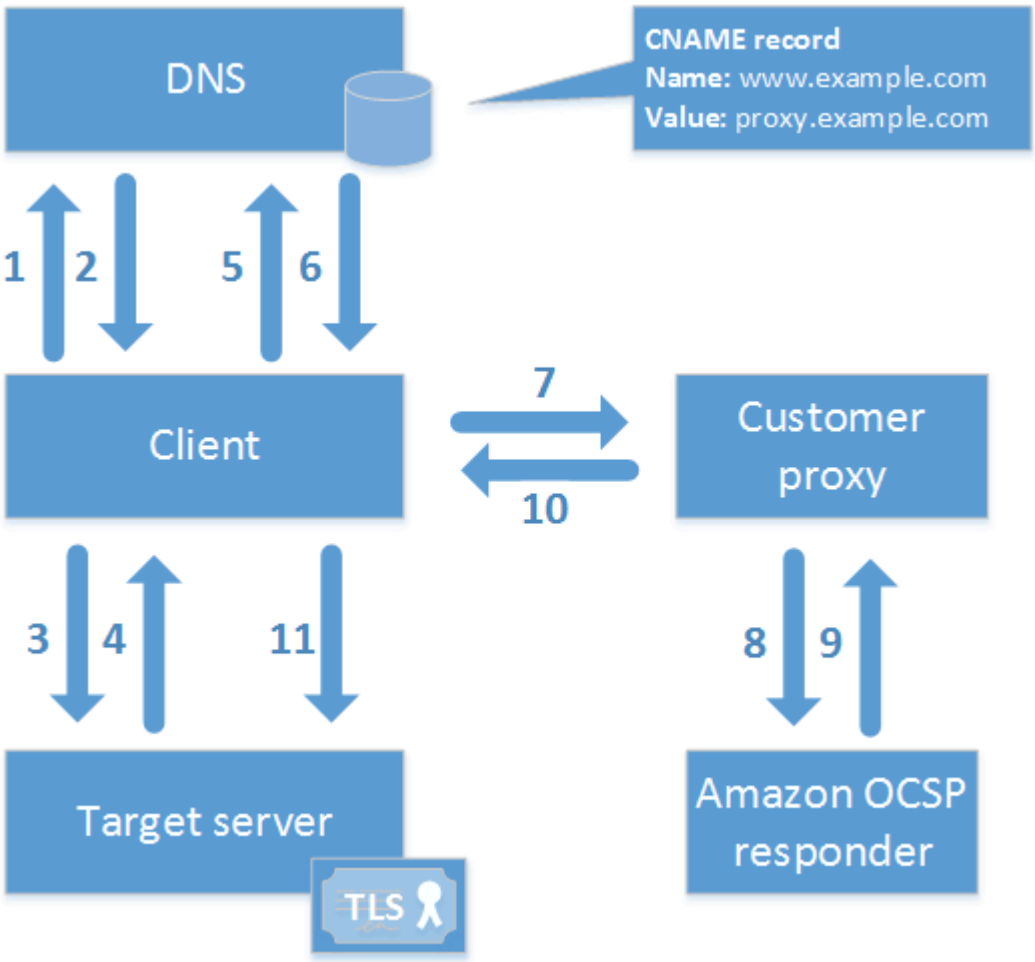
根據預設，當您啟用 OCSP 時 AWS 私有 CA，您發出的每個憑證都會包含 AWS OCSP 回應程式的 URL。這可讓要求密碼編譯安全連線的用戶端將 OCSP 驗證查詢直接傳送至。 AWS不過，在某些情況下，最好在您的憑證中陳述不同的 URL，同時還是最終將 OCSP 查詢提交給。 AWS

> **Note**
>
> 如需使用憑證撤銷清單 (CRL) 作為 OCSP 替代方案或補充的相關資訊，請參閱設定撤銷和規劃憑證撤銷清單 (CRL)。

設定 OCSP 的自訂 URL 涉及三個元素。

- CA 組態 — 在中為您的 CA 指定自訂 OCSP URL，如範例 2：在啟用 OCSP 的情況下建立 CA 並啟用自訂 CNAME中建立 CA (CLI) 的程序 所述。`RevocationConfiguration`
- DNS — 將 CNAME 記錄新增至您的網域組態，以將憑證中顯示的 URL 對應至代理伺服器 URL。如需詳細資訊，請參閱 建立 CA (CLI) 的程序 中的 範例 2：在啟用 OCSP 的情況下建立 CA 並啟用自訂 CNAME。
- 轉寄代理伺服器 — 設定 Proxy 伺服器，以透明方式將接收的 OCSP 流量轉送給 AWS OCSP 回應程式。

下圖說明這些元素如何協同運作。



如圖所示，自訂 OCSP 驗證程序包含下列步驟：

1. 用戶端會查詢目標網域的 DNS。
2. 用戶端接收目標 IP。
3. 客戶端打開與目標的 TCP 連接。
4. 用戶端會收到目標 TLS 憑證。
5. 用戶端會針對憑證中列出的 OCSP 網域查詢 DNS。
6. 用戶端接收代理 IP。
7. 用戶端傳送 OCSP 查詢到代理伺服器。
8. 代理伺服器會將查詢轉寄給 OCSP 回應程式。
9. 回應者會將憑證狀態傳回給 Proxy。
10. Proxy 會將憑證狀態轉送至用戶端。

11.如果憑證有效，用戶端會開始 TLS 交握。

> **Tip**
>
> 如上所述，您可以在設定 CA 之後，使用 Amazon CloudFront 和 Amazon 路線 53 來實作此範例。
>
> 1. 在中 CloudFront，建立發行版並設定它，如下所示：
>   - 建立符合您自訂 CNAME 的替代名稱。
>   - 將您的憑證繫結至該憑證。
>   - 設置八爪. ACM-PCA. <region>. 亞馬遜網站作為起源。
>   - 套用`Managed-CachingDisabled`原則。
>   - 將檢視器通訊協定原則設定為 HTTP 和 HTTPS。
>   - 將允許的 HTTP 方法設置為獲取，頭，選項，放置，發布，補丁，刪除。
> 2. 在路由 53 中，建立一個 DNS 記錄，將您的自訂 CNAME 對應至 CloudFront 發佈的 URL。

憑證機構模式

AWS 私有 CA 支援在兩種模式中的任何一種建立 CA。「一般目的」和「簡短 _ LIVED_ 憑證」模式會影響 CA 發行的憑證允許的有效期。

> **Note**
>
> AWS 私有 CA 不會對根 CA 憑證執行有效性檢查。

一般目的 (預設)

此模式允許 CA 簽發具有任何有效期的憑證。大多數應用程式使用此類型的憑證。通常，CA 也會指定撤銷機制。

短期生活證書

此模式定義專門發行憑證的 CA，最長有效期為七天。這些短期憑證過期的速度非常快，因此可以在沒有撤銷機制的情況下進行部署。對於某些應用程式而言，頻繁部署短期憑證比起產生網路和處理撤銷的額外負荷更有意義。

使用簡短證書模式的 CA 成本低於一般用途 CA。如需詳細資訊，請參閱AWS Private Certificate Authority 定價。

若要建立發行短期憑證的 CA，請使用建立 CA 的程序將UsageMode參數設定為「短期 _ LIVED_ 憑證」AWS CLI。

> **Note**
>
> AWS Certificate Manager 無法發行由具有短期模式的私有 CA 簽署的憑證。

下列服務支援使用短期憑 AWS 證：

- Amazon AppStream
- Amazon WorkSpaces

規劃復原力

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需區域和可用區域的相關 AWS 資訊，請參閱AWS 全域基礎結構。

備援與災難復原

規劃 CA 階層時，請考慮備援和 DR。 AWS 私有 CA 可在多個區域使用，可讓您在多個區域中建立冗餘 CA。該 AWS 私有 CA 服務以 99.9% 可用性的服務級別協議（SLA）運行。至少有兩種方法可供您考量備援和災難復原。您可以在根 CA 或最高次級 CA 設定備援。每種方法都有優缺點。

1. 您可以在兩個不同的 AWS 區域中建立兩個根 CA，以進行備援和災難復原。透過此設定，每個根 CA 會在一個 AWS 區域中獨立運作，在發生單一區域災難時保護您的安全。不過，建立備援根 CA 會增加操作複雜性：您必須將這兩個根 CA 憑證分配至您環境中瀏覽器和作業系統的信任存放區。
2. 您也可以建立多餘的從屬 CA 以部署在每個區 AWS 域中，並將它們鏈結至單一 AWS 區域中相同的唯一根 CA。採用這種方法的好處是，您只需要將單一根 CA 憑證分配至環境中的信任存放區即可。限制是，如果發生災難而影響根 CA 所在的 AWS 區域，則您沒有冗餘的根 CA。

AWS 私有 CA 最佳實務

最佳實務是可協助您更有效使用 AWS 私有 CA 的建議。以下最佳實務是根據目前 AWS Certificate Manager 和 AWS 私有 CA 客戶的實際體驗。

記錄 CA 結構和政策

AWS 建議記錄操作您 CA 所需要的所有政策和實務。這可能包括：

- CA 結構決策的理由
- 顯示 CA 及其關係的圖表
- CA 有效期間的政策
- CA 繼承規劃
- 路徑長度政策
- 許可目錄
- 管理控制結構的說明
- 安全

您可以在兩份文件中擷取此資訊，稱為認證原則 (CP) 和認證實務聲明 (CPS)。如需了解擷取 CA 操作重要資訊的框架，請參閱 RFC 3647。

如果可能的話，盡量減少使用根 CA

一般而言，根 CA 應僅用於發行中繼 CA 的憑證。這可以讓您透過不會造成損害的方式存放根 CA，並讓中繼 CA 執行發行終端實體憑證的日常任務。

但是，若您組織目前的實務是直接從根 CA 發行終端實體憑證，AWS 私有 CA 可支援此工作流程，同時改善安全和操作控制。在此案例中發行終端實體憑證需要 IAM 許可政策，允許您的根 CA 使用終端實體憑證範本。如需 IAM 政策的資訊，請參閱 適用於的 Identity and Access Management (IAM) AWS Private Certificate Authority。

> **Note**
>
> 此配置強加了可能導致操作挑戰的限制。例如，如果您的根 CA 遭到洩露或遺失，您必須建立新的根 CA，並將其分配到您環境中所有的用戶端。在完成此復原程序前，您將無法發行新的

憑證。直接從根 CA 發行憑證也會讓您無法限制存取及限制從您根發行的憑證數，即使這兩種皆是管理根 CA 的最佳實務。

為根 CA 提供自己的 AWS 帳戶

建議在兩個不同的AWS帳戶中建立根 CA 和從屬 CA 是建議的最佳作法。如此可以為您提供額外的保護，以及您根 CA 的存取控制。您可以透過將 CSR 從一個帳戶中的次級 CA 匯出，然後在不同帳戶中使用根 CA 進行簽署。這種方法的優點是您可以根據帳戶區別 CA 的控制。缺點是您無法使用AWS Management Console精靈簡化從根 CA 簽署下屬 CA 之 CA 憑證的程序。

Important

我們強烈建議您隨時使用多重要素驗證 (MFA)。AWS 私有 CA

單獨的管理員和發行者角

CA 管理員角色應與只需要發行終端實體憑證的使用者分開。如果您的 CA 管理員和憑證簽發者位於同一位置AWS 帳戶，您可以針對此目的建立 IAM 使用者來限制簽發者許可。

實作憑證的管理撤銷

當憑證被撤銷時，受管理的撤銷會自動向憑證用戶端提供通知。如果憑證的密碼編譯資訊遭到入侵或發出錯誤，您可能需要撤銷憑證。用戶端通常拒絕接受已撤銷的憑證。 AWS 私有 CA提供兩個管理撤銷的標準選項：線上憑證狀態通訊協定 (OCSP) 和憑證撤銷清單 (CRL)。如需詳細資訊，請參閱 設定憑證撤銷方法。

開啟 AWS CloudTrail

在建立並開始操作私有 CA 之前，請先開啟 CloudTrail 記錄功能。您可以使用擷取帳戶的 AWS API 呼叫歷史記錄 CloudTrail，以監控AWS部署。此歷史記錄包括從、AWS SDK AWS Management Console、和更高層級AWS服務進行AWS Command Line Interface的 API 呼叫。您也可以找出哪些使用者和帳戶呼叫 PCA API 操作、發出呼叫的來源 IP 地址，以及呼叫的發生時間。您可以使用 API 整合 CloudTrail 到應用程式中、為您的組織自動建立追蹤、檢查追蹤的狀態，以及控制系統管理員開啟和關閉 CloudTrail 登入的方式。如需詳細資訊，請參閱建立追蹤記錄。請前往 使用 CloudTrail 以查看 AWS 私有 CA 操作的範例線索。

旋轉 CA 私密金鑰

這是為您的私有 CA 定期更新私密金鑰的最佳實務。您可以透過匯入新的 CA 憑證，或是將私有 CA 取代為新的 CA，來更新金鑰。

> **Note**
>
> 如果您取代 CA 本身，請注意 CA 的 ARN 會變更。這會導致依賴硬式編碼 ARN 的自動化失敗。

刪除未使用的 CA

您可以永久刪除私有 CA。如果您不再需要 CA，或想要以具有較新私密金鑰的 CA 取代現有 CA，可執行此動作。若要安全刪除 CA，我們建議您依照 刪除您的私有 CA 中所述的流程操作。

> **Note**
>
> AWS 會在 CA 刪除之前持續向您收取費用。

封鎖對 CRL 的公開存取

AWS 私有 CA建議在包含 CRL 的儲存貯體上使用 Amazon S3 區塊公開存取 (BPA) 功能。這樣可以避免不必要地將私人 PKI 的詳細信息暴露給潛在的對手。BPA 是 S3 的最佳實務，預設會在新儲存貯體上啟用。在某些情況下需要額外的設置。如需詳細資訊，請參閱 啟用 S3 封鎖公用存取 (BPA) CloudFront。

Amazon EKS 應用程式最佳實務

使用AWS 私有 CA以 X.509 憑證佈建 Amazon EKS 時，請遵循 Amazon EKS 最佳實務指南中有關保護多租戶環境的建議。如需有關AWS 私有 CA與 Kubernetes 整合的一般資訊，請參閱。使用保護庫伯尼特 AWS 私有 CA

私人 CA 管理

您可以使用 AWS 私有 CA，建立完全 AWS 託管的根憑證授權單位和從屬憑證授權單位 (CA) 階層，供組織內部使用。若要管理憑證撤銷，您可以啟用線上憑證狀態通訊協定 (OCSP)、憑證撤銷清單 (CRL)，或同時啟用兩者。 AWS 私有 CA 儲存和管理您的 CA 憑證、CRL 和 OCSP 回應，而根授權單位的私密金鑰則由安全地儲存。 AWS

> **Note**
>
> 中的 OCSP 實作 AWS 私有 CA 不支援 OCSP 要求延伸模組。如果您提交包含多個憑證的 OCSP 批次查詢， AWS OCSP 回應程式只會處理佇列中的第一個憑證，並捨棄其他憑證。撤銷最多可能需要一個小時才會出現在 OCSP 回應中。

您可以 AWS 私有 CA 使用 AWS Management Console AWS CLI、和 AWS 私有 CA API 進行存取。下列主題說明如何使用主控台和 CLI。要了解有關 API 的更多信息，請參閱 AWS Private Certificate Authority API 參考。如需如何使用 API 的 Java 範例，請參閱 使用AWS 私有 CA應用程式介面 (Java 範例)。

主題

- 建立私有 CA
- 建立和安裝 CA 憑證
- 控制對私有 CA 的存取
- 列出私有 CA
- 檢視私人 CA
- 管理私有 CA 的標籤
- 更新您的私有 CA
- 刪除您的私有 CA
- 還原私有 CA

建立私有 CA

您可以使用本節中的程序來建立根 CA 或從屬 CA，產生符合您組織需求的可稽核信任關係階層。您可以使用 AWS Management Console、或 AWS CloudFormation的 PCA 部分來建立 CA。 AWS CLI

如需更新已建立之 CA 組態的相關資訊，請參閱更新您的私有 CA。

如需使用 CA 為您的使用者、裝置和應用程式簽署最終實體憑證的相關資訊，請參閱簽發私人終端實體證書。

> **Note**
>
> 從您帳戶的建立時間開始，會針對每個私有 CA 每月向您收取費用。
> 如需最新 AWS 私有 CA 定價資訊，請參閱AWS Private Certificate Authority 定價。您也可以使用定AWS 價計算器來估算成本。

主題

- 建立 CA (主控台) 的程序
- 建立 CA (CLI) 的程序
- 用 AWS CloudFormation 來建立 CA

建立 CA (主控台) 的程序

若要使用建立私有 CA，請完成下列步驟 AWS Management Console。

開始使用主控台

請登入您的 AWS 帳戶，然後在開啟 AWS 私有 CA 主機**https://console.aws.amazon.com/acm-pca/home**。

- 如果您要在沒有私人 CA 的區域中開啟主控台，則會顯示簡介頁面。選擇 [建立私人 CA]。
- 如果您要在已建立 CA 的區域中開啟主控台，則會開啟 [私人憑證授權單位] 頁面，其中包含 CA 清單。選擇 [建立 CA]。

模式選項

在主控台的 [模式選項] 區段中，選擇 CA 發行之憑證的到期模式。

- 一般用途 — 發行可以設定任何到期日的憑證。此為預設值。
- 短期憑證 — 發行有效期上限為七天的憑證。在某些情況下，短的有效期可以取代撤銷機制。

CA 類型選項

在主控台的 [類型選項] 區段中，選擇您要建立的私人憑證授權單位類型。

- 選擇根會建立新的 CA 階層。這個 CA 是以自我簽署的憑證為基礎的。它作為階層中其他 CA 和最終實體憑證的最終簽署權限。
- 選擇「從屬」會建立一個 CA，該 CA 必須由階層中的父項 CA 在其上方簽署。從屬 CA 通常用於建立其他從屬 CA 或發行終端實體憑證給使用者、電腦和應用程式。

> **Note**
>
> AWS 私有 CA 當您的下屬 CA 的父 CA 也由 AWS 私有 CA託管時，會提供自動簽署程序。您要做的就是選擇要使用的父 CA。
> 您的下屬 CA 可能需要由外部信任服務提供商簽名。如果是這樣，會 AWS 私有 CA 提供您一個憑證簽署要求 (CSR)，您必須下載並使用該要求才能取得已簽署的 CA 憑證。如需詳細資訊，請參閱 安裝由外部父 CA 簽署的從屬 CA 憑證。

主體辨別名稱選項

在主體辨別名稱選項下，設定私人 CA 的主旨名稱。您必須至少輸入下列其中一個選項的值：

- 組織 (O) — 例如，公司名稱
- 組織單位 (OU) — 例如，公司內的部門
- 國家名稱 (C) — 兩個字母的國家代碼
- 州或省名稱 — 州或省的全名
- 地區名稱 — 城市的名稱
- 通用名稱 (CN) — 用於識別 CA 的人類可讀字串。

> **Note**
>
> 您可以在發行時套用 APiPassThrough 範本，進一步自訂憑證的主旨名稱。如需詳細資訊和詳細範例，請參閱使用 API 傳遞範本發行具有自訂主體名稱的憑證。

由於支援憑證是自我簽署的，因此您為私有 CA 提供的主旨資訊可能比公用 CA 所包含的資訊更為稀疏。如需組成主旨辨別名稱之每個值的詳細資訊，請參閱 RFC 52 80。

關鍵演算法選項

在金鑰演算法選項下，選擇金鑰演算法和金鑰的位元大小。預設值是具有 2048 位元金鑰長度的 RSA 演算法。您可以從以下算法中進行選擇：

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

憑證撤銷選項

在 [憑證撤銷選項] 底下，您可以選取兩種與使用憑證的用戶端共用撤銷狀態的方法：

- 啟動 CRL 發佈
- 開啟 OCSP

您可以為 CA 設定這些撤銷選項，或兩者都不設定。雖然選擇性，但建議使用管理撤銷作為最佳作法。在完成此步驟之前，請設定憑證撤銷方法參閱以取得有關每種方法的優點、可能需要的初步設定以及其他撤銷功能的資訊。

> **Note**
>
> 如果您在未設定撤銷的情況下建立 CA，您隨時可以稍後進行設定。如需詳細資訊，請參閱 更新您的私有 CA。

若要設定 CRL

1. 在 [憑證撤銷選項] 下，選擇 [啟動 CRL 發佈]。
2. 若要為 CRL 項目建立 Amazon S3 儲存貯體，請選擇建立新的 S3 儲存貯體，然後輸入唯一的儲存貯體名稱。(您不需要包含指向儲存貯體的路徑。) 否則，在 S3 儲存貯體 URI 下，從清單中選擇現有儲存貯體。

   當您透過主控台建立新儲存貯體時，會 AWS 私有 CA 嘗試將必要的存取政策附加至儲存貯體，並停用其上的 S3 預設封鎖公用存取 (BPA) 設定。如果您改為指定現有值區，則必須確定帳戶和值區的 BPA 已停用。否則，建立 CA 的作業會失敗。如果 CA 已成功建立，您仍必須手動附加原則，

才能開始產生 CRL。使用中所述的其中一個策略模式Amazon S3 中 CRL 的訪問政策 。如需詳細資訊，請參閱使用 Amazon S3 主控台新增儲存貯體政策。

> **Important**
>
> 如果滿足下列所有條件，嘗試使用 AWS 私有 CA 主控台建立 CA 就會失敗：
>
> - 您正在設定 CRL。
> - 您 AWS 私有 CA 要求自動建立 S3 儲存貯體。
> - 您正在執行 S3 中的 BPA 設定。
>
> 在此情況下，主控台會建立值區，但會嘗試並無法公開存取。如果發生這種情況，請檢查您的 Amazon S3 設定，視需要停用 BPA，然後重複建立 CA 的程序。如需詳細資訊，請參閱封鎖 Amazon S3 儲存的公開存取。

3. 展開 CRL 設定以取得其他組態選項。

   - 新增自訂 CRL 名稱，為您的 Amazon S3 儲存貯體建立別名。此名稱包含在由 RFC 5280 定義的「CRL 發佈點」延伸模組中由 CA 所發行的憑證中。
   - 輸入您的 CRL 將保持有效期的天數。預設值為 7 天。對於在線 CRL，通常有效期為 2-7 天。AWS 私有 CA 嘗試在指定期間的中點重新產生 CRL。

4. 展開 S3 設定，以選用儲存貯體版本控制和儲存貯體存取記錄的組態

若要設定 OCSP

1. 在 [憑證撤銷選項] 底下，選擇 [開啟 OCSP]。
2. 在自訂 OCSP 端點 -選用欄位中，您可以為非 Amazon OCSP 端點提供完整網域名稱 (FQDN)。

   當您在此欄位中提供 FQDN 時，會將 FQDN AWS 私有 CA 插入每個已發行憑證的「授權單位資訊存取」延伸模組，以取代 OCSP 回應程式的預設 URL。 AWS 當端點收到包含自訂 FQDN 的憑證時，會查詢該位址是否有 OCSP 回應。為了使此機制正常運作，您需要採取兩個額外的動作：

   - 使用 Proxy 伺服器將到達自訂 FQDN 的流量轉送給 AWS OCSP 回應程式。
   - 將對應的 CNAME 記錄新增至您的 DNS 資料庫。

> **Tip**
>
> 如需使用自訂 CNAME 實作完整 OCSP 解決方案的詳細資訊，請參閱。設定 AWS 私有 CA OCSP 的自訂網址

例如，以下是自訂 OCSP 的 CNAME 記錄，就像它會出現在 Amazon 路線 53 中一樣。

記錄名稱	Type	路由政策	微分器	值/將流量路由到
替代例子	CNAME	簡便	-	代理例子

> **Note**
>
> CNAME 的值不得包含通訊協定前置詞，例如「http://」或「https://」。

新增標籤

在 [新增標籤] 底下，您可以選擇性地標記 CA。標籤是做為中繼資料的鍵/值對，可用來識別和整理 AWS 資源。如需 AWS 私有 CA 標籤參數的清單，以及如何在建立後將標籤新增至 CA 的指示，請參閱管理私有 CA 的標籤。

> **Note**
>
> 若要在建立程序期間將標籤附加至私有 CA，CA 管理員必須先將內嵌 IAM 政策與`CreateCertificateAuthority`動作建立關聯，並明確允許標記。如需詳細資訊，請參閱 Tag-on-create：建立時將標籤附加至 CA。

CA 權限選項

在 CA 權限選項下，您可以選擇性地將自動續訂權限委派給 AWS Certificate Manager 服務主體。如果授與此權限，ACM 只能自動更新此 CA 所產生的私有終端實體憑證。您可以隨時使用 AWS 私有 CA CreatePermissionAPI 或建立權限 CLI 命令指派續訂權限。

預設是啟用這些許可。

> **Note**
>
> AWS Certificate Manager 不支援短期憑證的自動更新。

定價

在 [定價] 底下，確認您瞭解私有 CA 的定價。

> **Note**
>
> 如需最新 AWS 私有 CA 定價資訊，請參閱AWS Private Certificate Authority 定價。您也可以使用定AWS 價計算器來估算成本。

建立 CA

檢查完所有輸入的資訊之後，請選擇「建立 CA」。CA 的詳細資料頁面隨即開啟，並將其狀態顯示為擱置憑證。

> **Note**
>
> 在詳細資料頁面上，您可以選擇 「動作」、「安裝 CA 憑證」來完成 CA 的設定，或者稍後返回「私人憑證授權單位」清單，並完成適用於您的情況的安裝程序：
>
> - 安裝根 CA 憑證
> - 安裝由主控的從屬 CA 憑證 AWS 私有 CA
> - 安裝由外部父 CA 簽署的從屬 CA 憑證

建立 CA (CLI) 的程序

使用create-certificate-authority命令建立私有 CA。您必須指定 CA 組態 (包含演算法和主體名稱資訊)、撤銷組態 (如果您打算使用 OCSP 和/或 CRL)，以及 CA 類型 (根或從屬)。組態和撤銷組態詳細資料包含在兩個檔案中，您可以作為指令引數提供這兩個檔案。或者，您還可以配置 CA 使用模式 (用於發行標準或短期證書) ，附加標籤並提供冪等性令牌。

如果您要設定 CRL，則在發出create-certificate-authority命令之前，必須先有一個安全的 Amazon S3 儲存貯體。如需詳細資訊，請參閱 Amazon S3 中 CRL 的訪問政策 。

CA 組態檔會指定下列資訊：

- 演算法的名稱
- 用於建立 CA 私密金鑰的金鑰大小
- CA 用於簽署的簽署演算法類型
- X.500 主旨資訊

OCSP 的撤銷組態會定義具有下列資訊的`OcspConfiguration`物件：

- 標`Enabled`誌設置為「真」。
- (選擇性) 宣告為的值的`OcspCustomCname`自訂 CNAME。

CRL 的撤銷配置定義了具有以下信息的`CrlConfiguration`對象：

- 標`Enabled`誌設置為「真」。
- CRL 有效期（以天為單位）（CRL 的有效期）。
- 將包含 CRL 的 Amazon S3 存儲桶。
- (選用) 確定 CRL 是否可公開存取的 S3 ObjectAcl 值。在此處顯示的範例中，公用存取遭到封鎖。如需詳細資訊，請參閱 啟用 S3 封鎖公用存取 (BPA) CloudFront。
- (選擇性) 由 CA 發行的憑證中包含的 S3 儲存貯體的 CNAME 別名。如果 CRL 不可公開訪問，這將指向一個分發機制，如 Amazon CloudFront。
- (選擇性) 具有下列資訊的`CrlDistributionPointExtensionConfiguration`物件：
  - 標`OmitExtension`誌設置為「真」或「假」。這可控制 CDP 延伸模組的預設值是否要寫入 CA 所發行的憑證。如需 CDP 延伸模組的詳細資訊，請參閱決定 CRL 發佈點 (CDP) URI 。如果 OmitExtension 是「真」，則 CustomCname 無法設置 A。

> **Note**
>
> 您可以同時定義`OcspConfiguration`物件和物件，在相同 CA 上啟用兩種撤銷機制。`CrlConfiguration`如果您未提供任何--revocation-configuration參數，則預設會停用這兩種機制。如果您稍後需要撤銷驗證支援，請參閱更新一個 CA。

下列範例假設您已使用有效的預設區域、端點和認證來設定.aws組態目錄。如需有關設定 AWS CLI 環境的資訊，請參閱組態和認證檔案設定。為了便於閱讀，我們在示例命令中以 JSON 文件形式提供 CA 配置和撤銷輸入。視需要修改範例檔案以供您使用。

除非另有說明，否則所有示例都使用以下ca_config.txt配置文件。

檔案:ca_config.txt

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

範例 1：建立啟用 OCSP 的 CA

在此範例中，撤銷檔案會啟用預設的 OCSP 支援，此支援會使用 AWS 私有 CA 回應程式來檢查憑證狀態。

檔案：適用於 OCSP 的 revoke_config.txt

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

命令

```
$ 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
```

如果成功，此命令會輸出新 CA 的 Amazon 資源名稱 (ARN)。

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

命令

```
$ 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
```

如果成功，此命令會輸出 CA 的 Amazon 資源名稱 (ARN)。

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

使用以下命令檢查 CA 的配置。

```
$ 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
```

此說明應包含以下部分。

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

範例 2：在啟用 OCSP 的情況下建立 CA 並啟用自訂 CNAME

在此範例中，撤銷檔案會啟用自訂 OCSP 支援。`OcspCustomCname`參數會使用完整網域名稱 (FQDN) 做為其值。

當您在此欄位中提供 FQDN 時，會將 FQDN AWS 私有 CA 插入每個已發行憑證的「授權單位資訊存取」延伸模組，以取代 OCSP 回應程式的預設 URL。 AWS 當端點收到包含自訂 FQDN 的憑證時，會查詢該位址是否有 OCSP 回應。為了使此機制正常運作，您需要採取兩個額外的動作：

- 使用 Proxy 伺服器將到達自訂 FQDN 的流量轉送給 AWS OCSP 回應程式。
- 將對應的 CNAME 記錄新增至您的 DNS 資料庫。

> **Tip**
>
> 如需使用自訂 CNAME 實作完整 OCSP 解決方案的詳細資訊，請參閱。設定 AWS 私有 CA OCSP 的自訂網址

例如，以下是自訂 OCSP 的 CNAME 記錄，就像它會出現在 Amazon 路線 53 中一樣。

記錄名稱	Type	路由政策	微分器	值/將流量路由到
替代例子	CNAME	簡便	-	代理例子

> **Note**
>
> CNAME 的值不得包含通訊協定前置詞，例如「http://」或「https://」。

檔案：適用於 OCSP 的 revoke_config.txt

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

命令

```
$ 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
```

如果成功，此命令會輸出 CA 的 Amazon 資源名稱 (ARN)。

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

使用以下命令檢查 CA 的配置。

```
$ 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
```

此說明應包含以下部分。

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

範例 3：建立含有附加 CRL 的 CA

在此範例中，撤銷組態會定義 CRL 參數。

檔案:revoke_config.txt

```
{
```

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

命令

```
$ 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
```

如果成功，此命令會輸出 CA 的 Amazon 資源名稱 (ARN)。

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

使用以下命令檢查 CA 的配置。

```
$ 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
```

此說明應包含以下部分。

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

範例 4：建立附加 CRL 並啟用自訂 CNAME 的 CA

在此範例中，撤銷組態定義了包含自訂 CNAME 的 CRL 參數。

檔案:revoke_config.txt

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

命令

```
$ 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
```

如果成功，此命令會輸出 CA 的 Amazon 資源名稱 (ARN)。

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

使用以下命令檢查 CA 的配置。

```
$ 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
```

此說明應包含以下部分。

```
"RevocationConfiguration": {
```

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

範例 5：建立 CA 並指定使用模式

在此範例中，建立 CA 時會指定 CA 使用模式。如果未指定，使用模式參數會預設為「一般用途」。在此範例中，參數設定為 SHORT_LIVED_FIRY，這表示 CA 將發行最長有效期為七天的憑證。在設定撤銷不方便的情況下，遭入侵的短期憑證會在正常作業中快速過期。因此，此範例 CA 缺少撤銷機制。

> **Note**
>
> AWS 私有 CA 不會對根 CA 憑證執行有效性檢查。

```
$ 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
```

使用中的describe-certificate-authority命令 AWS CLI 來顯示有關結果 CA 的詳細資訊，如下列命令所示：

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

範例 6：建立使用中目錄登入的 CA

您可以創建適合在 Microsoft 活動目錄（AD）的企業 NTAuth 存儲，在那裡它可以發出卡片登錄或域控制器證書使用的私有 CA。如需將 CA 憑證匯入 AD 的相關資訊，請參閱如何將協力廠商憑證授權單位 (CA) 憑證匯入企業 NTAuth 存放區。

Microsoft 認證工具可用於通過調用選項來發布 AD 中的 CA 證書。-dspublish使用 certutil 發佈至 AD 的憑證在整個樹系中受到信任。使用群組原則，您也可以將信任限制為整個樹系的子集，例如單一網域或網域中的一組電腦。若要登入才能運作，發行的 CA 也必須在 NTAuth 存放區中發行。如需詳細資訊，請參閱使用群組原則將憑證散發到用戶端電腦。

此範例使用下列`ca_config_AD.txt`組態檔案。

檔案: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"
         }
      ]
   }
}
```

命令

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

如果成功，此命令會輸出 CA 的 Amazon 資源名稱 (ARN)。

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

使用以下命令檢查 CA 的配置。

```
$ 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
```

此說明應包含以下部分。

```
...
```

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

範例 7：建立具有附加 CRL 且已發行憑證省略 CDP 副檔名的事項 CA

您可以建立適合發行 Matter 智慧家居標準憑證的私有 CA。在此範例中，中的 CA 組態`ca_config_PAA.txt`定義了「供應商識別碼」(VID) 設定為 FFF1 的「事項產品證明授權單位」(PAA)。

檔案: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"
      }
    ]
```

```
  }
}
```

撤銷組態會啟用 CRL，並設定 CA 省略任何已發行憑證中的預設 CDP URL。

檔案:revoke_config.txt

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

命令

```
$ 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
```

如果成功，此命令會輸出 CA 的 Amazon 資源名稱 (ARN)。

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

使用以下命令檢查 CA 的配置。

```
$ 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
```

此說明應包含以下部分。

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

用 AWS CloudFormation 來建立 CA

如需使用建立私有 CA 的詳細資訊 AWS CloudFormation，請參閱使AWS CloudFormation 用指南中的AWS 私有 CA 資源類型參考。

建立和安裝 CA 憑證

完成下列程序來建立及安裝您的私有 CA 憑證。您的 CA 接著便會準備好可供使用。

AWS 私有 CA 支援三種安裝 CA 憑證的案例：

- 為由下列項目主控的根 CA 安裝憑證 AWS 私有 CA
- 安裝父系授權單位是由 AWS 私有 CA託管的次級 CA 憑證
- 安裝父系授權單位是於外部進行託管的次級 CA 憑證

下列各節說明每個案例的程序。主控台程序會從主控台頁面的 Private CAs (私有 CA) 上開始。

相容簽署演算法

CA 憑證的簽署演算法支援取決於父 CA 和上的簽署演算法 AWS 區域。下列條件約束適用於主控台和 AWS CLI 作業。

- 具有 RSA 簽署演算法的父 CA 可以發行具有下列演算法的憑證：

- 沙
- 沙 384
- 沙

- 在舊版中 AWS 區域，具有 EDCSA 簽署演算法的父 CA 可以發行具有下列演算法的憑證：
  - 埃克萨
  - 沙 384 埃德薩
  - 沙 512 埃克

  舊版 AWS 區域 包括：

區域名稱	地理位置
eu-north-1	歐洲 (斯德哥爾摩)
me-south-1	Middle East (Bahrain)
ap-south-1	亞太區域 (孟買)
eu-west-3	Europe (Paris)
us-east-2	美國東部 (俄亥俄)
af-south-1	非洲 (開普敦)
eu-west-1	歐洲 (愛爾蘭)
eu-central-1	歐洲 (法蘭克福)
sa-east-1	南美洲 (聖保羅)
ap-east-1	亞太區域 (香港)

區域名稱	地理位置
us-east-1	美國東部 (維吉尼亞北部)
ap-northeast-2	亞太區域 (首爾)
eu-west-2	歐洲 (倫敦)
ap-northeast-1	亞太區域 (東京)
us-gov-east-1	AWS GovCloud (美國東部)
us-gov-west-1	AWS GovCloud (美國西部)
us-west-2	美國西部 (奧勒岡)
us-west-1	美國西部 (加利佛尼亞北部)
ap-southeast-1	亞太區域 (新加坡)
ap-southeast-2	亞太區域 (悉尼)

- 在非舊版中 AWS 區域，下列規則適用於 EDCSA：
  - 具有 EC_PRIME256v1 簽署演算法的父系 CA 可以使用 ECDSA P256 發行憑證。
  - 具有簽署演算法的父系 CA 可以使用 ECDSA P384 簽署演算法來發行憑證。

安裝根 CA 憑證

您可以從 AWS Management Console 或安裝根 CA 憑證 AWS CLI。

建立並安裝私人根 CA (主控台) 的憑證

1. (選擇性) 如果您尚未進入 CA 的詳細資料頁面，請在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台。在 [私人憑證授權單位] 頁面上，選擇狀態為擱置憑證或作用中的根 CA。
2. 選擇「動作」、「安裝 CA 憑證」以開啟「安裝根 CA 憑證」頁面。
3. 在「指定根 CA 憑證參數」下，指定下列憑證參數：

   - 有效性 — 指定 CA 憑證的到期日和時間。根 CA 憑證的 AWS 私有 CA 預設有效期為 10 年。
   - 簽章演算法 — 指定根 CA 發行新憑證時要使用的簽署演算法。可用的選項會根據建立 CA 的 AWS 區域 位置而有所不同。如需詳細資訊，請參閱相容簽署演算法支援的加密演算法、和SigningAlgorithm中的CertificateAuthorityConfiguration。
     - 沙
     - 沙 384
     - 沙

   檢查您的設定是否正確，然後選擇 [確認並安裝]。 AWS 私有 CA 匯出 CA 的 CSR、使用根 CA 憑證範本產生憑證，並自行簽署憑證。 AWS 私有 CA 然後匯入自我簽署的根 CA 憑證。
4. CA 的詳細資料頁面會在頂端顯示安裝狀態 (成功或失敗)。如果安裝成功，新完成的根 CA 會在 [一般] 窗格中顯示 [作用中] 狀態。

建立並安裝私有根 CA 的憑證 (AWS CLI)

1. 產生憑證簽署要求 (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
```

   產生的檔案 `ca.csr` (以 base64 格式編碼的 PEM 檔案) 具有下列外觀。

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

您可以使用 OpenSSL 來檢視和驗證企業社會責任的內容。

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

這會產生類似下列的輸出。

```
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. 使用上一個步驟中的 CSR 作為`--csr`參數的引數，發行根憑證。

> **Note**
>
> 如果您使用的是 1.6.3 AWS CLI 版或更新版本，請在指定所需的輸入檔案`fileb://`時使用前置詞。如此可確保正確 AWS 私有 CA 剖析 Base64 編碼的資料。

```
$ 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. 擷取根憑證。

```
$ 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
```

產生的檔案 `cert.pem` (以 base64 格式編碼的 PEM 檔案) 具有下列外觀。

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

您可以使用 OpenSSL 來檢視和驗證憑證的內容。

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

這會產生類似下列的輸出。

```
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. 匯入根 CA 憑證以將其安裝在 CA 上。

> **Note**
>
> 如果您使用的是 1.6.3 AWS CLI 版或更新版本，請在指定所需的輸入檔案`fileb://`時使用前置詞。如此可確保正確 AWS 私有 CA 剖析 Base64 編碼的資料。

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

檢查 CA 的新狀態。

```
$ 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
```

狀態現在會顯示為「作用中」。

```
{
```

```
    "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
            }
        }
    }
}
```

安裝由主控的從屬 CA 憑證 AWS 私有 CA

您可以使用 AWS Management Console 為 AWS 私有 CA 託管的下屬 CA 建立和安裝憑證。

若要建立並安裝您的 AWS 私有 CA 託管下屬 CA 憑證

1. (選擇性) 如果您尚未進入 CA 的詳細資料頁面，請在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台。在 [私人憑證授權單位] 頁面上，選擇狀態為擱置憑證或作用中的從屬 CA。
2. 選擇「動作」、「安裝 CA 憑證」以開啟「安裝從屬 CA 憑證」頁面。
3. 在 [安裝附屬 CA 憑證] 頁面的 [選取 CA 類型] 下，選擇AWS Private CA安裝由 AWS 私有 CA管理的憑證。
4. 在 [選取父項 CA] 底下，從 [父項私人 CA] 清單中選擇 CA。系統會篩選清單，以顯示符合下列準則的 CA：
   - 您有權使用 CA。
   - CA 不會自行簽署。
   - CA 處於狀態ACTIVE。
   - CA 模式為GENERAL_PURPOSE。
5. 在「指定從屬 CA 憑證參數」下，指定下列憑證參數：
   - 有效性 — 指定 CA 憑證的到期日和時間。
   - 簽章演算法 — 指定根 CA 發行新憑證時要使用的簽署演算法。選項為：
     - 沙
     - 沙 384
     - 沙
   - 路徑長度 — 從屬 CA 簽署新憑證時可新增的信任層數目。路徑長度為零 (預設值) 表示只能建立最終實體憑證，而不能建立 CA 憑證。長度為一以上的路徑表示次級 CA 可發行憑證來建立其他 CA 次級。
   - 範本 ARN — 顯示此 CA 憑證之組態範本的 ARN。如果您變更了指定的 Path length (路徑長度)，範本也會變更。如果您使用 CLI 問題憑證命令或 API IssueCertificate動作建立憑證，則必須手動指定 ARN。如需可用 CA 憑證範本的資訊，請參閱 瞭解憑證範本。
6. 檢查您的設定是否正確，然後選擇 [確認並安裝]。 AWS 私有 CA 匯出 CSR、使用從屬 CA 憑證範本產生憑證，並使用選取的父 CA 簽署憑證。 AWS 私有 CA 然後匯入已簽署的下屬 CA 憑證。
7. CA 的詳細資料頁面會在頂端顯示安裝狀態 (成功或失敗)。如果安裝成功，新完成的從屬 CA 會在 [一般] 窗格中顯示 [作用中] 狀態。

安裝由外部父 CA 簽署的從屬 CA 憑證

如建立 CA (主控台) 的程序或所述建立從屬私有 CA 之後建立 CA (CLI) 的程序 ，您可以選擇透過安裝由外部簽署授權單位簽署的 CA 憑證來啟動它。使用外部 CA 簽署您的下屬 CA 憑證時，您必須先設定外部信任服務提供者作為簽署授權單位，或安排使用第三方提供者。

> **Note**
>
> 建立或取得外部信任服務提供者的程序不在本指南的範圍內。

建立下屬 CA 並可存取外部簽署授權單位之後，請完成下列工作：

1. 從取得憑證簽署要求 (CSR) AWS 私有 CA。
2. 將 CSR 提交給您的外部簽署授權單位，並取得已簽署的 CA 憑證以及任何鏈結憑證。
3. 將 CA 憑證和鏈結匯入， AWS 私有 CA 以啟動您的下屬 CA。

如需詳細程序，請參閱外部簽署的私有 CA 憑證 。

控制對私有 CA 的存取

對私有 CA 具有必要權限的任何使用者都 AWS 私有 CA 可以使用該 CA 簽署其他憑證。CA 擁有者可以發行憑證，或將簽發憑證所需的權限委派給位於相同憑證的 AWS Identity and Access Management (IAM) 使用者 AWS 帳戶。如果 CA 擁有者透過以資源為基礎的政策授權，則位於不同 AWS 帳戶中的使用者也可以發行憑證。

授權使用者，無論是單一帳戶還是跨帳戶，都可以在發行憑證時使用 AWS 私有 CA 或 AWS Certificate Manager 資源。從 AWS 私有 CA IssueCertificateAPI 或問題憑證 CLI 命令發出的憑證不受管理。此類憑證需要在目標裝置上手動安裝，並在到期時手動續訂。管理從 ACM 主控台、ACM RequestCertificateAPI 或要求憑證 CLI 命令發出的憑證。這類憑證可以輕鬆地安裝在與 ACM 整合的服務中。如果 CA 管理員允許，且核發者的帳戶具有 ACM 的服務連結角色，則受管理憑證會在到期時自動續約。

主題

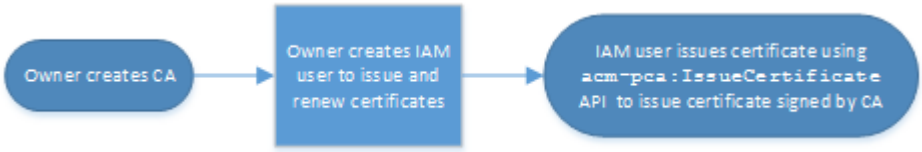
- 為 IAM 使用者建立單一帳戶許可
- 附加跨帳戶存取政策

為 IAM 使用者建立單一帳戶許可

當 CA 管理員 (也就是 CA 的擁有者) 和憑證簽發者位於單一 AWS 帳戶時，最佳做法是建立具有有限權限的 AWS Identity and Access Management (IAM) 使用者，以區隔發行者和管理員角色。如需將 IAM 搭配 AWS 私有 CA使用的詳細資訊以及許可範例，請參閱適用於的 Identity and Access Management (IAM) AWS Private Certificate Authority。

單一帳戶案例 1：發行未受管理憑證

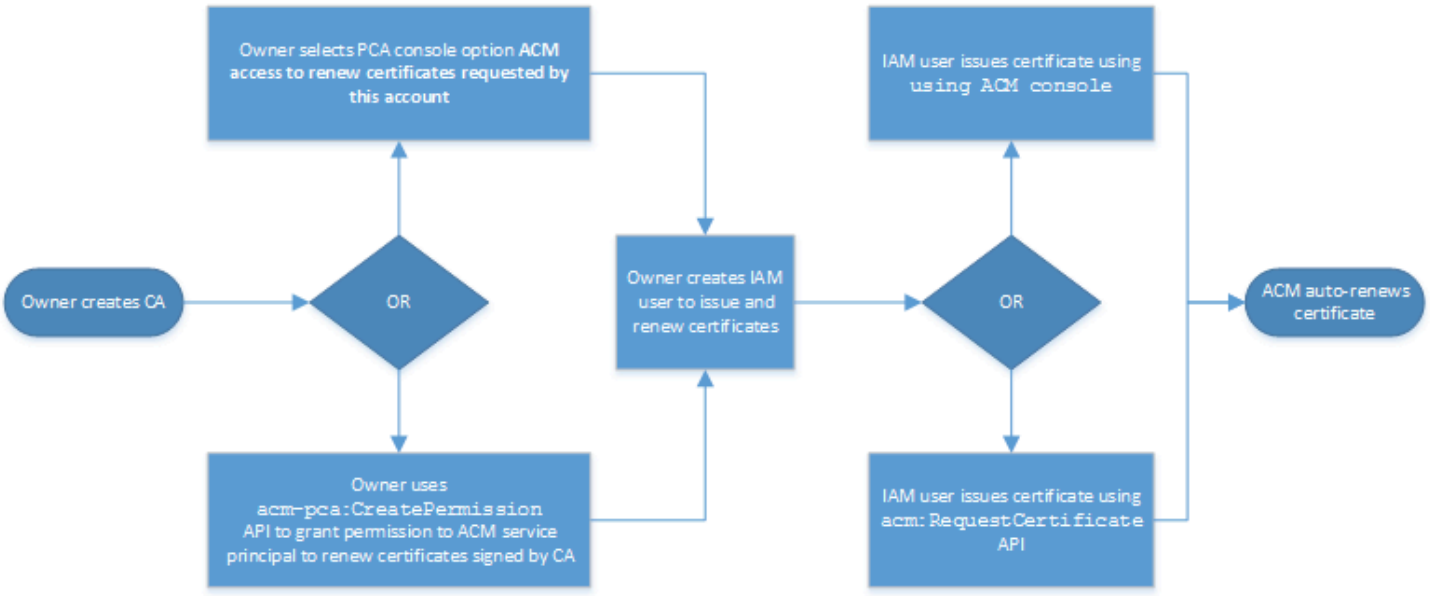
在此情況下，帳戶擁有者會建立私有 CA，然後建立具有發行私有 CA 簽署之憑證的權限的 IAM 使用者。IAM 使用者透過呼叫 AWS 私有 CA `IssueCertificate` API 來發出憑證。



以這種方式發行的憑證不受管理，這表示系統管理員必須匯出這些憑證，並將其安裝在要使用的裝置上。它們也必須在到期時手動更新。使用此 API 發行憑證需要在 OpenSSL 或類似程式之外產生的憑證簽署要求 (CSR) 和 key pair。 AWS 私有 CA 如需詳細資訊，請參閱`IssueCertificate`文件。

單一帳戶案例 2：透過 ACM 發行受管理憑證

第二種情況涉及來自 ACM 和 PCA 的 API 操作。帳戶擁有者會像以前一樣建立私有 CA 和 IAM 使用者。然後，帳戶擁有者會授與 ACM 服務主體的權限，以便自動更新由此 CA 簽署的任何憑證。IAM 使用者再次發行憑證，但這次呼叫 ACM `RequestCertificate` API 來處理 CSR 和金鑰產生。憑證到期時，ACM 會自動執行續約工作流程。



帳戶擁有者可選擇在建立 CA 期間或使用 PCA `CreatePermission` API 期間或之後透過管理主控台授與續訂權限。從此工作流程建立的受管理憑證可與 ACM 整合的 AWS 服務搭配使用。

下節包含授與續訂權限的程序。

將憑證續訂權限指派給 ACM

透過 AWS Certificate Manager (ACM) 中的受管續約，您可以自動執行公有憑證和私有憑證的憑證續約程序。為了讓 ACM 自動更新私有 CA 所產生的憑證，CA 本身必須將所有可能的權限授與 ACM 服務主體。如果 ACM 沒有這些續訂權限，CA 的擁有者 (或授權代表) 必須在每個私有憑證到期時手動重新發行。

> **Important**
>
> 僅當 CA 擁有者和憑證簽發者位於相同 AWS 帳戶時，才會套用這些指派續訂權限的程序。如需跨帳戶案例，請參閱附加跨帳戶存取政策。

您可以在建立私有 CA 的期間，或是在其之後 CA 處於 `ACTIVE` 狀態的任何時間委派續約許可。

您可以透過 AWS 私有 CA 主控台、AWS Command Line Interface (AWS CLI)，或 AWS 私有 CA API 來管理私有 CA 許可：

若要將私人 CA 權限指派給 ACM (主控台)

1. 登入您的 AWS 帳戶，然後在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台。
2. 在 [私人憑證授權單位] 頁面上，從清單中選擇您的私有 CA。
3. 選擇 [動作]、[設定 CA 權限]。
4. 選取授權 ACM 存取以更新此帳戶要求的憑證。
5. 選擇儲存。

若要在 AWS 私有 CA (AWS CLI) 中管理 ACM 權限

使用 [建立權限] 命令將權限指派給 ACM。您必須指派必要的權限 (`IssueCertificate`、`GetCertificate`、和`ListPermissions`)，ACM 才能自動更新您的憑證。

```
$ 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
```

使用 list-permissions 命令來列出 CA 委派的許可。

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

使用 [刪除權限] 命令撤銷 CA 指派給 AWS 服務主體的權限。

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

附加跨帳戶存取政策

當 CA 管理員和憑證簽發者位於不同的 AWS 帳戶時，CA 管理員必須共用 CA 存取權。這是通過將以資源為基礎的策略附加到 CA 來完成。該政策將發行許可授予特定主體，該主體可以是 AWS 帳戶擁有者、IAM 使用者、 AWS Organizations ID 或組織單位 ID。

CA 管理員可以透過下列方式附加及管理原則：

- 在管理主控台中，使用 AWS Resource Access Manager (RAM)，這是跨帳戶共用 AWS 資源的標準方法。當您與其他帳號中的主參 AWS RAM 與者共用 CA 資源時，必要的以資源為基礎的策略會自動附加至 CA。如需有關 RAM 的詳細資訊，請參閱使AWS RAM 用者指南。

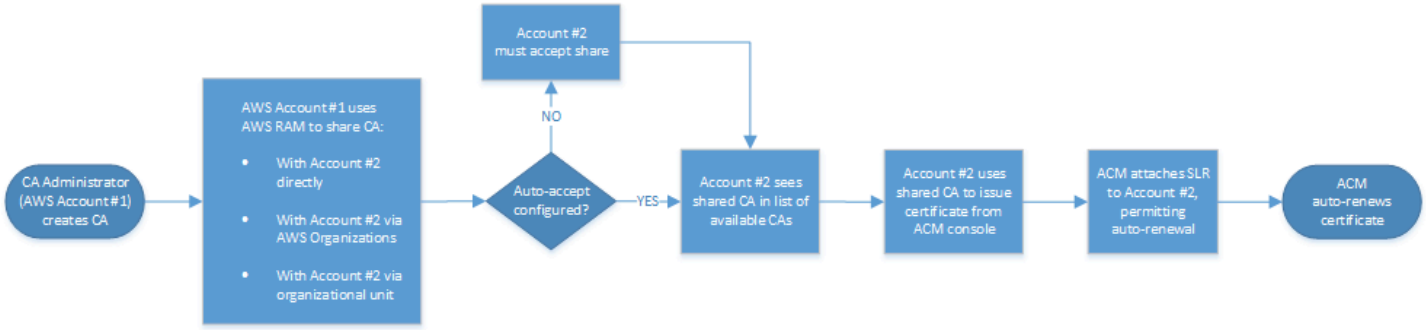
  > **Note**
  >
  > 您可以選擇 CA，然後選擇 [動作]、[管理資源共用]，輕鬆開啟 RAM 主控台。

- 以程式設計方式使用 PCA API PutPolicyGetPolicy、和 DeletePolicy.
- 手動方式使用 PCA 命令放入原則、取得原則和刪除原則。 AWS CLI

只有控制台方法需要 RAM 訪問。

跨帳戶案例 1：從主控台發出受管理憑證

在此情況下，CA 管理員會使用 AWS Resource Access Manager (AWS RAM) 與另一個 AWS 帳戶共用 CA 存取權，讓該帳戶發行受管理的 ACM 憑證。圖表顯示 AWS RAM 可直接與帳戶共用 CA，或間接透過帳戶所屬成員的 AWS Organizations ID 共用 CA。

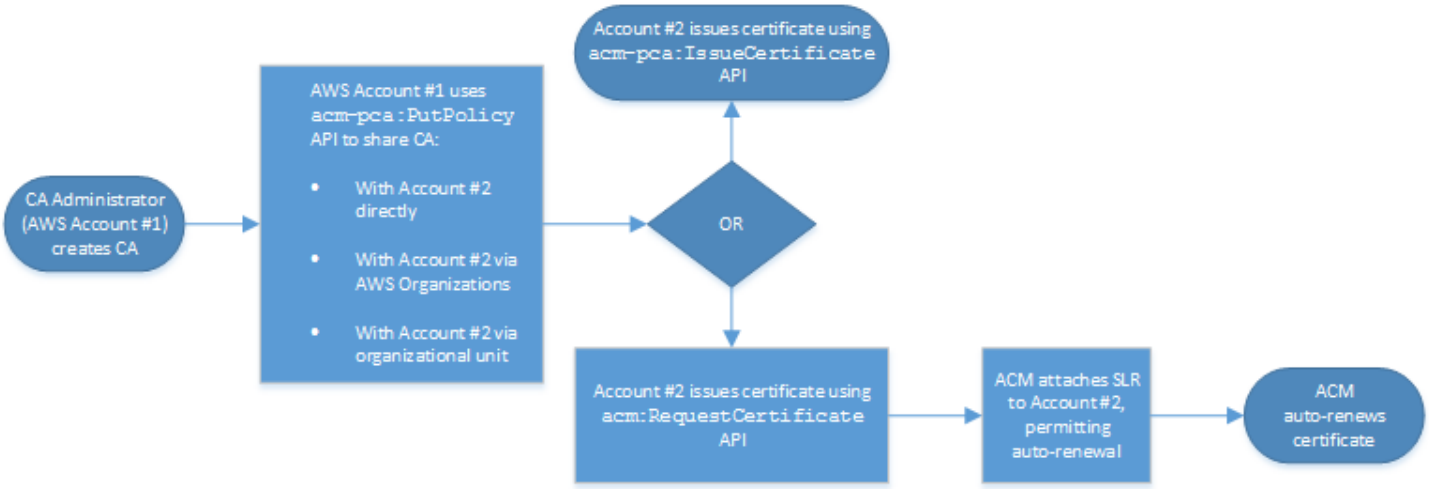


RAM 透過共用資源之後 AWS Organizations，收件者主參與者必須接受資源，資源才會生效。收件者可以設定 AWS Organizations 為自動接受提供的共用。

> **Note**
>
> 收件者帳戶負責在 ACM 中設定自動續約。一般而言，在第一次使用共用 CA 時，ACM 會安裝服務連結角色，以允許其進行無人看管憑證呼叫。 AWS 私有 CA如果失敗 (通常是因為缺少權限)，則不會自動更新來自 CA 的憑證。只有 ACM 使用者可以解決問題，而不是 CA 管理員。如需詳細資訊，請參閱搭配 ACM 使用服務連結角色 (SLR)。

跨帳戶案例 2：使用 API 或 CLI 發行受管理和非受管憑證

第二個案例展示了使用和 AWS 私有 CA API 可能的共享 AWS Certificate Manager 和發行選項。所有這些操作也可以使用相應的 AWS CLI 命令來執行。



由於此範例中會直接使用 API 作業，因此憑證簽發者可以選擇兩種 API 作業來發行憑證。PCA API 動作會`IssueCertificate`產生未受管理的憑證，該憑證不會自動更新，而且必須匯出並手動安裝。ACM API 動作會RequestCertificate產生受管理的憑證，該憑證可輕鬆安裝在 ACM 整合式服務上，並自動續訂。

> **Note**
>
> 收件者帳戶負責在 ACM 中設定自動續約。一般而言，在第一次使用共用 CA 時，ACM 會安裝服務連結角色，以便在其上進行無人看管憑證呼叫。 AWS 私有 CA如果失敗 (通常是因為缺少權限)，來自 CA 的憑證將不會自動更新，只有 ACM 使用者才能解決問題，而不是 CA 管理員。如需詳細資訊，請參閱搭配 ACM 使用服務連結角色 (SLR)。

列出私有 CA

您可以使用 AWS 私有 CA 主控台或 AWS CLI 列出您擁有或有權存取的私有 CA。

使用控制台列出可用的 CA

1. 登入您的 AWS 帳戶，然後在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台。
2. 檢閱私人憑證授權單位清單中的資訊。您可以使用右上角的頁碼，瀏覽 CA 的多個頁面。每個 CA 會佔用一列，並針對每個 CA 顯示下列部分或全部資料欄：

- 主旨 — CA 的辨別名稱資訊摘要。
- Id — CA 的 32 位元組十六進位唯一識別碼。
- 狀態 — CA 狀態。可能的值為 [建立]、[擱置憑證]、[作用中]、[已刪除]、[停用]、[過期] 和
- 類型 — CA 的類型。可能的值為「根」和「從屬」。
- 模式 — CA 的模式。可能的值為「一般用途」(發行可以設定任何到期日的憑證) 和短期憑證 (發行有效期上限為七天的憑證)。在某些情況下，短的有效期可以取代撤銷機制。預設值為「一般用途」。
- 擁有者 — 擁有 CA 的 AWS 帳號。這可能是您的帳戶或已委派 CA 管理權限給您的帳戶。
- 金鑰演算法 — CA 支援的公開金鑰演算法。可能的值包括 RSA、RSA、普利梅 256 v1 和
- 簽署演算法 — CA 用來簽署憑證要求的演算法。(請勿與發行憑證時用來簽署憑證的`SigningAlgorithm`參數混淆。) 可能的值是 SHA256WITHECDSA、SHA384WITHECDSA、SHA512WITHECDSA、SHA256WITHRSA、SHA384WIT 和渴望。SHA512WITHRSA

> **Note**
>
> 您可以選擇主控台右上角的設定圖示，自訂要顯示的欄以及其他設定。

若要使用列出可用的 CA AWS CLI

使用命list-certificate-authorities令列出可用的 CA，如下列範例所示：

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

此命令會傳回與以下內容相似的資訊：

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

檢視私人 CA

您可以使用 ACM 主控台或檢視有關私有 CA 的詳細中繼資料，並視需要變更數個值。 AWS CLI 如需更新 CA 的詳細資訊，請參閱更新您的私有 CA。

在主控台中檢視 CA 詳細資訊

1. 登入您的 AWS 帳戶，然後在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台。
2. 檢閱私人憑證授權單位清單。您可以使用右上角的頁碼，瀏覽 CA 的多個頁面。
3. 若要顯示所列 CA 的詳細中繼資料，請依照您要檢查的 CA 選擇選項按鈕。這會開啟包含下列索引標籤檢視的詳細資料窗格：

   - 主旨標籤 — CA 辨別名稱的相關資訊。如需詳細資訊，請參閱 主體辨別名稱選項。顯示的欄位包括：
     - 主旨 — 提供的名稱資訊欄位摘要
     - 組織 (O) — 例如，公司名稱
     - 組織單位 (OU) — 例如，公司內的部門
     - 國家名稱 (C) — 兩個字母的國家代碼
     - 州或省名稱 — 州或省的全名
     - 地區名稱 — 城市的名稱
     - 通用名稱 (CN) — 用於識別 CA 的人類可讀字串。
   - CA 憑證索引標籤 — CA 憑證有效性的相關資訊
     - 有效期限 — 直到 CA 憑證有效的日期和時間
     - 到期日 — 到期前的天數
   - 撤銷組態索引標籤 — 您目前對憑證撤銷選項的選擇。選擇編輯進行更新。
     - 憑證撤銷清單 (CRL) 發佈 — 已啟用或停用的狀態
     - 線上憑證狀態通訊協定 (OCSP) — 啟用或停用的狀態
   - [權限] 索引標籤 — 您目前透過 AWS Certificate Manager (ACM) 選取此 CA 的憑證續約權限。選擇編輯進行更新。
   - ACM 續約授權 — 授權或未經授權的狀態
   - [標籤] 索引標籤 — 您目前為此 CA 指派的可自訂標籤。選擇管理要更新的標籤。
   - 資源共用標籤 — 您目前透過 AWS Resource Access Manager (RAM) 指定此 CA 的資源共用率。選擇管理要更新的資源共用。

- 名稱 — 資源共用的名稱
- 狀態 — 資源共用的狀態

4. 選擇要檢查的 CA 的 ID 欄位，以開啟 [一般] 窗格。CA 的 32 位元組十六進位唯一識別碼會出現在頂端。窗格提供下列其他資訊：

   - 狀態 — CA 狀態。可能的值為 [建立]、[擱置憑證]、[作用中]、[已刪除]、[停用]、[過期] 和
   - ARN — CA 的 Amazon 資源名稱。
   - 擁有者 — 擁有 CA 的 AWS 帳號。這可能是您的帳戶 (S elf) 或已委派 CA 管理權限給您的帳戶。
   - CA 類型 — CA 的類型。可能的值為「根」和「從屬」。
   - 建立時間 — 建立 CA 的日期和時間。
   - 到期日 — CA 憑證到期的日期和時間。
   - 模式 — CA 的模式。可能的值為一般用途 (可以設定任何到期日的憑證) 和短期憑證 (有效期上限為七天的憑證)。在某些情況下，短的有效期可以取代撤銷機制。預設值為「一般用途」。
   - 金鑰演算法 — CA 支援的公開金鑰演算法。可能的值包括 RSA 2048、RSA 4096、ECDSA P2567 和 ECDSA P384。
   - 簽署演算法 — CA 用來簽署憑證要求的演算法。(請勿與發行憑證時用來簽署憑證的`SigningAlgorithm`參數混淆。) 可能的值包括 SHA256 ECDSA, SHA384 埃克迪安局, SHA512 埃迪安局, SHA256 RSA, SHA384 RSA, 和渴望 RSA SHA512
   - 金鑰儲存安全標準 — 符合聯邦資訊處理標準的等級。可能的值為 FIPS 140-2 等級 3 或更高，以及 FIPS 140- 2 等級 3 或更高。此參數因「 AWS 區域」而異。

若要檢視和修改 CA 詳細資訊，請使用 AWS CLI

使用中的describe-certificate-authority命令 AWS CLI 來顯示 CA 的詳細資訊，如下列命令所示：

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

此命令會傳回與以下內容相似的資訊：

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

如需有關從命令列更新私有 CA 的資訊，請參閱更新一個 CA。

管理私有 CA 的標籤

標籤是做為中繼資料的單字或片語，用於識別和組織您的 AWS 資源。每個標籤皆包含鍵與值。您可以使用 AWS 私有 CA 主控台、 AWS Command Line Interface (AWS CLI) 或 PCA API 來新增、檢視或移除私有 CA 的標籤。

您可以隨時為私人 CA 新增或移除自訂標籤。例如，您可以使用索引鍵值配對 (例如`Environment=Prod`或) `Environment=Beta` 來標記私有 CA，以識別 CA 適用的環境。如需詳細資訊，請參閱建立私有 CA。

> **Note**
>
> 若要在建立程序期間將標籤附加至私有 CA，CA 管理員必須先將內嵌 IAM 政策與`CreateCertificateAuthority`動作建立關聯，並明確允許標記。如需詳細資訊，請參閱 Tag-on-create：建立時將標籤附加至 CA。

其他 AWS 資源也支援標記。您可以將相同的標籤指派給不同的資源，以指出這些資源是相關的。例如，您可以將標籤指派給 CA、Elastic Load Balancing 負載平衡器及其他相關資

源。Website=example.com如需標記 AWS 資源的詳細資訊，請參閱 Amazon EC2 使用者指南中的標記您的 Amazon EC2 資源。

下列基本限制適用於 AWS 私有 CA 標籤：

- 每個私有 CA 的標籤數上限為 50。
- 標籤鍵的長度上限為 128 個字元。
- 標籤值的長度上限為 256 個字元。
- 標籤鍵和值可以包含下列字元：A-Z、a-z 和 .:+= @_%- (連字號)。
- 標籤金鑰與值皆區分大小寫。
- aws: 和 rds: 字首是保留給 AWS 使用的字詞；您無法新增、編輯或刪除鍵是以 aws: 或 rds: 開頭的標籤。以aws:及rds:不會計入tags-per-resource 配額的預設標記。
- 如果您打算在多個服務和資源中使用標記結構描述，請記住，其他服務對於允許的字元可能有不同的限制。請參閱文件以了解該服務。
- AWS 私有 CA 標籤無法在中的 Resource Groups 和標籤編輯器中使用 AWS Management Console。

您可以從 AWS 私有 CA 主控台、AWS Command Line Interface (AWS CLI) 或 AWS 私有 CA API 來標記私有 CA。

標記私有 CA (主控台)

1. 登入您的 AWS 帳戶，然後在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台。
2. 在 [私人憑證授權單位] 頁面上，從清單中選擇您的私有 CA。
3. 在清單下方的詳細資料區域中，選擇「標籤」標籤。顯示現有標籤的清單。
4. 選擇管理標籤。
5. 選擇 Add new tag (新增標籤)。
6. 輸入鍵值組。
7. 選擇儲存。

標記私有 CA (AWS CLI)

使用命tag-certificate-authority令將標籤新增至您的私有 CA。

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

使用 list-tags 命令來列出私有 CA 的標籤。

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

使用untag-certificate-authority命令從私有 CA 移除標籤。

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

更新您的私有 CA

您可以在建立私有 CA 之後更新其狀態或變更其撤銷組態。本主題提供 CA 狀態和 CA 生命週期的詳細資料，以及 CA 的主控台和 CLI 更新範例。

正在更新 CA 狀態

由使用者動作或某些情況下的服務動作所產生的 AWS 私有 CA 結果所管理的 CA 狀態。例如，CA 狀態會在到期時變更。CA 管理員可使用的狀態選項會因 CA 目前的狀態而有所不同。

AWS 私有 CA 可以報告下列狀態值。此表格顯示每個狀態中可用的 CA 功能。

> **Note**
>
> 對於除了DELETED和以外的所有狀態值FAILED，我們會向您收取 CA 的費用。

Status	頒發憑證	使用 OCSP 驗證憑證	產生 CRL	產生稽核	您可以更新 CA 證書	憑證可以撤銷	我們會向您收取 CA 費用
CREATING— 正在建立 CA。	否	否	否	否	否	否	是
PENDING_CERTIFICATE — CA 已建立，需要憑證才能運作。 *	否	否	否	否	否	否	是
ACTIVE	是	是	是	是	是	是	是
DISABLED— 您已手動停用 CA。	否	是	是	是	否	是	是
EXPIRED— CA 憑證已過期。 **	否	否	否	否	是	否	是
FAILED	動CreateCertificateAuthority 作失敗。這可能是因為網路中斷、後端 AWS 故障或其他錯誤而發生。失敗的 CA 無法復原。請刪除 CA 並建立新的 CA。						否
DELETED	您的 CA 在恢復期內，可以有 7-30 天的時間。在此期間之後，CA 將會永久刪除。 • 如果您對狀態為 DELETED 且憑證已過期的 CA 呼叫 RestoreCertificateAuthority API，則 CA 將會設為 EXPIRED。 • 如需刪除 CA 的詳細資訊，請參閱 刪除您的私有 CA。						否

* 若要完成啟用，您需要產生 CSR、從 CA 取得已簽署的 CA 憑證，然後將憑證匯入 AWS 私有 CA。CSR 可以提交至您的新 CA (用於自我簽署)，或提交至內部部署根目錄或從屬 CA。如需詳細資訊，請參閱 建立和安裝 CA 憑證。

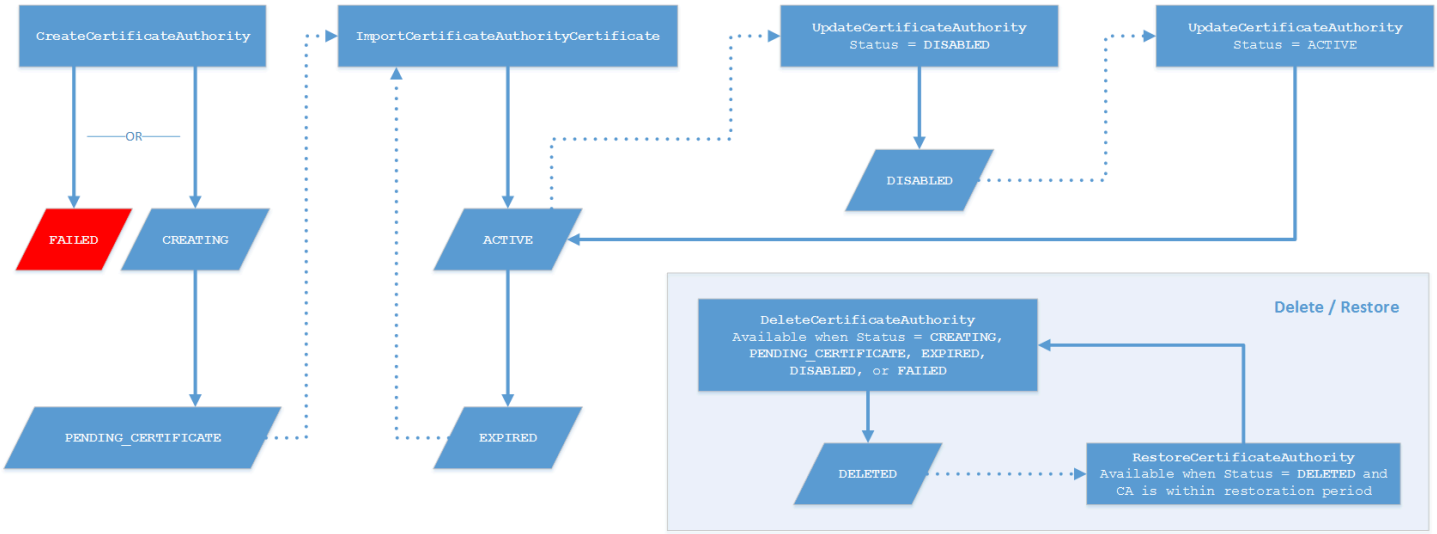
** 您無法直接變更過期 CA 的狀態。如果您匯入 CA 的新憑證，請將狀態 AWS 私有 CA 重設為，ACTIVE除非將狀態設定為憑證到期DISABLED之前。

有關過期 CA 憑證的其他注意事項：

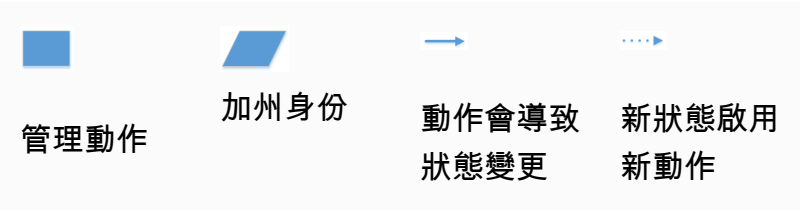
- CA 憑證不會自動更新。如需透過自動化續約的資訊 AWS Certificate Manager，請參閱將憑證續訂權限指派給 ACM。
- 如果您嘗試使用過期的 CA 發行新憑證，則 IssueCertificate API 會傳回 InvalidStateException。過期的根 CA 必須先自我簽署新的根 CA 憑證，才能發行新的次級憑證。
- 如果 CA 憑證已過期，The ListCertificateAuthorities 和 DescribeCertificateAuthority API 便會傳回 EXPIRED 狀態，無論 CA 狀態是設為 ACTIVE 或是 DISABLED。但是，如果過期的 CA 已設為 DELETED，則狀態會傳回 DELETED。
- UpdateCertificateAuthority API 無法更新已過期 CA 的狀態。
- RevokeCertificateAPI 無法用於撤銷任何過期的憑證，包括 CA 憑證。

CA 狀態和 CA 生命週期

下圖會將 CA 的生命週期做為 CA 狀態與管理動作的互動顯示。



圖表鍵



在圖表頂端，管理動作會透過 AWS 私有 CA 主控台、CLI 或 API 套用。這些動作會帶領 CA 經歷建立、啟用、過期和續約。CA 狀態會在回應中變更為手動動作或自動更新 (以實線顯示)。在大多數的情況下，新的狀態會產生可讓 CA 管理員套用的新可能動作 (以虛線顯示)。右下方的內凹顯示可能的狀態值，允許刪除和還原動作。

更新 CA（控制台）

下列程序顯示如何使用更新現有 CA 組態 AWS Management Console。

更新 CA 狀態（控制台）

在此範例中，已啟用 CA 的狀態會變更為停用。

若要更新 CA 的狀態

1. 登入您的 AWS 帳戶，然後在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台
2. 在 [私人憑證授權單位] 頁面上，從清單中選擇目前作用中的私人 CA。
3. 在 [動作] 功能表上，選擇 [停用] 以停用私有 CA。

更新 CA 的撤銷配置（控制台）

您可以更新私有 CA 的撤銷組態，例如新增或移除 OCSP 或 CRL 支援，或修改其設定。

> **Note**
>
> CA 的撤銷組態變更不會影響已發行的憑證。若要讓受管理的撤銷運作，必須重新發行較舊的憑證。

對於 OCSP，您可以變更下列設定：

- 啟用或停用 OCSP。
- 啟用或停用自訂 OCSP 完整網域名稱 (FQDN)。
- 變更 FQDN。

對於 CRL，您可以變更下列任何設定：

- 私有 CA 是否會產生憑證撤銷清單 (CRL)
- CRL 到期的天數。請注意， AWS 私有 CA 開始嘗試在您指定的天數 ½ 重新產生 CRL。
- 儲存 CRL 所在的 Amazon S3 儲存貯體的名稱。
- 用於在公開檢視中隱藏 Amazon S3 儲存貯體名稱的別名。

> **Important**
>
> 變更上述任何參數都可能會產生負面影響。範例包括停用 CRL 產生、變更有效期，或在您將私有 CA 置於生產環境之後變更 S3 儲存貯體。此類變更可能會破壞取決於 CRL 和目前 CRL 組態的現有憑證。您可以安全地變更別名，只要舊別名仍連結到正確的儲存貯體。

更新撤銷設定

1. 登入您的 AWS 帳戶，然後在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台。
2. 在 [私人憑證授權單位] 頁面上，從清單中選擇私有 CA。這會開啟 CA 的詳細資料面板。
3. 選擇撤銷組態索引標籤，然後選擇編輯。
4. 在憑證撤銷選項下，會顯示兩個選項：
   - 啟動 CRL 發佈
   - 開啟 OCSP

   您可以為 CA 設定這些撤銷機制，也可以設定這兩種撤銷機制。雖然選擇性，但建議使用管理撤銷作為最佳作法。在完成此步驟之前，請設定憑證撤銷方法參閱以取得有關每種方法的優點、可能需要的初步設定以及其他撤銷功能的資訊。

若要設定 CRL

1. 選取 [啟動 CRL 發佈]。
2. 若要為您的 CRL 項目建立 Amazon S3 儲存貯體，請選取建立新的 S3 儲存貯體。提供唯一的值區名稱。(您不需要包含指向儲存貯體的路徑。) 否則，請不要選取此選項，然後從 S3 儲存貯體名稱清單中選擇現有儲存貯體。

   如果您建立新值區，請建 AWS 私有 CA 立並附加必要的存取原則至該值區。如果您決定使用現有值區，則必須先附加存取原則，才能開始產生 CRL。使用中所述的其中一個策略模式Amazon S3 中 CRL 的訪問政策 。如需附加政策的相關資訊，請參閱使用 Amazon S3 主控台新增儲存貯體政策。

   > **Note**
   >
   > 當您使用 AWS 私有 CA 主控台時，如果同時符合下列兩種情況，嘗試建立 CA 就會失敗：
   >
   > - 您正在 Amazon S3 儲存貯體或帳戶上強制執行區塊公開存取設定。
   > - 您 AWS 私有 CA 要求自動建立 Amazon S3 儲存貯體。
   >
   > 在此情況下，主控台預設會嘗試建立可公開存取的儲存貯體，而 Amazon S3 拒絕此動作。如果發生這種情況，請檢查您的 Amazon S3 設定。如需詳細資訊，請參閱封鎖 Amazon S3 儲存的公開存取。

3. 展開 Advanced (進階) 以取得其他組態選項。
   - 新增自訂 CRL 名稱，為您的 Amazon S3 儲存貯體建立別名。此名稱包含在由 RFC 5280 定義的「CRL 發佈點」延伸模組中由 CA 所發行的憑證中。
   - 輸入您的 CRL 將保持有效的天數。預設值為 7 天。對於在線 CRL，通常有效期為 2-7 天。AWS 私有 CA 嘗試在指定期間的中點重新產生 CRL。
4. 選擇「完成後保存更改」。

若要設定 OCSP

1. 在 [憑證撤銷] 頁面上，選擇 [開啟 OCSP]。
2. (選擇性) 在「自訂 OCSP 端點」欄位中，為您的 OCSP 端點提供完整網域名稱 (FQDN)。

當您在此欄位中提供 FQDN 時，會將 FQDN AWS 私有 CA 插入每個已發行憑證的「授權單位資訊存取」延伸模組，以取代 OCSP 回應程式的預設 URL。 AWS 當端點收到包含自訂 FQDN 的憑證時，會查詢該位址是否有 OCSP 回應。為了使此機制正常運作，您需要採取兩個額外的動作：

- 使用 Proxy 伺服器將到達自訂 FQDN 的流量轉送給 AWS OCSP 回應程式。
- 將對應的 CNAME 記錄新增至您的 DNS 資料庫。

> **Tip**
>
> 如需使用自訂 CNAME 實作完整 OCSP 解決方案的詳細資訊，請參閱。設定 AWS 私有 CA OCSP 的自訂網址

例如，以下是自訂 OCSP 的 CNAME 記錄，就像它會出現在 Amazon 路線 53 中一樣。

記錄名稱	Type	路由政策	微分器	值/將流量路由到
替代例子	CNAME	簡便	-	代理例子

> **Note**
>
> CNAME 的值不得包含通訊協定前置詞，例如「http://」或「https://」。

3. 選擇「完成後保存更改」。

更新一個 CA

下列程序顯示如何使用更新現有 CA 的狀態和撤銷組態。 AWS CLI

> **Note**
>
> CA 的撤銷組態變更不會影響已發行的憑證。若要讓受管理的撤銷運作，必須重新發行較舊的憑證。

若要更新您的私有 CA (AWS CLI) 狀態

使用 update-certificate-authority 命令。

當您擁有要設定為狀態的現有 CA 時，此功DISABLED能非常有用ACTIVE。若要開始，請使用下列命令確認 CA 的初始狀態。

```
$ 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
```

這會產生類似下列的輸出。

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

下列命令會將私有 CA 的狀態設定為ACTIVE。只有在 CA 上安裝了有效的憑證時，才能執行此操作。

```
$ 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"
```

檢查 CA 的新狀態。

```
$ 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
```

狀態現在會顯示為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
            }
        }
    }
}
```

在某些情況下，您可能擁有未設定撤銷機制的作用中 CA。如果您要開始使用憑證撤銷清單 (CRL)，請使用下列程序。

若要將 CRL 新增至現有 CA ()AWS CLI

1. 使用下面的命令來檢查 CA 的當前狀態。

```
$ 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
```

輸出會確認 CA 具有狀態，ACTIVE但未設定為使用 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. 使用名稱建立並儲存檔案，例如`revoke_config.txt`定義 CRL 組態參數。

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

> **Note**
>
> 更新 Matter 裝置證明 CA 以啟用 CRL 時，您必須將其設定為省略已發行憑證中的 CDP 延伸，以協助符合目前的 Matty 標準。若要這麼做，請定義 CRL 組態參數，如下圖所示：
>
> ```
> {
> ```

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

3. 使用update-certificate-authority命令和撤銷配置文件來更新 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. 再次檢查 CA 的狀態。

```
$ 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
```

輸出會確認 CA 現在已設定為使用 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
            }
        }
    }
}
```

在某些情況下，您可能想要新增 OCSP 撤銷支援，而不是像先前程序一樣啟用 CRL。在這種情況下，請使用以下步驟。

若要將 OCSP 支援新增至現有的 CA ()AWS CLI

1. 使用名稱建立並儲存檔案，例如`revoke_config.txt`定義 OCSP 參數。

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

2. 使用update-certificate-authority命令和撤銷配置文件來更新 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. 再次檢查 CA 的狀態。

```
$ 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
```

輸出會確認 CA 現在已設定為使用 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**
>
> 您也可以在 CA 上設定 CRL 和 OCSP 支援。

刪除您的私有 CA

您可以從 AWS Management Console 或 AWS CLI 永久刪除私人 CA。例如，您可能想要刪除某 CA，將其取代成具有新私有金鑰的新 CA。若要安全地刪除 CA，請遵循下列步驟：

1. 建立替換 CA。
2. 一旦新的私有 CA 在生產環境中，請停用舊的私有 CA，但不要立即刪除。
3. 請將舊 CA 保持停用，直到其發行的所有憑證皆過期為止。
4. 刪除舊 CA。

AWS 私有 CA 在處理刪除要求之前，不會檢查所有已發行的憑證是否已過期。您可以產生稽核報告來判斷哪些憑證已過期。雖然 CA 已停用，您仍可撤銷憑證，但無法發行新的憑證。

如果您必須在所有發行的憑證過期前刪除私有 CA，我們建議您撤銷 CA 憑證。CA 憑證將會列於上層 CA 的 CRL 中，私有 CA 將不受用戶端信任。

> **Important**
>
> 私有 CA 處於 PENDING_CERTIFICATE、CREATING、EXPIRED、DISABLED 或 FAILED 狀態時，可以刪除。若要刪除處於 ACTIVE 狀態的 CA，您必須先進行停用，否則刪除請求會導致例外狀況。如果您要刪除PENDING_CERTIFICATE或DISABLED狀態的私人 CA，可以將還原期間長度設定為 7-30 天，預設值為 30。在此期間，狀態會設為 DELETED 且 CA 可進行還原。處於CREATING或FAILED狀態時刪除的私有 CA 沒有指派的還原期間，因此無法還原。如需詳細資訊，請參閱 還原私有 CA。
>
> 私有 CA 刪除後，您不需付費。不過，如果還原已刪除的 CA，您需支付刪除到還原這段期間的費用。如需詳細資訊，請參閱 定價。

刪除私有 CA (主控台)

1. 登入您的 AWS 帳戶，然後在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台。

2. 在 [私人憑證授權單位] 頁面上，從清單中選擇您的私有 CA。
3. 如果您的 CA 處於此狀ACTIVE態，您必須先將其停用。在 Actions (動作) 選單上，選擇 Disable (停用)。出現提示時，選擇我了解風險，繼續。
4. 對於非處於ACTIVE狀態的 CA，請選擇動作 > 刪除。
5. 如果您的 CA 處於DISABLED、或PENDING_CERTIFICATE狀態EXPIRED，[刪除 CA] 頁面可讓您指定 7-30 天的還原期間。如果您的私有 CA 不在這些狀態之一，則稍後將無法還原，並且刪除是永久性的。
6. 選擇刪除。
7. 如果您確定要刪除私有 CA，請在系統提示時選擇 Permanently delete (永久刪除)。私有 CA 的狀態會變更為 DELETED。不過，您可以在還原期間結束之前還原私有 CA。若要檢查DELETED狀態中私有 CA 的還原期間，請呼叫DescribeCerticateAuthority或 ListCertificateAuthoritiesAPI 作業。

刪除私有 CA (AWS CLI)

使用delete-certificate-authority命令刪除私有 CA。

```
$ 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
```

還原私有 CA

您可以還原已刪除的私有 CA，只要 CA 在刪除後仍處於您指定的還原期間內即可。恢復期為 7-30 天。此期間結束後，私有 CA 將永久刪除。如需詳細資訊，請參閱 刪除您的私有 CA。您不能還原已永久刪除的私有 CA。

> **Note**
>
> 私有 CA 刪除後，您不需付費。不過，如果還原已刪除的 CA，您需支付刪除到還原這段期間的費用。如需詳細資訊，請參閱 定價。

還原私有 CA (主控台)

您可以使用 AWS Management Console 來還原私有 CA。

若要還原私有 CA (主控台)

1. 登入您的 AWS 帳戶，然後在 https://console.aws.amazon.com/acm-pca/home 開啟 AWS 私有 CA 主控台。
2. 在 [私人憑證授權單位] 頁面上，從清單中選擇已刪除的私有 CA。
3. 在操作功能表上，選擇 Restore (還原)。
4. 在 [還原 CA] 頁面上，再次選擇 [還原]。
5. 如果成功，私有 CA 的狀態會設為其刪除前狀態。再次選擇 [動作]、[啟用] 和 [啟用]，將其狀態變更為ACTIVE。如果私有 CA 在刪除時處於 PENDING_CERTIFICATE 狀態，您必須先匯入 CA 憑證到私有 CA，才能進行啟用。

還原私有 CA (AWS CLI)

使用此restore-certificate-authority命令可還原處於DELETED狀態的已刪除私有 CA。以下步驟討論刪除、還原，然後重新啟用私有 CA 所需的整個程序。

若要刪除、還原和重新啟用私有 CA (AWS CLI)

1. 刪除私有 CA。

   執行delete-certificate-authority命令以刪除私有 CA。如果私有 CA 的狀態為DISABLED或PENDING_CERTIFICATE，您可以設定--permanent-deletion-time-in-days參數，以指定私有 CA 的還原期間為 7 -30 天。如果您未指定還原期間，預設為 30 天。如果成功，這個命令會將私有 CA 的狀態設為 DELETED。

   > **Note**
   >
   > 若要可供還原，在刪除當時的私有 CA 狀態必須為 DISABLED 或 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. 還原私有 CA。

執行restore-certificate-authority命令以還原私有 CA。您必須在您以 delete-certificate-authority 命令設定的還原期間過期前執行命令。如果成功，此命令會將私有 CA 的狀態設為其刪除前狀態。

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

3. 將私有 CA 設為 ACTIVE。

   執行update-certificate-authority命令，將私有 CA 的狀態變更為ACTIVE。

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

憑證管理

建立並啟動私有憑證授權單位 (CA) 並設定其存取權之後，您或您的授權使用者即可執行本節中討論的工作。如果您尚未為 CA 設定 AWS Identity and Access Management (IAM) 政策，可以在本指南的 [Identity and Access Management] 一節中進一步了解如何設定這些政策。如需在單一帳戶和跨帳戶案例中設定 CA 存取權的相關資訊，請參閱。控制對私有 CA 的存取

主題

- 簽發私人終端實體證書
- 擷取私人憑證
- 列出私有憑證
- 匯出私人憑證及其秘密金鑰
- 撤銷私人憑證
- 自動匯出更新的憑證
- 瞭解憑證範本

簽發私人終端實體證書

使用私有 CA 後，您可以從 AWS Certificate Manager (ACM) 或. AWS 私有 CA 下表比較了這兩種服務的功能。

功能	ACM	AWS 私有 CA
頒發最終實體證書	✓（使用RequestCertificate或控制台）	✓ (使用 IssueCertificate)
與負載平衡器和面向網際網路的服務建立關聯 AWS	✓	不支援
受管理憑證續約	✓	透過 AC M 間接支援
主控台支援	✓	不支援
API 支援	✓	✓
CLI 支援	✓	✓

AWS 私有 CA建立憑證時，它會遵循指定憑證類型和路徑長度的範本。如果沒有範本 ARN 提供給建立憑證的 API 或 CLI 陳述式，則預設會套用 EndEntityCertificate/V1 範本。如需可用憑證範本的詳細資訊，請參閱瞭解憑證範本。

雖然 ACM 憑證是以公眾信任為基礎設計，AWS 私有 CA但可滿足您私有 PKI 的需求。因此，您可以使用 ACM 不允許的方式使用 AWS 私有 CA API 和 CLI 來設定憑證。這些索引標籤包括以下項目：

- 建立具有任何主體名稱的憑證。
- 使用任何支援的私密金鑰演算法和金鑰長度。
- 使用任何支援的簽署演算法。
- 指定私有 CA 和私有憑證的任何有效期間。

使用建立私有 TLS 憑證之後AWS 私有 CA，您可以將其匯入 ACM，並將其與支援的AWS服務搭配使用。

> **Note**
>
> 使用issue-certificate命令或 IssueCertificateAPI 動作使用以下程序建立的憑證無法直接匯出以供外部使用AWS。不過，您可以使用私有 CA 簽署透過 ACM 發行的憑證，而且這些憑證可以連同其密鑰一起匯出。如需詳細資訊，請參閱 ACM 使用指南中的要求私人憑證和匯出私有憑證。

簽發標準證書（AWS CLI）

您可以使用 AWS 私有 CA CLI 命令問題憑證或 API 動作IssueCertificate來要求最終實體憑證。此命令需要您要使用的私有 CA 的 Amazon Resource Name (ARN) 來發行憑證。您也必須使用 OpenSSL 之類的程式產生憑證簽署要求 (CSR)。

如果您使用 AWS 私有 CA API 或AWS CLI發行私有憑證，則憑證不受管理，表示您無法使用 ACM 主控台、ACM CLI 或 ACM API 來檢視或匯出憑證，而且憑證不會自動更新。不過，您可以使用 PCA 取得憑證命令來擷取憑證詳細資料，如果您擁有 CA，則可以建立稽核報告。

建立憑證時的考量

- 根據 RFC 5280 規定，您提供的網域名稱 (技術上是一般名稱) 的長度不得超過 64 個八位元組 (字元)，包括句點。若要新增較長的網域名稱，請在「主旨替代名稱」欄位中指定名稱，此欄位最多支援 253 個八位元組的名稱。

- 如果您使用的是 1.6.3 AWS CLI 版或更新版本，請在指定以 base64 編碼的輸入檔案 (例如 CSR) `fileb://` 時使用前置詞。這樣可以確保正確AWS 私有 CA解析數據。

下列 OpenSSL 指令會為憑證產生 CSR 和私密金鑰：

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

您可以按照以下方式檢查 CSR 的內容：

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

產生的輸出應類似於下列縮寫範例：

```
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
```

下列命令會建立憑證。由於未指定範本，因此依預設會發出基礎終端實體憑證。

```
$ 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"
```

傳回已發行憑證的 ARN：

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

> **Note**
>
> AWS 私有 CA收到issue-certificate指令時，立即傳回含序號的 ARN。不過，憑證處理會以非同步方式進行，而且仍可能失敗。如果發生這種情況，使用新 ARN 的get-certificate命令也將失敗。

使用 API 傳遞範本發行具有自訂主體名稱的憑證

在此範例中，會發行包含自訂主體名稱元素的憑證。除了提供類似於中的 CSR 之外簽發標準證書（AWS CLI），您還可以將兩個額外引數傳遞至issue-certificate命令：APIPassThrough範本的 ARN，以及指定自訂屬性及其物件識別碼 (OID) 的 JSON 組態檔。您不能與一起使用CustomAttributes。但是，您可以將標準 OID 作為StandardAttributes中的CustomAttributes一部分傳遞。下表中列出了預設的主體名稱 OID (來自 RFC 4519 和全域 OID 參照資料庫的資訊)：

主旨名稱	縮寫	物件識別碼
countryName	c	2.5.4.6
commonName	cn	2.5.4.3
DN 限定詞 [辨別名稱限定詞]		2.5.4.46

主旨名稱	縮寫	物件識別碼
產生限定元		2.5.4.44
givenName		2.5.4.42
（字的）起首字母		2.5.4.43
局部性	l	2.5.4.7
組織名稱	o	2.5.4.10
organizationalUnitName	ou	2.5.4.11
雅號		2.5.4.65
序號		2.5.4.5
聖 [州]		2.5.4.8
姓	sn	2.5.4.4
標題		2.5.4.12
網域元件	dc	0.9.2342.19200300.100.1.25
userid		0.9.2342.19200300.100.1.1

範例組態檔案`api_passthrough_config.txt`包含下列程式碼：

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

使用下列命令來發行憑證：

```
$ 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
```

傳回已發行憑證的 ARN：

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

在本機擷取憑證，如下所示：

```
$ 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
```

您可以使用 OpenSSL 來檢查憑證的內容：

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

> **Note**
>
> 您也可以建立私有 CA，將自訂屬性傳遞至其發行的每個憑證。

使用 API 傳遞範本發行具有自訂擴充功能的憑證

在此範例中，會發行包含自訂擴充功能的憑證。為此，您需要將三個參數傳遞給issue-certificate命令：APIPassThrough 範本的 ARN，以及指定自訂擴充功能的 JSON 設定檔，以及如中所示的 CSR。簽發標準證書（AWS CLI）

範例組態檔案api_passthrough_config.txt包含下列程式碼：

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

定制證書頒發如下：

```
$ 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
```

傳回已發行憑證的 ARN：

```
{
```

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

在本機擷取憑證，如下所示：

```
$ 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
```

您可以使用 OpenSSL 來檢查憑證的內容：

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

擷取私人憑證

您可以使用 AWS 私有 CA API 和 AWS CLI 來發行私有憑證。如果您要這麼做，則可以使用 AWS CLI 或 AWS 私有 CA API 來擷取該憑證。如果您使用 ACM 建立私有 CA 並要求憑證，則必須使用 ACM 匯出憑證和加密的私密金鑰。如需詳細資訊，請參閱匯出私有憑證。

若要擷取最終實體憑證

使用 [取得憑證] AWS CLI 命令擷取私有終端實體憑證。您也可以使用 GetCertificateAPI 操作。我們建議使用 jq（類似 sed 的解析器）格式化輸出。

> **Note**
>
> 如果您想要撤銷憑證，可以使用 get-certificate 命令以十六進制格式擷取序號。您也可以建立稽核報告以擷取十六進位序號。如需詳細資訊，請參閱 將稽核報告與您的私有 CA 搭配使用。

```
$ 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'
```

此命令會以下列標準格式輸出憑證和憑證鏈結。

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

擷取 CA 憑證

您可以使用 AWS 私有 CA API 和 AWS CLI 為私有 CA 擷取憑證授權機構 (CA) 憑證。執行 get-certificate-authority-certificate 命令。您也可以呼叫 GetCertificateAuthorityCertificate 操作。我們建議使用 jq（類似 sed 的解析器）格式化輸出。

```
$ 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'
```

此指令會以下列標準格式輸出 CA 憑證。

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

列出私有憑證

若要列出私有憑證，請產生稽核報告、從 S3 儲存貯體擷取該憑證，然後視需要剖析報告內容。如需有關建立AWS 私有 CA稽核報告的資訊，請參閱將稽核報告與您的私有 CA 搭配使用。如需從 S3 儲存貯體擷取物件的相關資訊，請參閱 Amazon 簡單儲存服務使用者指南中的下載物件。

下列範例說明建立稽核報告及剖析這些報表以取得有用資料的方法。結果以 JSON 格式化，並使用類似 sed 的解析器 jq 過濾數據。

1. 建立稽核報告。

下列指令會產生指定 CA 的稽核報告。

```
$ 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
```

成功時，命令會傳回新稽核報告的 ID 和位置。

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

2. 擷取稽核報告並格式化。

此命令會擷取稽核報告，在標準輸出中顯示其內容，並篩選結果以僅顯示在 2020-12-01 或之後發行的憑證。

```
$ 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")'
```

退回的項目如下所示：

```
{
   "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. 在本機儲存稽核報告。

如果要執行多個查詢，可以方便地將稽核報告儲存至本端檔案。

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

與以前相同的過濾器產生相同的輸出：

```
$ 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. 在日期範圍內查詢

您可以查詢日期範圍內發行的憑證，如下所示：

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

篩選的內容會顯示在標準輸出中：

```
{
   "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. 依照指定的範本搜尋憑證。

下列命令會使用範本 ARN 篩選報告內容：

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

輸出會顯示相符的憑證記錄：

```
{
   "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. 篩選已撤銷的憑證

若要尋找所有已撤銷的憑證，請使用下列命令：

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

已撤銷的憑證會顯示如下：

```
{
   "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. 使用規則運算式篩選。

下列命令會搜尋包含「leaf」字串的主旨名稱：

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

相符的憑證記錄會傳回如下：

```
{
   "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"
}
```

匯出私人憑證及其秘密金鑰

AWS 私有 CA無法直接匯出已簽署並發行的私有憑證。但是，您可以使AWS Certificate Manager用匯出此類憑證及其加密的秘密金鑰。然後，憑證完全可移植，以便在私有 PKI 中的任何位置部署。如需詳細資訊，請參閱AWS Certificate Manager使用指南中的匯出私人憑證。

為使用 ACM 主控台發行的私有憑證、ACM API 的`RequestCertificate`動作或中 ACM 區段中的request-certificate命令，AWS Certificate Manager提供受管理的續約功能，做為其他好處。AWS CLI如需有關續約的詳細資訊，請參閱在私有 PKI 中續約憑證。

撤銷私人憑證

您可以使用撤銷AWS 私有 CA憑證AWS CLI命令或 RevokeCertificateAPI 動作來撤銷憑證。例如，如果憑證的秘密金鑰遭到入侵，或其關聯的網域變成無效，則可能需要在排程到期前撤銷憑證。為了使撤銷生效，每當嘗試建立安全網路連線時，使用憑證的用戶端都需要檢查撤銷狀態的方法。

AWS 私有 CA提供兩種完全受控的機制來支援撤銷狀態檢查：線上憑證狀態通訊協定 (OCSP) 和憑證撤銷清單 (CRL)。使用 OCSP 時，用戶端會查詢即時傳回狀態的授權撤銷資料庫。使用 CRL 時，用戶端會根據定期下載和儲存的已撤銷憑證清單來檢查憑證。用戶端拒絕接受已撤銷的憑證。

OCSP 和 CRL 都取決於內嵌在憑證中的驗證資訊。因此，核發 CA 必須設定為在發行前支援其中一種或兩種機制。如需透過選取和實作管理撤銷的資訊AWS 私有 CA，請參閱設定憑證撤銷方法。

撤銷的憑證一律會記錄在AWS 私有 CA稽核報告中。

> **Note**
>
> 跨帳戶憑證簽發者需要其他權限才能撤銷他們所發行的憑證；否則，CA 擁有者必須執行撤銷。若要啟用跨帳戶發行者撤銷，CA 管理員必須建立兩個 RAM 共用，兩者都指向相同的 CA：
>
> 1. 與AWSRAMRevokeCertificateCertificateAuthority許可的共享。
> 2. 與AWSRAMDefaultPermissionCertificateAuthority許可的共享。

撤銷憑證

使用 RevokeCertificateAPI 動作或撤銷憑證命令撤銷私有 PKI 憑證。序號必須為十六進位格式。您可以透過呼叫 get-certificate 命令來擷取序號。revoke-certificate 命令不會傳回回應。

```
$ 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"
```

已撤銷的憑證和 OCSP

當您撤銷憑證時，OCSP 回應最多可能需要 60 分鐘才能反映新狀態。一般而言，OCSP 傾向於支援更快速的撤銷資訊散發，因為與用戶端可快取數天的 CRL 不同，OCSP 回應通常不會由用戶端快取。

CRL 中的已撤銷憑證

CRL 通常在憑證撤銷後約 30 分鐘更新。如果 CRL 更新因任何原因失敗，則每 15 分鐘AWS 私有 CA會進一步嘗試一次。

使用 Amazon CloudWatch，您可以為指標CRLGenerated和MisconfiguredCRLBucket. 如需詳細資訊，請參閱支援的 CloudWatch量度。如需建立及設定 CRL 的詳細資訊，請參閱 規劃憑證撤銷清單 (CRL)。

以下範例會示範憑證撤銷清單 (CRL) 中的已撤銷憑證。

```
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
```

稽核報告中的已撤銷憑證

所有憑證 (包括撤銷的憑證) 皆包含在私有 CA 的稽核報告中。以下範例會示範具有一個已發行憑證和一個已撤銷憑證的稽核報告。如需詳細資訊，請參閱 [將稽核報告與您的私有 CA 搭配使用]。

```
[
```

```
   {
      "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"
   }
]
```


自動匯出更新的憑證

當您使AWS 私有 CA用建立 CA 時，您可以將該 CA 匯入，AWS Certificate Manager並讓 ACM 管理憑證發行和續訂。如果要更新的憑證與整合式服務相關聯，則服務會順暢地套用新憑證。但是，如果憑證原本是匯出供 PKI 環境中的其他位置使用 (例如，在內部部署伺服器或設備中)，則您需要在續訂後再次匯出憑證。

如需使用 Amazon EventBridge 和 AWS Lambda 將 ACM 匯出程序自動化的範例解決方案，請參閱自動匯出續約憑證。

瞭解憑證範本

AWS 私有 CA使用配置模板來發行 CA 證書和最終實體證書。當您從 PCA 主控台發行 CA 憑證時，會自動套用適當的根或從屬 CA 憑證範本。

如果您使用 CLI 或 API 發行憑證，您可以提供範本 ARN 作為`IssueCertificate`動作的參數。如果不提供 ARN，則依預設會套用`EndEntityCertificate/V1`樣板。如需詳細資訊，請參閱IssueCertificateAPI 和問題憑證命令文件。

> **Note**
>
> AWS Certificate Manager(ACM) 對私有 CA 具有跨帳戶共用存取權的使用者可以發行 CA 簽署的受管理憑證。跨帳戶發行者受到以資源為基礎的政策的限制，並且只能存取下列最終實體憑證範本：
>
> - EndEntityCertificate/1
> - EndEntityClientAuthCertificate/1
> - EndEntityServerAuthCertificate/1
> - BlankEndEntityCertificate_API通過/V1
> - BlankEndEntityCertificate_APICSR通過/V1
> - 次級PathLen緩存證書 _ 0/V1
>
> 如需詳細資訊，請參閱 資源型政策。

主題

- 模板品種
- 操作模板順序
- 範本定義

模板品種

AWS 私有 CA支持四個品種的模板。

- 基本模板

預先定義的範本，其中不允許使用傳遞參數。

- 企業社會責任通訊模板

  允許 CSR 傳遞來擴充其對應基礎範本版本的範本。CSR 中用來發行憑證的擴充功能會複製到已發行的憑證。如果 CSR 包含與範本定義衝突的擴充值，範本定義將永遠具有較高的優先順序。如需優先順序的詳細資訊，請參閱操作模板順序。

- API 直通模板

  通過允許 API 傳遞擴展其相應基本模板版本的模板。要求憑證的實體可能不知道系統管理員或其他中繼系統所知道的動態值、可能無法在範本中定義，也可能無法在 CSR 中使用。不過，CA 管理員可以從其他資料來源 (例如 Active Directory) 擷取其他資訊，以完成要求。例如，如果機器不知道它屬於哪個組織單位，系統管理員可以查詢 Active Directory 中的資訊，並將資訊包含在 JSON 結構中，將其新增至憑證要求。

  `IssueCertificate`動作`ApiPassthrough`參數中的值會複製到已發行的憑證。如果`ApiPassthrough`參數包含與範本定義衝突的資訊，範本定義將永遠具有較高的優先順序。如需優先順序的詳細資訊，請參閱操作模板順序。

- 通過模板

  允許 API 和 CSR 傳遞來擴充其對應基礎範本版本的範本。用於發行憑證的 CSR 中的擴充功能會複製到已發行的憑證，而`IssueCertificate`動作`ApiPassthrough`參數中的值也會複製過來。在範本定義、API 傳遞值和 CSR 傳遞延伸模組發生衝突的情況下，範本定義具有最高優先順序，接著是 API 傳遞值，後面是 CSR 傳遞延伸模組。如需優先順序的詳細資訊，請參閱操作模板順序。

下表列出了所有支援的範本類型，以AWS 私有 CA及其定義的連結。

> **Note**
>
> 若要取得有關 GovCloud 區域中範本 ARN 的資訊，請參閱《AWS GovCloud (US)使用指南》AWS Private Certificate Authority中的〈〉。

基本模板

範本名稱	範本 ARN	憑證類型
CodeSigningCertificate/1	arn:aws:acm-pca:::template/CodeSigningCertificate/V1	程式碼簽署
EndEntityCertificate/1	arn:aws:acm-pca:::template/EndEntityCertificate/V1	終端實體
EndEntityClientAuthCertificate/1	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate/V1	終端實體
EndEntityServerAuthCertificate/1	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate/V1	終端實體
OCSP SigningCertificate	arn:aws:acm-pca:::template/OCSPSigningCertificate/V1	OCSP 簽署
RootCACertificate/V1	arn:aws:acm-pca:::template/RootCACertificate/V1	CA
次級PathLen緩存證書 _ 0/V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1	CA
次級PathLen緩存證書 _ 1/V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1/V1	CA

範本名稱	範本 ARN	憑證類型
次級PathLen緩存證書 _ 2/V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2/V1	CA
下屬緩PathLen存證書 _ 3/V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3/V1	CA

企業社會責任通訊模板

範本名稱	範本 ARN	憑證類型
BlankEndEntityCertificate_企業社會責任通過/V1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CSRPassthrough/V1	終端實體
BlankEndEntityCertificate_ _ 企業社會責任傳CriticalBasicConstraints遞 /V1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_CSRPassthrough/V1	終端實體
BlankSubordinate緩存 _ 0_ 企業社PathLen會責任傳遞 /V1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
BlankSubordinate緩存 _ 1_ 企業社PathLen會責任傳遞 /V1	arn:aws:acm-pca:::template/BlankSubordinateCACertifica	CA

範本名稱	範本 ARN	憑證類型
	te_PathLen1_CSRPas sthrough/V1	
BlankSubordinate緩存 _ 2_ 企業社 PathLen會責任傳遞 /V1	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen2_CSRPas sthrough/V1	CA
BlankSubordinate緩存 _ 3_ 企業社 PathLen會責任傳遞 /V1	arn:aws:acm-pca::: template/BlankSubo rdinateCACertifica te_PathLen3_CSRPas sthrough/V1	CA
CodeSigningCertificate_企業社會 責任通過/V1	arn:aws:acm-pca::: template/CodeSigni ngCertificate_CSRP assthrough/V1	程式碼簽署
EndEntityCertificate_企業社會責任 通過/V1	arn:aws:acm-pca::: template/EndEntity Certificate_CSRPas sthrough/V1	終端實體
EndEntityClientAuthCertificate_企 業社會責任通過/V1	arn:aws:acm-pca::: template/EndEntity ClientAuthCertific ate_CSRPassthrough/V1	終端實體
EndEntityServerAuthCertificate_企 業社會責任通過/V1	arn:aws:acm-pca::: template/EndEntity ServerAuthCertific ate_CSRPassthrough/V1	終端實體

範本名稱	範本 ARN	憑證類型
OCSP _ 企業社SigningCertificate 會責任傳遞 /V1	arn:aws:acm-pca:::template/OCSPSigningCertificate_CSRPassthrough/V1	OCSP 簽署
次級PathLen緩存證書 _ 0_ 企業社會責任傳遞 /V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_CSRPassthrough/V1	CA
次級PathLen緩存證書 _ 1_ 企業社會責任傳遞 /V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_CSRPassthrough/V1	CA
次級PathLen緩存證書 _ 2_ 企業社會責任傳遞 /V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_CSRPassthrough/V1	CA
次級PathLen緩存證書 _ 3_ 企業社會責任傳遞 /V1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_CSRPassthrough/V1	CA

API 直通模板

範本名稱	範本 ARN	憑證類型
BlankEndEntityCertificate_API通過/V1	arn:aws:acm-pca:::template/BlankEndE	終端實體

範本名稱	範本 ARN	憑證類型
	ntityCertificate_APIPassthrough/V1	
BlankEndEntityCertificate_ APCriticalBasicConstraints I通過/V1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APIPassthrough/V1	終端實體
CodeSigningCertificate_API通過/V1	arn:aws:acm-pca:::template/CodeSigningCertificate_APIPassthrough/V1	程式碼簽署
EndEntityCertificate_API通過/V1	arn:aws:acm-pca:::template/EndEntityCertificate_APIPassthrough/V1	終端實體
EndEntityClientAuthCertificate_API通過/V1	arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_APIPassthrough/V1	終端實體
EndEntityServerAuthCertificate_API通過/V1	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APIPassthrough/V1	終端實體
OCSP _API通行證/V SigningCertificate 1	arn:aws:acm-pca:::template/OCSPSigningCertificate_APIPassthrough/V1	OCSP 簽署

範本名稱	範本 ARN	憑證類型
根據快取 _ API 傳遞 /V1	arn:aws:acm-pca:::template/RootCACertificate_APIPassthrough/V1	CA
BlankRoot緩存_API通過/V1	arn:aws:acm-pca:::template/BlankRootCACertificate_APIPassthrough/V1	CA
BlankRoot緩存 _ 0_API通過/V PathLen 1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen0_APIPassthrough/V1	CA
BlankRoot緩存 _ 1_API通過/V PathLen 1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen1_APIPassthrough/V1	CA
BlankRoot緩存_2_API通過/V PathLen 1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen2_APIPassthrough/V1	CA
BlankRoot緩存_3_API通過/V PathLen 1	arn:aws:acm-pca:::template/BlankRootCACertificate_PathLen3_APIPassthrough/V1	CA
次級緩存證書 _ 0_API通過/V PathLen 1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APIPassthrough/V1	CA

範本名稱	範本 ARN	憑證類型
BlankSubordinate緩存 _ 0_API通過/V PathLen 1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APIPassthrough/V1	CA
次級緩存證書 _ 1_API通過/V PathLen 1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
BlankSubordinate緩存 _ 1_API通過/V PathLen 1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APIPassthrough/V1	CA
次級緩存證書 _ 2_API通過/V PathLen 1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
BlankSubordinate緩存_2_API通過/V PathLen 1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APIPassthrough/V1	CA
次級緩存證書 _ 3_API通過/V PathLen 1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APIPassthrough/V1	CA

範本名稱	範本 ARN	憑證類型
BlankSubordinate緩存_3_API通過/V PathLen 1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APIPassthrough/V1	CA

通過模板

範本名稱	範本 ARN	憑證類型
BlankEndEntityCertificate_APICSR通過/V1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_APICSRPassthrough/V1	終端實體
BlankEndEntityCertificate_ 套件傳遞/CriticalBasicConstraintsV1	arn:aws:acm-pca:::template/BlankEndEntityCertificate_CriticalBasicConstraints_APICSRPassthrough/V1	終端實體
CodeSigningCertificate_APICSR通過/V1	arn:aws:acm-pca:::template/CodeSigningCertificate_APICSRPassthrough/V1	程式碼簽署
EndEntityCertificate_APICSR通過/V1	arn:aws:acm-pca:::template/EndEntityCertificate_APICSRPassthrough/V1	終端實體
EndEntityClientAuthCertificate_APICSR通過/V1	arn:aws:acm-pca:::template/EndEntity	終端實體

範本名稱	範本 ARN	憑證類型
	ClientAuthCertificate_APICSRPassthrough/V1	
EndEntityServerAuthCertificate_APICSR通過/V1	arn:aws:acm-pca:::template/EndEntityServerAuthCertificate_APICSRPassthrough/V1	終端實體
OCSP _APIC 通行證 /V SigningCertificate 1	arn:aws:acm-pca:::template/OCSPSigningCertificate_APICSRPassthrough/V1	OCSP 簽署
次級緩存證書 _ 0_APICSR通過/V PathLen 1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
BlankSubordinate緩存 _ 0_APICSUR通過/V PathLen 1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen0_APICSRPassthrough/V1	CA
次級緩存證書 _ 1_APICSR通過/V PathLen 1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA

範本名稱	範本 ARN	憑證類型
BlankSubordinate緩存 _ 1_APICSUR通過/V PathLen 1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen1_APICSRPassthrough/V1	CA
次級緩存證書 _ 2_APICSR通過/3_API通過V PathLen 1 PathLen	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2_APICSRPassthrough/V1	CA
BlankSubordinate緩存_2_APICSR通過/V PathLen 1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen2_APICSRPassthrough/V1	CA
次級緩存證書 _ 3_APICSR通過/V PathLen 1	arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen3_APICSRPassthrough/V1	CA
BlankSubordinate緩存 _ 3_APICSR通過/V PathLen 1	arn:aws:acm-pca:::template/BlankSubordinateCACertificate_PathLen3_APICSRPassthrough/V1	CA

操作模板順序

已發行憑證中包含的資訊可以來自四個來源：範本定義、API 傳遞、CSR 傳遞和 CA 組態。

只有在您使用 API 傳遞或 APICSR 直通範本時，才會遵守 API 傳遞值。只有在您使用 CSR 傳遞或 APICSR 直通範本時，才會受到尊重。當這些資訊來源發生衝突時，通常會套用一般規則：對於每個擴充功能值，範本定義具有最高優先順序，接著是 API 傳遞值，接著是 CSR 傳遞延伸模組。

範例

1. EndEntityClientAuthCertificate_APIPASS 的範本定義會定義具有「TLS 網頁伺服器驗證、TLS 網頁用戶端驗證」值的 ExtendedKeyUsage 擴充功能。如果 ExtendedKeyUsage 在 CSR 或`IssueCertificateApiPassthrough`參數中定義，則 ExtendedKeyUsage 會忽略 的`ApiPassthrough`值，因為範本定義優先順序，而忽略 ExtendedKeyUsage 值的 CSR 值，因為範本不是 CSR 傳遞多種。

   > **Note**
   >
   > 儘管如此，範本定義會複製 CSR 中的其他值，例如「主旨」和「主旨替代名稱」。即使範本不是 CSR 傳遞品種，這些值仍會從 CSR 中取得，因為範本定義永遠具有最高優先順序。

2. EndEntityClientAuthCertificate_APICSRPASS 的範本定義會定義從 API 或 CSR 複製的主旨替代名稱 (SAN) 延伸模組。如果 SAN 延伸模組已在 CSR 中定義並在`IssueCertificate ApiPassthrough`參數中提供，則 API 傳遞值將優先考慮，因為 API 傳遞值優先於 CSR 傳遞值。

範本定義

下列各節提供有關支援AWS 私有 CA憑證範本的組態詳細資料。

BlankEndEntityCertificateAPI通過/V1 定義 (_P)

使用空白的終端實體憑證範本，您可以發行僅存在 X.509 Basic 條件約束的最終實體憑證。這是AWS私有 CA可以發行的最簡單的終端實體憑證，但可以使用 API 結構進行自訂。基本條件約束延伸會定義憑證是否為 CA 憑證。對於 Basic 條件約束，空白的最終實體憑證範本會強制執行 FALSE 值，以確保最終實體憑證已發行，而非 CA 憑證。

您可以使用空白傳遞範本來發行智慧卡憑證，這些憑證需要金鑰用法 (KU) 和延伸金鑰使用量 (EKU) 的特定值。例如，延伸金鑰使用可能需要用戶端驗證和智慧卡登入，而金鑰的使用可能需要「數位簽章」、「不可否認」和「金鑰加密」。與其他傳遞模板不同，空白最終實體證書模板允許配置 KU 和 EKU 擴展，其中 KU 可以是九個支持的值（數字簽名，不可否認性，密鑰加密，數據加密，密鑰協議，CRLSign，僅加密和僅限解密）中的任何一個，EKU 可以是支持的任何值（服務器認證客戶端，cotauProtection，編碼設計，時間戳記和 OCSP 簽署) 加上自訂擴充功能。 keyCertSign

BlankEndEntityCertificate_API通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	加:假
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankEndEntityCertificate_APICSR通過/V1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankEndEntityCertificate_APICSR通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	加:假
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankEndEntityCertificate_ CriticalBasicConstraints _ 介面傳遞 /V1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankEndEntityCertificate_ 套件傳遞/CriticalBasicConstraintsV1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	嚴重，CA：假
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態、API 或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankEndEntityCertificate_ CriticalBasicConstraints API通過/V1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankEndEntityCertificate_ AP CriticalBasicConstraints I通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	嚴重，CA：假
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 配置或 API 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankEndEntityCertificateCriticalBasicConstraints_ 企業社會責任傳遞 /V1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankEndEntityCertificate_ _ 企業社會責任傳CriticalBasicConstraints遞 /V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	嚴重，CA：假
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankEndEntityCertificate_企業社會責任通過/V1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankEndEntityCertificate_企業社會責任通過/V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	加:假
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate快取 _ 0_ 企業社會責任傳遞 /V1 定義 PathLen

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存 _ 0_ 企業社PathLen會責任傳遞 /V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 0
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate緩存_0_APICSR通過/V PathLen 1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存 _ 0_APICSUR通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 0
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]

參數	值
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate快取 _ 0_API通過/V PathLen 1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存 _ 0_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 0
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[從 CA 組態傳遞]

BlankSubordinate緩存_1_API通過/V PathLen 1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存 _ 1_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 1
授權金鑰識別碼	[來自 CA 證書的滑雪]

參數	值
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate快取 _ 1_ 企業社會責任傳遞 /V1 定義 PathLen

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存 _ 1_ 企業社PathLen會責任傳遞 /V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 1
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate緩存_1_APICSR通過/V PathLen 1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存 _ 1_APICSUR通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]

參數	值
基本限制	關鍵、CA:TRUE、pathlen: 1
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate緩存_2_API通過/V PathLen 1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存_2_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 2
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate快取 _ 2_ 企業社會責任傳遞 /V1 定義 PathLen

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存 _ 2_ 企業社PathLen會責任傳遞 /V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 2
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate緩存_2_APICSR通過/V PathLen 1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存 _ 2_APICSUR通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 2
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate緩存_3_API通過/V PathLen 1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存_3_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 3
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate快取 _ 3_ 企業社會責任傳遞 /V1 定義 PathLen

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存 _ 3_ 企業社PathLen會責任傳遞 /V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 3
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

BlankSubordinate緩存_3_APICSR通過/V PathLen 1 定義

如需空白範本的一般資訊，請參閱BlankEndEntityCertificateAPI通過/V1 定義 (_P)。

BlankSubordinate緩存 _ 3_ API 通過 PathLen

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 3
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

CodeSigningCertificate/V1 定義

此範本用來建立用於進行程式碼簽署的憑證。您可以搭配任何採用私有 CA 基礎設施的程式碼簽署解決方案，使用來自 AWS 私有 CA 的程式碼簽署憑證。例如，使用 AWS IoT 程式碼簽署的客戶可以使用 AWS 私有 CA 產生程式碼簽署憑證，並將其匯入 AWS Certificate Manager。如需詳細資訊，請參閱程式碼簽章的用途為何AWS IoT？ 並獲取和導入代碼簽名證書。

CodeSigningCertificate/1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	CA:FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]

參數	值
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要的數位簽章
延伸金鑰使用	重要，程式碼簽章
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在範本中。

CodeSigningCertificate_APICSR通過/V1 定義

此範本可擴充 CodeSigningCertificate /V1 以支援 API 和 CSR 傳遞值。

CodeSigningCertificate_APICSR通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	CA:FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要的數位簽章
延伸金鑰使用	重要，程式碼簽章
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

CodeSigningCertificateAPI通過/V1 定義 (_P)

此範本與CodeSigningCertificate範本相同，但有一個區別：在此範本中，如果未在範本中指定擴充功能，則會透過 API 將其他擴充功能AWS 私有 CA傳遞至憑證。範本中指定的擴充功能一律會覆寫 API 中的擴充功能。

CodeSigningCertificate_API通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	CA:FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要的數位簽章
延伸金鑰使用	重要，程式碼簽章
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

CodeSigningCertificate_企業社會責任通過/V1 定義

此範本與 CodeSigningCertificate 範本之間唯一的不同點在於，如果沒有在此範本中指定延伸，則 AWS 私有 CA 會將憑證簽署請求 (CSR) 中的其他延伸傳遞至憑證。範本中指定的延伸一律會覆寫 CSR 中的延伸。

CodeSigningCertificate_企業社會責任通過/V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]

參數	值
基本限制	CA:FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要的數位簽章
延伸金鑰使用	重要，程式碼簽章
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在範本中。

EndEntityCertificate/V1 定義

此範本用來建立終端實體 (例如作業系統或 Web 伺服器) 的憑證。

EndEntityCertificate/1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 伺服器身分驗證、TLS Web 用戶端身分驗證
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在範本中。

EndEntityCertificate_APICSR通過/V1 定義

此範本可擴充 EndEntityCertificate /V1 以支援 API 和 CSR 傳遞值。

EndEntityCertificate_APICSR通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 伺服器身分驗證、TLS Web 用戶端身分驗證
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

EndEntityCertificateAPI通過/V1 定義 (_P)

此範本與`EndEntityCertificate`範本相同，但有一個區別：在此範本中，如果未在範本中指定擴充功能，則會透過 API 將其他擴充功能AWS 私有 CA傳遞至憑證。範本中指定的擴充功能一律會覆寫 API 中的擴充功能。

EndEntityCertificate_API通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]

參數	值
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 伺服器身分驗證、TLS Web 用戶端身分驗證
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

EndEntityCertificate_企業社會責任通過/V1 定義

此範本與 `EndEntityCertificate` 範本之間唯一的不同點在於，如果沒有在此範本中指定延伸，則 AWS 私有 CA 會將憑證簽署請求 (CSR) 中的其他延伸傳遞至憑證。範本中指定的延伸一律會覆寫 CSR 中的延伸。

EndEntityCertificate_企業社會責任通過/V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 伺服器身分驗證、TLS Web 用戶端身分驗證

參數	值
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在範本中。

EndEntityClientAuthCertificate/V1 定義

此範本與延伸金鑰使用量值中`EndEntityCertificate`唯一的不同，這會將其限制為 TLS Web 用戶端驗證。

EndEntityClientAuthCertificate/1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 用戶端身分驗證
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在範本中。

EndEntityClientAuthCertificate_APICSR通過/V1 定義

此範本可擴充 EndEntityClientAuthCertificate /V1 以支援 API 和 CSR 傳遞值。

EndEntityClientAuthCertificate_APICSR通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 用戶端身分驗證
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

EndEntityClientAuthCertificateAPI通過/V1 定義 (_P)

此範本與 `EndEntityClientAuthCertificate` 範本之間唯一的不同點在於，在此範本中，如果未在範本中指定擴充功能，則會透過 API 將其他擴充功能AWS 私有 CA傳遞至憑證。範本中指定的擴充功能一律會覆寫 API 中的擴充功能。

EndEntityClientAuthCertificate_API通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]

參數	值
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 用戶端身分驗證
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

EndEntityClientAuthCertificate_企業社會責任通過/V1 定義

此範本與 `EndEntityClientAuthCertificate` 範本之間唯一的不同點在於，如果沒有在此範本中指定延伸，則 AWS 私有 CA 會將憑證簽署請求 (CSR) 中的其他延伸傳遞至憑證。範本中指定的延伸一律會覆寫 CSR 中的延伸。

EndEntityClientAuthCertificate_企業社會責任通過/V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 用戶端身分驗證
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在範本中。

EndEntityServerAuthCertificate/V1 定義

此範本與延伸金鑰使用量值中的EndEntityCertificate唯一不同，這會將其限制為 TLS Web 伺服器驗證。

EndEntityServerAuthCertificate/1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 伺服器身分驗證
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在範本中。

EndEntityServerAuthCertificate_APICSR通過/V1 定義

此範本可擴充 EndEntityServerAuthCertificate /V1 以支援 API 和 CSR 傳遞值。

EndEntityServerAuthCertificate_APICSR通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]

參數	值
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 伺服器身分驗證
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

EndEntityServerAuthCertificateAPI通過/V1 定義 (_P)

此範本與 `EndEntityServerAuthCertificate` 範本之間唯一的不同點在於，在此範本中，如果未在範本中指定擴充功能，則會透過 API 將其他擴充功能AWS 私有 CA傳遞至憑證。範本中指定的擴充功能一律會覆寫 API 中的擴充功能。

EndEntityServerAuthCertificate_API通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 伺服器身分驗證
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

EndEntityServerAuthCertificate_企業社會責任通過/V1 定義

此範本與 EndEntityServerAuthCertificate 範本之間唯一的不同點在於，如果沒有在此範本中指定延伸，則 AWS 私有 CA 會將憑證簽署請求 (CSR) 中的其他延伸傳遞至憑證。範本中指定的延伸一律會覆寫 CSR 中的延伸。

EndEntityServerAuthCertificate_企業社會責任通過/V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	CA：FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章、金鑰加密
延伸金鑰使用	TLS Web 伺服器身分驗證
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在範本中。

OCSP 定SigningCertificate義

此範本用來建立用於簽署 OCSP 回應的憑證。範本與範本完全相同，不同之處在於延伸金鑰用法值會指定 OCSP 簽章而非程式碼簽章。CodeSigningCertificate

OCSP SigningCertificate

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	CA:FALSE

參數	值
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要的數位簽章
延伸金鑰使用	重要、OCSP 簽署
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在範本中。

OCSP 傳遞/SigningCertificateV1 定義

此範本可擴充 OCSP SigningCertificate /V1，以支援 API 和 CSR 傳遞值。

OCSP _APIC 通行證 /V SigningCertificate 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	CA:FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要的數位簽章
延伸金鑰使用	重要、OCSP 簽署
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

OCSP 傳遞傳遞/SigningCertificateV1 定義

此範本與 `OCSPSigningCertificate` 範本之間唯一的不同點在於，在此範本中，如果未在範本中指定擴充功能，則會透過 API 將其他擴充功能AWS 私有 CA傳遞至憑證。範本中指定的擴充功能一律會覆寫 API 中的擴充功能。

OCSP _API通行證/V SigningCertificate 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	`CA:FALSE`
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要的數位簽章
延伸金鑰使用	重要、OCSP 簽署
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

OCSP SigningCertificate _ 企業社會責任傳遞 /V1 定義

此範本與 `OCSPSigningCertificate` 範本之間唯一的不同點在於，如果沒有在此範本中指定延伸，則 AWS 私有 CA 會將憑證簽署請求 (CSR) 中的其他延伸傳遞至憑證。範本中指定的延伸一律會覆寫 CSR 中的延伸。

OCSP _ 企業社SigningCertificate會責任傳遞 /V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]

參數	值
基本限制	CA:FALSE
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要的數位簽章
延伸金鑰使用	重要、OCSP 簽署
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在範本中。

root快取證明/V1 定義

此範本用來發行自我簽署的根 CA 憑證。CA 憑證包含關鍵的基本限制條件延伸，其中會將 CA 欄位設為 TRUE，指定憑證可以用來發行 CA 憑證。範本不會指定路徑 length (pathLenConstraint)，因為這可能會阻止 future 階層的擴充。其中已排除延伸金鑰使用方式，以防止使用 CA 憑證做為 TLS 用戶端或伺服器憑證。因為無法撤銷自我簽署憑證，所以沒有指定任何 CRL 資訊。

RootCACertificate/V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵，CA:TRUE
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章 keyCertSign、CRL 符號
CRL 發佈點	N/A

根據快取 _ API 傳遞 /V1 定義

此範本會延伸根快取/V1，以支援 API 傳遞值。

根據快取 _ API 傳遞 /V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵，CA:TRUE
授權金鑰識別碼	[來自 API 的直通]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章 keyCertSign、CRL 符號
CRL 發佈點 *	N/A

BlankRoot緩存_API通過/V1 定義

使用空白的根憑證範本，您可以發行僅存在 X.509 基本條件約束的根憑證。這是AWS 私有 CA可以發行的最簡單的根憑證，但可以使用 API 結構進行自訂。基本條件約束延伸會定義憑證是否為 CA 憑證。空白的根憑證範本會TRUE針對基本條件約束強制執行的值，以確保已發行根 CA 憑證。

您可以使用空白傳遞根範本來發行需要特定金鑰使用值 (KU) 的根憑證。例如，金鑰使用可能需要 keyCertSign andcRLSign，但不需要digitalSignature。與其他非空白根傳遞憑證範本不同，空白根憑證範本允許設定 KU 延伸模組，其中 KU 可以是九個支援的值 (digitalSignature、、、、nonRepudiation、keyEnciphermentdataEnciphermentkeyAgreeme和decipherOnly) 中的任何一個。

BlankRoot緩存_API通過/V1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]

參數	值
基本限制	關鍵，CA:TRUE
主旨金鑰識別碼	[衍生自企業社會責任]

BlankRoot快取 _ 0_API通過/V PathLen 1 定義

如需有關空白根 CA 範本的一般資訊，請參閱???。

BlankRoot緩存 _ 0_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 0
主旨金鑰識別碼	[衍生自企業社會責任]

BlankRoot緩存_1_API通過/V PathLen 1 定義

如需有關空白根 CA 範本的一般資訊，請參閱???。

BlankRoot緩存 _ 1_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 1
主旨金鑰識別碼	[衍生自企業社會責任]

BlankRoot緩存_2_API通過/V PathLen 1 定義

如需有關空白根 CA 範本的一般資訊，請參閱???。

BlankRoot緩存_2_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 2
主旨金鑰識別碼	[衍生自企業社會責任]

BlankRoot緩存_3_API通過/V PathLen 1 定義

如需有關空白根 CA 範本的一般資訊，請參閱???。

BlankRoot緩存_3_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 3
主旨金鑰識別碼	[衍生自企業社會責任]

次級緩存 _ 0/V PathLen 1 定義

此範本可用來發行路徑長度為的0從屬 CA 憑證。CA 憑證包含關鍵的基本限制條件延伸，其中會將 CA 欄位設為 TRUE，指定憑證可以用來發行 CA 憑證。其中並未包含延伸金鑰使用方式，該金鑰使用方式會防止將 CA 憑證做為 TLS 用戶端或伺服器憑證使用。

如需認證路徑的詳細資訊，請參閱對認證路徑設定長度限制條件。

次級PathLen緩存證書 _ 0/V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 0
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定為啟用 CRL 產生時，才會將 CRL 發佈點包含在隨此範本所發行的憑證中。

次級緩存 _ 0_APICSR通過/V1 定義 PathLen

此範本擴充次級快取 PathLen 0/V1，以支援 API 和 CSR 傳遞值。

次級緩存證書 _ 0_APICSR通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 0
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

次級緩存 _ 0_API通過/V PathLen 1 定義

此範本擴充次級快取 _ 0/V1，以支援 API 傳遞PathLen值。

次級緩存證書 _ 0_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 0
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

次級快取 _ 0_ 企業社會責任傳遞 /V PathLen 1 定義

此範本與 SubordinateCACertificate_PathLen0 範本之間唯一的不同點在於，如果沒有在此範本中指定延伸，則 AWS 私有 CA 會將憑證簽署請求 (CSR) 中的其他延伸傳遞至憑證。範本中指定的延伸一律會覆寫 CSR 中的延伸。

> **Note**
>
> 必須在以外建立包含自訂其他擴充功能的 CSR AWS 私有 CA。

次級PathLen緩存證書 _ 0_ 企業社會責任傳遞 /V1

參數	值
主體替代名稱	[來自企業社會責任的直通]

參數	值
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 0
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在使用此範本發行的憑證中。

次級緩存 _ 1/V PathLen 1 定義

此範本可用來發行路徑長度為的1從屬 CA 憑證。CA 憑證包含重要的基本條件約束延伸，其中 CA 欄位設定TRUE為指定憑證可用來發行 CA 憑證。其中並未包含延伸金鑰使用方式，該金鑰使用方式會防止將 CA 憑證做為 TLS 用戶端或伺服器憑證使用。

如需認證路徑的詳細資訊，請參閱對認證路徑設定長度限制條件。

次級PathLen緩存證書 _ 1/V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 1
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在使用此範本發行的憑證中。

次級緩存 _ 1_APICSR通過/V1 定義 PathLen

此範本擴充次級快取的 PathLen 1/V1，以支援 API 和 CSR 傳遞值。

次級緩存證書 _ 1_APICSR通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 1
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

次級緩存 _ 1_API通過/V PathLen 1 定義

此範本擴充次級快取 _ 0/V1，以支援 API 傳遞PathLen值。

次級緩存證書 _ 1_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 1
授權金鑰識別碼	[來自 CA 證書的滑雪]

參數	值
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

次級緩存 _ 1_CSR通過/V1 定義 PathLen

此範本與 SubordinateCACertificate_PathLen1 範本之間唯一的不同點在於，如果沒有在此範本中指定延伸，則 AWS 私有 CA 會將憑證簽署請求 (CSR) 中的其他延伸傳遞至憑證。範本中指定的延伸一律會覆寫 CSR 中的延伸。

> **Note**
>
> 必須在以外建立包含自訂其他擴充功能的 CSR AWS 私有 CA。

次級PathLen緩存證書 _ 1_ 企業社會責任傳遞 /V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 1
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在使用此範本發行的憑證中。

次級緩存 _ 2/V PathLen 1 定義

此範本用來發行路徑長度為 2 的次級 CA 憑證。CA 憑證包含重要的基本條件約束延伸，其中 CA 欄位設定TRUE為指定憑證可用來發行 CA 憑證。其中並未包含延伸金鑰使用方式，該金鑰使用方式會防止將 CA 憑證做為 TLS 用戶端或伺服器憑證使用。

如需認證路徑的詳細資訊，請參閱對認證路徑設定長度限制條件。

次級PathLen緩存證書 _ 2/V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 2
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在使用此範本發行的憑證中。

次級緩存 _ 2_APICSR通過/V1 定義 PathLen

此範本延伸次級快取 _ PathLen 2/V1，以支援 API 和 CSR 傳遞值。

次級緩存證書 _ 2_APICSR通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 2

參數	值
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

次級緩存 _ 2_API通過/V PathLen 1 定義

此範本擴充次級快取 _ 2/V1，以支援 API 傳遞PathLen值。

次級緩存證書 _ 2_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 2
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

次級緩存 _ 2_CSR通過/V1 定義 PathLen

此範本與 SubordinateCACertificate_PathLen2 範本之間唯一的不同點在於，如果沒有在此範本中指定延伸，則 AWS 私有 CA 會將憑證簽署請求 (CSR) 中的其他延伸傳遞至憑證。範本中指定的延伸一律會覆寫 CSR 中的延伸。

> **Note**
>
> 必須在以外建立包含自訂其他擴充功能的 CSR AWS 私有 CA。

次級PathLen緩存證書 _ 2_ 企業社會責任傳遞 /V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 2
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在使用此範本發行的憑證中。

下級緩存_3/V1 定義 PathLen

此範本用來發行路徑長度為 3 的次級 CA 憑證。CA 憑證包含重要的基本條件約束延伸，其中 CA 欄位設定TRUE為指定憑證可用來發行 CA 憑證。其中並未包含延伸金鑰使用方式，該金鑰使用方式會防止將 CA 憑證做為 TLS 用戶端或伺服器憑證使用。

如需認證路徑的詳細資訊，請參閱對認證路徑設定長度限制條件。

下屬緩PathLen存證書 _ 3/V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]

參數	值
基本限制	關鍵、CA:TRUE、pathlen: 3
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在使用此範本發行的憑證中。

次級緩存 _ 3_APICSR通過/V1 定義 PathLen

此範本擴充次級快取 PathLen 3/V1，以支援 API 和 CSR 傳遞值。

次級緩存證書 _ 3_APICSR通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 3
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

次級緩存 _ 3_API通過/V PathLen 1 定義

此範本擴充次級快取 _ 3/V1，以支援 API 傳遞PathLen值。

次級緩存證書 _ 3_API通過/V PathLen 1

參數	值
主體替代名稱	[來自 API 或企業社會責任的傳遞]
主旨	[來自 API 或企業社會責任的傳遞]
基本限制	關鍵、CA:TRUE、pathlen: 3
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[從 CA 組態傳遞]

* 只有在 CA 設定啟用 CRL 產生時，才會將 CRL 發佈點包含在範本中。

次級緩存 _ 3_CSR通過/V1 定義 PathLen

此範本與 SubordinateCACertificate_PathLen3 範本之間唯一的不同點在於，如果沒有在此範本中指定延伸，則 AWS 私有 CA 會將憑證簽署請求 (CSR) 中的其他延伸傳遞至憑證。範本中指定的延伸一律會覆寫 CSR 中的延伸。

> **Note**
>
> 必須在以外建立包含自訂其他擴充功能的 CSR AWS 私有 CA。

次級PathLen緩存證書 _ 3_ 企業社會責任傳遞 /V1

參數	值
主體替代名稱	[來自企業社會責任的直通]
主旨	[來自企業社會責任的直通]
基本限制	關鍵、CA:TRUE、pathlen: 3

參數	值
授權金鑰識別碼	[來自 CA 證書的滑雪]
主旨金鑰識別碼	[衍生自企業社會責任]
金鑰用途	重要、數位簽章keyCertSign 、CRL 符號
CRL 發佈點 *	[來自 CA 組態或 CSR 的傳遞]

* 只有在 CA 已設為啟用產生 CRL 時，CRL 分佈點才會包含在使用此範本發行的憑證中。

使用AWS 私有 CA應用程式介面 (Java 範例)

您可以使用 AWS Private Certificate Authority API，透過編寫程式的方式傳送 HTTP 請求，與服務互動。服務會傳回 HTTP 回應。如需詳細資訊，請參閱 AWS Private Certificate AuthorityAPI 參考。

除了 HTTP API，您還可以使用 AWS 開發套件和命令列工具來與 AWS 私有 CA 互動。建議透過 HTTP API 進行。如需詳細資訊，請參閱 Amazon Web Services 適用工具。下列主題說明如何使用 AWS SDK for Java 來程式設計 AWS 私有 CA API。

的 GetCertificateAuthorityCsr, GetCertificate, 和DescribeCertificateAuthorityAuditReport操作支持服務員. 您可以使用等待程式以根據特定資源的存在或狀態來控制程式碼的進度。如需詳細資訊，請參閱AWS開發人員部落格中的下列主題以及服務員。AWS SDK for Java

主題

- 以程式設計方式建立並啟動根 CA
- 以編程方式創建和激活下屬 CA
- CreateCertificateAuthority
- 用 CreateCertificateAuthority來支援使用中目錄
- CreateCertificateAuthorityAuditReport
- CreatePermission
- DeleteCertificateAuthority
- DeletePermission
- DeletePolicy
- DescribeCertificateAuthority
- DescribeCertificateAuthorityAuditReport
- GetCertificate
- GetCertificateAuthorityCertificate
- GetCertificateAuthorityCsr
- GetPolicy
- ImportCertificateAuthorityCertificate
- IssueCertificate
- ListCertificateAuthorities
- ListPermissions

- ListTags
- PutPolicy
- RestoreCertificateAuthority
- RevokeCertificate
- TagCertificateAuthorities
- UntagCertificateAuthority
- UpdateCertificateAuthority
- 使用自訂主體名稱建立 CA 和憑證
- 使用自訂擴充功能建立憑證

以程式設計方式建立並啟動根 CA

此 Java 範例顯示如何使用下列 AWS 私有 CA API 動作來啟動根 CA：

- 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);
    }
}
```

以編程方式創建和激活下屬 CA

此 Java 範例顯示如何使用下列 AWS 私有 CA API 動作來啟動從屬 CA：

- 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

下面的 Java 示例演示了如何使用該CreateCerticateAuthority操作。

此操作可建立私有次級憑證授權機構 (CA)。您必須指定 CA 組態、撤銷組態、CA 類型和選用的冪等符記。

CA 組態指定以下項目：

- 用來建立 CA 私有金鑰的演算法名稱和金鑰大小
- CA 用於簽署的簽署演算法類型
- X.500 主旨資訊

CRL 組態指定以下項目：

- CRL 過期期間天數 (CRL 有效期間)
- 將包含 CRL 的 Amazon S3 存儲桶
- 包含於 CA 發行的憑證的 S3 儲存貯體 CNAME 別名

如果成功，此函數會傳回 CA 的 Amazon Resource Name (ARN)。

```
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);
    }
}
```

您的輸出應類似以下內容：

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

用 CreateCertificateAuthority來支援使用中目錄

下面的 Java 示例演示了如何使用CreateCerticateAuthority操作來創建一個 CA，可以安裝在 Microsoft 活動目錄（AD）的企業 NTAuth 存儲。

此作業會使用自訂物件識別碼 (OID) 建立私有根憑證授權單位 (CA)。如需詳細資訊和對等作業的AWS CLI範例，請參閱建立使用中目錄登入的 CA。

如果成功，此函數會傳回 CA 的 Amazon Resource Name (ARN)。

```
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);
    }
}
```

您的輸出應類似以下內容：

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

CreateCertificateAuthorityAuditReport

下面的 Java 示例演示了如何使用該CreateCertificateAuthorityAuditReport操作。

此操作可在每次發行或撤銷憑證時建立稽核報告。報告會儲存在您在輸入時指定的 Amazon S3 儲存貯體中。您可以每 30 分鐘產生一份新的報告。

```
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);

    }
}
```

您的輸出應類似以下內容：

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

CreatePermission

下面的 Java 示例演示了如何使用該CreatePermission操作。

此操作會將私有 CA 的存取許可指派給指定 AWS 服務委託人。您可以將建立及擷取私有 CA 憑證的許可授予服務，及列出由私有 CA 授予的作用中許可。若要透過 ACM 自動更新憑證，您必須將 CA 中所有可能的權限 (IssueCertificateGetCertificate、和ListPermissions) 指派給 ACM 服務主體 (acm.amazonaws.com)。您可以通過調用ListCertificateAuthorities函數找到 CA 的 ARN。

建立權限後，您可以使用函數檢查它，或使用ListPermissions函DeletePermission數將其刪除。

```
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

下面的 Java 示例演示了如何使用該DeleteCertificateAuthority操作。

此作業會刪除您使用作業建立的私人憑證授權單位 (CA)。CreateCertificateAuthorityDeleteCertificateAuthority 操作需要您提供要刪除 CA 的 ARN。您可以通過調用ListCertificateAuthorities操作找到 ARN。如果私有 CA 的狀態為 CREATING 或 PENDING_CERTIFICATE，您可以立即刪除。不過，如果您已經匯入憑證，則無法立即將其刪除。您必須先呼叫UpdateCertificateAuthority作業來停用 CA，並將Status參數設定為DISABLED。您接著可以在 DeleteCertificateAuthority 操作中使用 PermanentDeletionTimeInDays 參數來指定天數 (7 到 30)。在此期間，私有 CA 皆可以還原至 disabled 狀態。依預設，如果您未設定

PermanentDeletionTimeInDays 參數，還原期間為 30 天。過了此期間，私有 CA 會永久刪除，無法還原。如需詳細資訊，請參閱 還原 CA。

如需說明如何使用RestoreCertificateAuthority作業的 Java 範例，請參閱RestoreCertificateAuthority。

```
package com.amazonaws.samples;

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

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

import com.amazonaws.services.acmpca.model.DeleteCertificateAuthorityRequest;

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

public class DeleteCertificateAuthority {

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

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

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

```
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol,
 endpointRegion);

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

      // Create a 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

下面的 Java 示例演示了如何使用該DeletePermission操作。

此作業會刪除私有 CA 使用作CreatePermissions業委派給AWS服務主體的權限。您可以通過調用ListCertificateAuthorities函數找到 CA 的 ARN。您可以通過調用ListPermissions函數來檢查 CA 授予的權限。

```
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

下面的 Java 示例演示了如何使用該DeletePolicy操作。

此作業會刪除附加至私有 CA 的資源型原則。以資源為基礎的政策是用來啟用跨帳戶 CA 共用。您可以通過調用ListCertificateAuthorities操作來找到私有 CA 的 ARN。

相關 API 動作包括PutPolicy和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

下面的 Java 示例演示了如何使用該DescribeCertificateAuthority操作。

此操作可列出私有憑證授權機構 (CA) 的相關資訊。您必須指定私有 CA 的 ARN (Amazon Resource Name)。輸出包含 CA 的狀態。此可為下列任何一項：

- CREATING— AWS 私有 CA 正在建立您的私有憑證授權單位。
- PENDING_CERTIFICATE— 憑證處於擱置狀態。您必須使用現場部署的根 CA 或次級 CA 來簽署私有 CA CSR，然後將其匯入 PCA。
- ACTIVE— 您的私人 CA 處於活動狀態。
- DISABLED— 您的私人 CA 已被禁用。
- EXPIRED— 您的私有 CA 憑證已過期。
- FAILED— 您的私人 CA 無法建立。
- DELETED— 您的私有 CA 在恢復期內，之後它將被永久刪除。

```
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

下面的 Java 示例演示了如何使用該DescribeCertificateAuthorityAuditReport操作。

作業會列出您透過呼叫作業所建立之CreateCertificateAuthorityAuditReport特定稽核報告的相關資訊。稽核資訊會在每次使用憑證授權單位 (CA) 私密金鑰時建立。私密金鑰會在您發行憑證、的私有金鑰、簽署 CRL 或撤銷憑證時使用。

```
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);
   }
}
```

您的輸出應類似以下內容：

```
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

下面的 Java 示例演示了如何使用該GetCertificate操作。

此操作可從您的私有 CA 擷取憑證。當您呼叫IssueCertificate作業時，會傳回憑證的 ARN。呼叫 GetCertificate 操作時，您必須指定私有 CA 的 ARN，和發行憑證的 ARN。如果憑證的狀態為 ISSUED，便可擷取憑證。您可以呼叫CreateCertificateAuthorityAuditReport作業以建立報告，其中包含私有 CA 所發行和撤銷之所有憑證的相關資訊。

```
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);
   }
}
```

您的輸出應為類似以下的憑證授權單位 (CA) 憑證鏈和您指定的憑證。

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

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

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

GetCertificateAuthorityCertificate

下面的 Java 示例演示了如何使用該GetCertificateAuthorityCertificate操作。

此操作能為您的私有憑證授權機構 (CA) 擷取憑證和憑證鏈。憑證和鏈結都是 PEM 格式的 base64 編碼字串。憑證鏈不包含 CA 憑證。憑證鏈中的每個憑證都會依序簽署。

```
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);
   }
}
```

您的輸出應為類似以下您指定的憑證授權單位 (CA) 的憑證和憑證鏈。

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

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

GetCertificateAuthorityCsr

下面的 Java 示例演示了如何使用該GetCertificateAuthorityCsr操作。

此操作能為您的私有憑證授權機構 (CA) 擷取憑證簽署要求 (CSR)。當您呼叫CreateCertificateAuthority作業時，會建立 CSR。將 CSR 帶到您的內部部署 X.509 基礎設施，並使用您的根 CA 或次級 CA 簽署。然後呼叫作業，將已簽署的憑證匯入回 ACM PCA。ImportCertificateAuthorityCertificateCSR 會以 PEM 格式的 Base64 編碼字串傳回。

```
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);
    }
}
```

您指定的憑證授權單位 (CA) 的輸出應類似下列。憑證簽署要求 (CSR) 是以 base64 編碼 (PEM 格式)。儲存到本機檔案，並帶到您的內部部署 X.509 基礎設施，然後使用您的根 CA 或次級 CA 簽署。

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

GetPolicy

下面的 Java 示例演示了如何使用該GetPolicy操作。

作業會擷取附加至私有 CA 的以資源為基礎的原則。以資源為基礎的政策是用來啟用跨帳戶 CA 共用。您可以通過調用ListCertificateAuthorities操作來找到私有 CA 的 ARN。

相關 API 動作包括PutPolicy和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

下面的 Java 示例演示了如何使用該ImportCertificateAuthorityCertificate操作。

此操作能將簽署的私有 CA 憑證匯入 AWS 私有 CA。在呼叫此作業之前，您必須先呼叫CreateCertificateAuthority作業來建立私有憑證授權單位。然後，您必須呼叫GetCertificateAuthorityCsr作業來產生憑證簽署要求 (CSR)。將 CSR 帶到現場部署的 CA，並使用您的根憑證或次級憑證簽署。建立憑證鏈，並將簽署的憑證和憑證鏈複製到您的工作目錄。

```
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

下面的 Java 示例演示了如何使用該IssueCertificate操作。

此作業會使用您的私有憑證授權單位 (CA) 來發行終端實體憑證。此操作會傳回憑證的 Amazon Resource Name (ARN)。您可以呼叫GetCertificate並指定 ARN 來擷取憑證。

> **Note**
>
> 此IssueCertificate作業會要求您指定憑證範本。此範例使用EndEntityCertificate/V1範本。如需有關所有可用範本的資訊，請參閱瞭解憑證範本。

```
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);
    }
}
```

您的輸出應類似以下內容：

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

ListCertificateAuthorities

以下 Java 範例示範如何使用ListCertificateAuthorities操作。

此作業會列出您使用CreateCertificateAuthority操作。

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

如果您有任何可列出的憑證授權機構，則您的輸出應類似下列：

```
[{
 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

下面的 Java 示例演示了如何使用該ListPermissions操作。

此操作會列出您私有 CA 所指派的許可 (若有的話)。權限 (包括IssueCertificateGetCertificateListPermissions、和) 可以指派給作CreatePermission業的AWS服務主體，並隨DeletePermissions作業撤銷。

```
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);
    }
}
```

若指定的私有 CA 已將許可指派給服務委託人，您的輸出應該會和以下內容相似：

```
[{
       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

下面的 Java 示例演示了如何使用該ListTags操作。

此操作可列出與私有 CA 相關的標籤 (如果有)。標籤是您可以用於識別及組織 CA 的標記。每個標籤皆包含索引鍵與選用值。呼叫TagCertificateAuthority作業，將一或多個標籤新增至 CA。呼叫作UntagCertificateAuthority業以移除標籤。

```
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);
    }
}
```

如果您有任何可列出的標籤，則您的輸出應類似下列：

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

PutPolicy

下面的 Java 示例演示了如何使用該PutPolicy操作。

此作業會將以資源為基礎的原則附加至私有 CA，以啟用跨帳戶共用。經原則授權時，居住在另一個AWS帳戶的主體可以使用不擁有的私有 CA 發行和續約私有終端實體憑證。您可以通過調用ListCertificateAuthorities操作來找到私有 CA 的 ARN。如需政策範例，請參閱以資源為基礎的政策 AWS 私有 CA指引。

將原則附加至 CA 之後，您可以使用動作檢查該原則，或使用GetPolicy動DeletePolicy作將其刪除。

```
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

下面的 Java 示例演示了如何使用該RestoreCertificateAuthority操作。您可以隨時還原仍處於還原期間的私有 CA。目前，這個期間可以從刪除之日起持續 7 到 30 天，並可在刪除 CA 時加以定義。如需詳細資訊，請參閱 還原 CA。另請參閱 DeleteCertificateAuthority Java 範例。

```
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

下面的 Java 示例演示了如何使用該RevokeCertificate操作。

此作業會撤銷您呼叫作IssueCertificate業所發出的憑證。如果您在建立或更新私有 CA 時啟用憑證撤銷清單 (CRL)，則 CRL 中會包含撤銷憑證的相關資訊。 AWS 私有 CA將 CRL 寫入您指定的 Amazon S3 儲存貯體。如需詳細資訊，請參閱CrlConfiguration結構。

```
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

下面的 Java 示例演示了如何使用該TagCertificateAuthority操作。

此操作會將一或多個標籤新增到您的私有 CA。標籤是您可以用於識別及組織 AWS 資源的標記。每個標籤皆包含索引鍵與選用值。呼叫此操作時，您可以依據 Amazon Resource Name (ARN) 來指定私有 CA。使用鍵值組指定標籤。若要識別該 CA 的特定特性，您可以將標籤套用至一個私有 CA。或者，若您要篩選這些 CA 之間的共同關係，您可以將相同的標籤套用至多個私有 CA。若要移除一或多個標籤，請使用此UntagCertificateAuthority作業。呼叫ListTags作業以查看與 CA 相關聯的標籤。

```
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

下面的 Java 示例演示了如何使用該UntagCertificateAuthority操作。

此操作會從私有 CA 移除一或多個標籤。一個標籤包含一對索引鍵/值對。如果您在呼叫此操作時沒有指定標籤的值部分，便會移除標籤，不論其值為何。如果您指定一個值，標籤只會在與指定的值相關聯時移除。若要將標籤新增至私有 CA，請使用此TagCertificateAuthority作業。呼叫ListTags作業以查看與 CA 相關聯的標籤。

```
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

下面的 Java 示例演示了如何使用該UpdateCertificateAuthority操作。

此操作會更新私有憑證授權機構 (CA) 的狀態或組態。您的私有 CA 必須處於 ACTIVE 或 DISABLED 狀態才能更新。您可以停用 ACTIVE 狀態的私有 CA，或將處於 DISABLED 狀態的 CA 再次啟用。

```
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;
        }
    }
}
```

使用自訂主體名稱建立 CA 和憑證

所以此CustomAttribute物件可讓管理員將自訂物件識別碼 (OID) 傳遞至私有 CA 和憑證。自訂 OID 可用來建立反映組織結構和需求的特殊主體名稱階層。必須使用下列其中一個建立自訂憑證ApiPassthrough範本。如需範本的詳細資訊，請參閱模板品種。如需使用自訂屬性的詳細資訊，請參閱簽發私人終端實體證書和建立 CA (CLI) 的程序 。

您無法使用StandardAttributes結合CustomAttributes。但是，您可以將標準 OID 作為CustomAttributes。下表列出預設的主旨名稱 OID：

主旨名稱	物件識別碼
國家	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

主旨名稱	物件識別碼
Locality	2.5.4.7
組織	2.5.4.10
OrganizationalUnit	2.5.4.11
Pseudonym	2.5.4.65
SerialNumber	2.5.4.5
州	2.5.4.8
Surname	2.5.4.4
標題	2.5.4.12

主題

- 使用建立 CA CustomAttribute
- 發行憑證 CustomAttribute

使用建立 CA 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);
    }
```

```
}
```

發行憑證 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);
    }
}
```

使用自訂擴充功能建立憑證

所以此CustomExtension物件可讓管理員在私有憑證中設定自訂 X.509 延伸模組。必須使用下列其中一個建立自訂憑證 ApiPassthrough 範例範本 如需範本的詳細資訊，請參閱模板品種。如需使用自訂範本的詳細資訊，請參閱簽發私人終端實體證書。

主題

- 使用啟動下屬 CA NameConstraints 擴展
- 簽發具有 QC 語句擴展的證書

使用啟動下屬 CA NameConstraints 擴展

```
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);
    }
}
```

簽發具有 QC 語句擴展的證書

```
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);
    }
}
```

使用 AWS 私有 CA API 來實現問題標準（Java 示例）

您可以使用 AWS Private Certificate Authority API 建立符合 Mat ter 連線標準的憑證。Matter 指定的憑證組態可改善多個工程平台之物聯網 (IoT) 裝置的安全性和一致性。如需有關物件的詳細資訊，請參閱建置。

本節中的 Java 範例會透過傳送 HTTP 要求與服務互動。服務會傳回 HTTP 回應。如需詳細資訊，請參閱 AWS Private Certificate AuthorityAPI 參考。

除了 HTTP API，您還可以使用 AWS 開發套件和命令列工具來與 AWS 私有 CA 互動。建議透過 HTTP API 進行。如需詳細資訊，請參閱 Amazon Web Services 適用工具。下列主題說明如何使用 AWS SDK for Java 來程式設計 AWS 私有 CA API。

的 GetCertificateAuthorityCsr, GetCertificate, 和DescribeCertificateAuthorityAuditReport操作支持服務員. 您可以使用等待程式以根據特定資源的存在或狀態來控制程式碼的進度。如需詳細資訊，請參閱AWS開發人員部落格中的下列主題以及服務員。AWS SDK for Java

於 2023 年 10 月發布的事項 1.2 支持使用證書撤銷列表（CRL）的 DAC 撤銷。為了協助您符合目前的 Matter 標準，當您針對核發 Metter 憑證的 CA 啟用 CRL 撤銷時，在`CrlDistributionPointExtensionConfiguration`結構中的`CrlConfiguration`物件中，將設定`OmitExtension`為。`true`

通常，CA 會將 CRL 發佈點 (CDP) 內嵌在其發行的憑證中，以便執行憑證鏈結驗證的信賴方可以擷取 CRL 並檢查憑證狀態。在「事項」中，CDP URI 不會寫入憑證。相反地，使用者會從「問題分散式合規分類帳」(DCL) (受信任的 Metter 資料存放區) 擷取 CDP。您必須將 CDP URI 上傳至物件 DCL，以便在驗證 DAC 時可以探索它。如需有關決定 CDP URI 的詳細資訊，請參閱決定 CRL 發佈點 (CDP) URI 。如需有關 Matter 的詳細資訊，請參閱 Mat ter DCL 文件。

主題

- 啟動產品驗證授權單位 (PAA)
- 啟用產品驗證中間 (PAI)
- 建立裝置驗證憑證 (DAC)
- 為節點操作憑證 (NOC) 啟動根 CA。
- 啟動節點作業憑證 (NOC) 的從屬 CA
- 建立節點作業憑證 (NOC)

啟動產品驗證授權單位 (PAA)

此 Java 範例顯示如何使用根據快取 _ API 傳遞 /V1 定義範本來建立和安裝產品證明的物質根 CA (PAA) 憑證。對於 PaaS 而言， AuthorityKeyIdentifier (AKI) 副檔名是選擇性的。若要設定 AKI，您必須產生一個以 BASE64 編碼的 AKI 值，並將其傳遞至. CustomExtension

此範例會呼叫下列 AWS 私有 CA API 動作：

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

如果您遇到問題，請參閱疑難排解一節使用物質標準中的。

```
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);
    }
}
```

啟用產品驗證中間 (PAI)

此 Java 範例顯示如何使用BlankSubordinate快取 _ 0_API通過/V PathLen 1 定義範本來建立和安裝產品證明的物件從屬 CA (PAI) 憑證。您必須產生 Base64 編碼的 KeyUsage 值，並將其傳遞至. CustomExtension

此範例會呼叫下列 AWS 私有 CA API 動作：

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

如果您遇到問題，請參閱疑難排解一節使用物質標準中的。

```
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);
    }
}
```

建立裝置驗證憑證 (DAC)

此 Java 範例顯示如何使用 BlankEndEntityCertificateCriticalBasicConstraints_ API 傳遞 /v1 範本來建立物件裝置驗證憑證。您必須產生 Base64 編碼的 KeyUsage 值，並將其傳遞至. CustomExtension

此範例會呼叫下列 AWS 私有 CA API 動作：

- IssueCertificate

如果您遇到問題，請參閱疑難排解一節使用物質標準中的。

```
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);
    }
}
```

為節點操作憑證 (NOC) 啟動根 CA。

此 Java 範例顯示如何使用根據快取 _ API 傳遞 /V1 定義範本來建立和安裝物件根 CA 憑證以發行 NOC。對於 NOC 根 CA 憑證而言， AuthorityKeyIdentifier (AKI) 擴充功能是選擇性的。若要設定 AKI，您必須產生一個以 BASE64 編碼的 AKI 值，並將其傳遞至. CustomExtension

此範例會呼叫下列 AWS 私有 CA API 動作：

- CreateCertificateAuthority
- GetCertificateAuthorityCsr
- IssueCertificate
- GetCertificate
- ImportCertificateAuthorityCertificate

如果您遇到問題，請參閱疑難排解一節使用物質標準中的。

```
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);
    }
```

```
}
```

啟動節點作業憑證 (NOC) 的從屬 CA

此 Java 範例顯示如何使用BlankSubordinate快取 _ 0_API通過/V PathLen 1 定義範本發行和安裝「事項從屬 CA 憑證」以發行 NOC。您必須產生 Base64 編碼的 KeyUsage 值，並將其傳遞至. CustomExtension

此範例會呼叫下列 AWS 私有 CA API 動作：

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

如果發生問題，請參閱疑難排解一節使用物質標準中的。

```
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);
    }
}
```

建立節點作業憑證 (NOC)

此 Java 範例顯示如何使用 BlankEndEntityCertificateCriticalBasicConstraints_ API 傳遞 /v1 範本來建立事項節點作業憑證。您必須產生 Base64 編碼的 KeyUsage 值，並將其傳遞至. CustomExtension

此範例會呼叫下列 AWS 私有 CA API 動作：

- IssueCertificate

如果您遇到問題，請參閱疑難排解一節使用物質標準中的。

```
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);
   }
}
```

使用 AWS 私有 CA API 實施移動駕駛執照（MDL）標準（Java 實例）

您可以使用 AWS Private Certificate Authority API 建立符合行動駕駛執照 (MDL) ISO/IEC 標準的憑證。本標準為執行與行動裝置相關的駕駛執照建立介面規格，包括憑證組態。

本節中的 Java 範例會透過傳送 HTTP 要求與服務互動。服務會傳回 HTTP 回應。如需詳細資訊，請參閱 AWS Private Certificate Authority API 參考。

除了 HTTP API 之外，您還可以使用AWS開發套件和AWS CLI工具進行管理AWS 私有 CA。我們建議您使用 SDK 或AWS CLI透過 HTTP API 來使用。如需詳細資訊，請參閱 Amazon Web Services 適用工具。下列主題說明如何使用 AWS SDK for Java 來程式設計 AWS 私有 CA API。

的 `GetCertificateAuthorityCsr`, `GetCertificate`, 和`DescribeCertificateAuthorityAuditReport`操作支持服務員. 您可以使用等待程式以根據特定資源的存在或狀態來控制程式碼的進度。如需詳細資訊，請參閱AWS開發人員部落格中的下列主題和 AWS SDK for Java 中的「服務員」。

主題

- 啟動發行機構憑證授權單位 (IACA) 憑證
- 建立文件簽署者憑證

啟動發行機構憑證授權單位 (IACA) 憑證

此 Java 範例顯示如何使用BlankRoot快取 _ 0_API通過/V PathLen 1 定義範本來建立和安裝符合 ISO/IEC MDL 標準的發行機構憑證授權單位 (IACA) 憑證。您必須為、和產生 base64 編碼的值 `KeyUsage` `IssuerAlternativeName` `CRLDistributionPoint`，並傳遞這些值。`CustomExtensions`

> **Note**
>
> IACA 連結憑證會建立從舊 IACA 根憑證到新 IACA 根憑證的信任路徑。發行機構可以在 IACA 重新密鑰過程中生成和分發 IACA 鏈接證書。您無法使用具有設定的 IACA 根憑證來發行 IACA 連結憑`pathLen=0`證。

此範例會呼叫下列 AWS 私有 CA API 動作：

- 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);
    }
}
```

建立文件簽署者憑證

此 Java 範例顯示如何使用 BlankEndEntityCertificate_APIP通/v1 範本來建立符合 ISO/ I EC MDL 標準的文件簽署者憑證。您必須為、產生 base64 編碼的值 KeyUsageIssuerAlternativeName，CRLDistributionPoint並傳遞這些值。CustomExtensions

此範例會呼叫下列 AWS 私有 CA API 動作：

- 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);
    }
}
```

外部簽署的私有 CA 憑證

如果您私有 CA 階層的信任根必須是 AWS 私有 CA 外部的 CA，則您可以建立並自我簽署您自己的根 CA。或者，您也可以取得由您組織營運之外部私有 CA 所簽署的私有 CA 憑證。無論其來源為何，您都可以使用此外部獲得的 CA 來簽署AWS 私有 CA管理的私有下屬 CA 憑證。

> **Note**
>
> 建立或取得外部信任服務提供者的程序不在本指南的範圍內。

搭配使用外部父項 CA，AWS 私有 CA可讓您依照 RFC 5280 的「名稱條件約束」區段中所定義強制執行 CA 名稱條件約束。名稱限制可讓 CA 管理員限制憑證中的主旨名稱。

如果您打算使用外部 CA 簽署私有下屬 CA 憑證，在AWS 私有 CA擁有工作 CA 之前，需要完成三項工作：

1. 產生憑證簽署要求 (CSR)。
2. 將 CSR 提交給您的外部簽署授權單位，並以簽署的憑證和憑證鏈結傳回。
3. 在中安裝已簽署的憑證AWS 私有 CA。

下列程序說明如何使用AWS Management Console或來完成這些工作AWS CLI。

取得並安裝外部簽署的 CA 憑證 (主控台)

1. (選擇性) 如果您尚未進入 CA 的詳細資料頁面，請在 https://console.aws.amazon.com/acm-pca/home 開啟AWS 私有 CA主控台。在 [私人憑證授權單位] 頁面上，選擇狀態為 [擱置憑證]、[作用中]、[停用] 或 [已過期] 的從屬 CA。
2. 選擇「動作」、「安裝 CA 憑證」以開啟「安裝從屬 CA 憑證」頁面。
3. 在 [安裝附屬 CA 憑證] 頁面的 [選取 CA 類型] 下，選擇 [外部私有 CA]。
4. 在此 CA 的 CSR 下，主控台會顯示 CSR 的以 BAS64 編碼的 ASCII 文字。您可以使用「複製」按鈕複製文字，也可以選擇將 CSR 匯出至檔案並將其儲存在本機。

   > **Note**
   >
   > 應對和貼上時，必須保留 CSR 文字的確切格式。

5. 如果您無法立即執行離線步驟從外部簽署授權單位取得已簽署的憑證，您可以在擁有簽署的憑證和憑證鏈結之後關閉該頁面並返回頁面。

   否則，如果您已準備就緒，請執行下列任一項作業：

   - 將憑證主體和憑證鏈結的 Base64 編碼 ASCII 文字貼到各自的文字方塊中。
   - 選擇上傳，將憑證主體和憑證鏈結從本機檔案載入各自的文字方塊中。

6. 選擇確認並安裝。

取得並安裝外部簽署的 CA 憑證 (CLI)

1. 使用get-certificate-authority-csr命令擷取私有 CA 的憑證簽署要求 (CSR)。如果您想要傳送 CSR 至顯示內容，請使用 `--output text` 選項，消除每一行結尾處的 CR/LF 字元。若要傳送 CSR 至檔案，請使用重新導向選項 (>)，後面加上檔案名稱。

   ```
   $ 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
   ```

   將 CSR 儲存為本機檔案後，您可以使用下列 OpenSSL 指令來檢查它：

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

   此命令會產生類似如下的輸出。請注意，CA 延伸為 TRUE，表示該 CSR 適用於 CA 憑證。

   ```
   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. 將 CSR 提交給您的外部簽署授權單位，並取得包含 Base64 PEM 編碼之已簽署憑證和憑證鏈結的檔案。

3. 使用指import-certificate-authority-certificate令將私有 CA 憑證檔案和鏈結檔案匯入到中AWS 私有 CA。

```
$ aws acm-pca import-certificate-authority-certificate \
--certificate-authority-arn arn:aws:acm-pca:region:account:\
certificate-authority/12345678-1234-1234-1234-123456789012 \
```

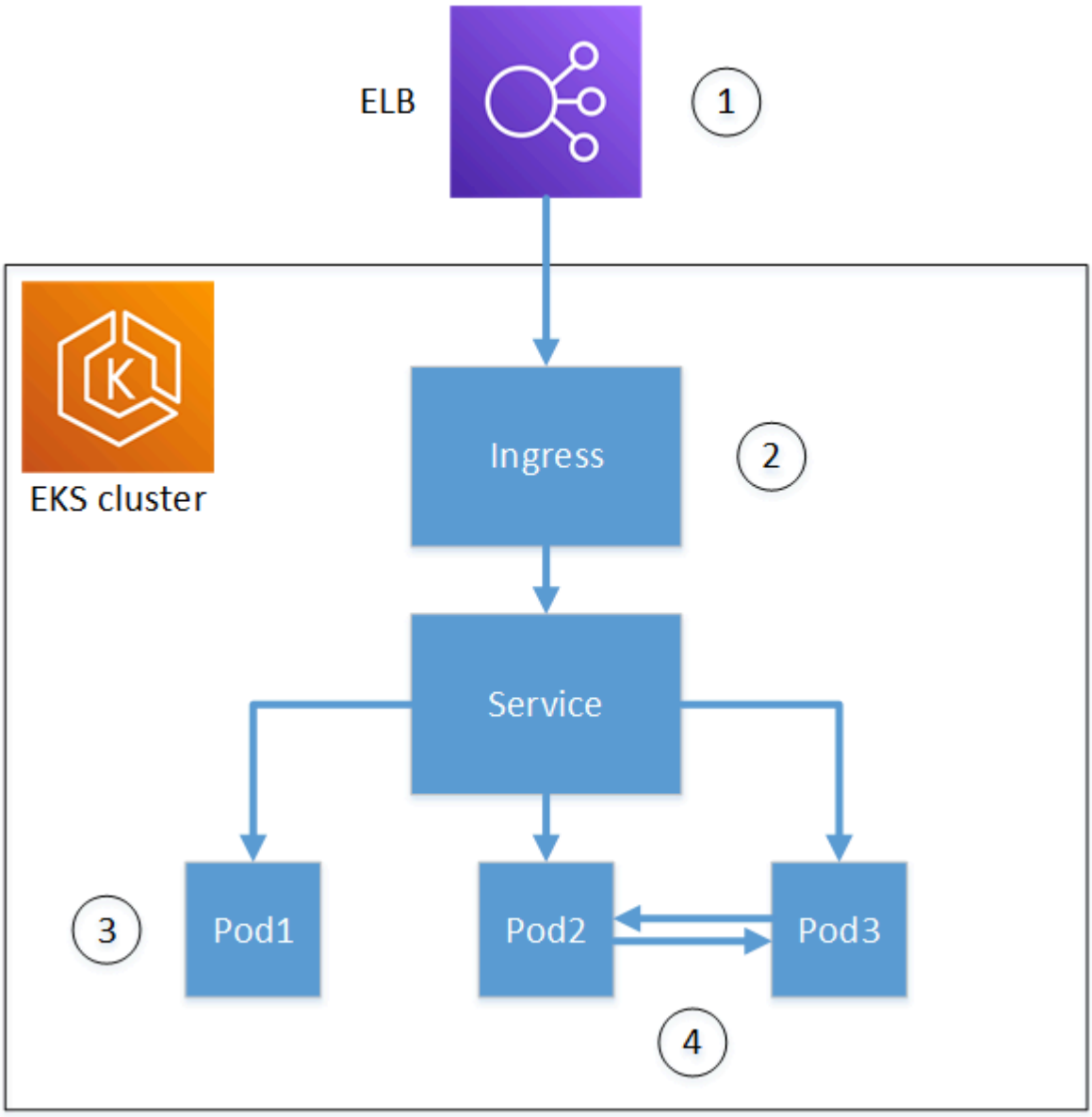
```
--certificate file://C:\example_ca_cert.pem \
--certificate-chain file://C:\example_ca_cert_chain.pem
```

使用保護庫伯尼特 AWS 私有 CA

Kubernetes 容器和應用程式會使用數位憑證，透過 TLS 提供安全驗證和加密。Kubernetes 中 TLS 憑證生命週期管理廣泛採用的解決方案是憑證管理員，這是 Kubernetes 的附加元件，可要求憑證、將憑證散佈至 Kubernetes 容器，以及自動執行憑證續約。

AWS 私有 CA為 cert-manager 提供開放原始碼外掛程式 aws-privateca-issuer，適用於想要設定 CA 而不在叢集中儲存私密金鑰的 cert-manager 使用者。具備控制 CA 作業存取與稽核之法規需求的使用者，可以使用此解決方案來改善稽核性並支援合規性。您可以將AWS私有 CA 發行者外掛程式搭配使用 Amazon Elastic Kubernetes Service (Amazon EKS)、自我管理的 Kubernetes，或在現場部署 Kubernetes 中使用。AWS

下圖顯示在 Amazon EKS 叢集中使用 TLS 時可用的一些選項。此範例叢集包含各種資源，位於負載平衡器後方。這些數字可識別 TLS 保護通訊的可能端點，包括外部負載平衡器、輸入控制器、服務中的個別網繭，以及一對網繭彼此安全通訊。



1. 在負載平衡器終止。

   Elastic Load Balancing (ELB) 是一種AWS Certificate Manager整合式服務，這表示您可以使用私有 CA 佈建 ACM、使用它簽署憑證，然後使用 ELB 主控台進行安裝。此解決方案提供遠端用戶端與負載平衡器之間的加密通訊。資料會以未加密的方式傳遞至 EKS 叢集。或者，您可以提供私有憑證給非AWS負載平衡器來終止 TLS。

2. 在 Kubernetes 入口控制器處終止。

   入口控制器作為原生負載平衡器和路由器位於 EKS 叢集內部。如果您已同時安裝 cert-manager 和 aws-privateca-issuer，並使用私有 CA 佈建叢集，Kubernetes 可以在控制器上安裝已簽署的 TLS

憑證，讓它作為叢集的外部通訊端點。負載平衡器與入口控制器之間的通訊會加密，而在輸入後，資料會以未加密的方式傳送至叢集的資源。

3. 在網繭上終止。

   每個網繭都是一或多個共用儲存區和網路資源的容器群組。如果您已同時安裝憑證管理員和並aws-privateca-issuer使用私有 CA 佈建叢集，Kubernetes 可以視需要在網繭上安裝已簽署的 TLS 憑證。根據預設，叢集中的其他網繭無法使用終止於網繭的 TLS 連線。

4. 網繭之間的安全通訊。

   您也可以使用憑證佈建多個網繭，讓它們彼此通訊。可能的情況如下：

   - 使用 Kubernetes 產生的自我簽署憑證進行佈建。這可保護網繭之間的通訊安全，但自我簽署憑證不符合 HIPAA 或 FIPS 需求。
   - 使用私有 CA 簽署的憑證進行佈建。與上面的數字 2 和 3 一樣，這需要同時安裝 cert-manager 和 aws-privateca-issuer，並使用私有 CA 佈建叢集。然後，Kubernetes 可以視需要在網繭上安裝已簽署的 TLS 憑證。

跨帳戶使用憑證管理員

具有 CA 跨帳戶存取權的系統管理員可以使用憑證管理員來佈建 Kubernetes 叢集。如需詳細資訊，請參閱 跨帳戶存取私有 CA 的安全性最佳做法。

> **Note**
>
> 只有特定AWS 私有 CA憑證範本可用於跨帳戶案例。the section called “支援的憑證範本 ”如需可用範本的清單，請參閱。

支援的憑證範本

下表列出可與憑證管理員搭配使用以佈建 Kubernetes 叢集的AWS 私有 CA範本。

庫伯氏支援的範本	Support 跨帳戶使用
BlankEndEntityCertificate_企業社會責任通過/V1 定義	
CodeSigningCertificate/V1 定義	

庫伯氏支援的範本	Support 跨帳戶使用
EndEntityCertificate/V1 定義	✓
EndEntityClientAuthCertificate/V1 定義	✓
EndEntityServerAuthCertificate/V1 定義	✓
OCSP 定SigningCertificate義	

範例解決方

下列整合解決方案示範如何在 Amazon EKS 叢集AWS 私有 CA上設定對的存取權限。

- 使用和 Amazon EKS 啟用 TLS 的庫伯尼特群集 AWS 私有 CA
- 使用新的 AWS Load Balancer 控制器在 Amazon EKS 上設定 end-to-end TLS 加密

AWS 私有 CA Connector for Active Directory

什麼是活動目錄的AWS Private CA連接器

AWS Private CA可以發行和管理所需的憑證AWS Managed Microsoft AD。使用 Active Directory 的 AWS 私有 CA連接器 (AD 連接器)，您可以將內部部署企業或其他協力廠商 CA 取代為您擁有的受管理私有 CA，從而為 AD 管理的使用者、群組和機器提供憑證註冊。

您可以使用 AD 的連接器，藉由將 AD 和公開金鑰基礎結構移轉至雲端，以消除內部部署基礎結構。AWS Managed Microsoft AD對於希望與內部部署 AD AWS Private CA 搭配使用的客戶，此功能也與 AWS Managed Microsoft AD Connector 整合。

主題

- 您是 AD 使用者的首次使用連接器嗎？
- 存取 AD 的連接器
- AD 連接器的定價

您是 AD 使用者的首次使用連接器嗎？

如果您是 AD 連接器的第一次使用者，建議您先閱讀下列章節：

- 什麼是 AWS 私有 CA？
- 什麼是 AWS Directory Service？

存取 AD 的連接器

您可以透過主控台和 API 存取 AD 的連接器。AWS CLI您可以從主控台、主控台存取AWS Private CA主控台中的AWS Directory Service連接器，或在搜尋列中AWS Management Console搜尋 AD 的連接器。

AD 連接器的定價

AD 連接器是作為一項功能提供，無需額外費用。AWS 私有 CA您只需支付私有憑證授權單位以及透過這些授權單位發行的憑證費用。

如需最新AWS 私有 CA定價資訊，請參閱AWS Private Certificate Authority定價。您也可以使用定AWS價計算器來估算成本。

開始使用作用中目錄的AWS Private CA連接器

使用適用於使用中目錄的AWS Private CA連接器，您可以將憑證從您的私人 CA 發行至您的作用中目錄物件，以進行驗證和加密。建AWS Private Certificate Authority立連接器時，請在 VPC 中為您建立端點，讓目錄物件要求憑證。

若要發行憑證，請為連接器建立連接器和 AD 相容範本。建立範本時，您可以為 AD 群組設定註冊權限。

主題

- AD 先決條件的連接器
- 建立連接器
- 設定 AD 政策
- 建立範本
- 管理 AD 群組權限

AD 先決條件的連接器

AD 的連接器需要以下內容。

若要建立 AD 的連接器，您需要設定 AWS Private Certificate Authority (CA) 和目錄。然後，您需要與 AD 服務的連接器共用私有 CA 和目錄。您還需要正確設定安全群組和 IAM 政策，才能建立連接器。

AWS 私有 CA

設定AWS 私有 CA用於核發憑證給您的目錄物件。如需詳細資訊，請參閱 私人 CA 管理。

AWS 私有 CA必須處於`Active`狀態才能建立 AD 的連接器。私有 CA 的主旨名稱必須包含一般名稱。如果您嘗試使用沒有通用名稱的私有 CA 建立連接器，連接器建立將會失敗。

Active Directory

除了之外AWS 私有 CA，您還需要虛擬私有雲（VPC）中的活動目錄。AD 連接器支援下列提供的目錄類型AWS Directory Service：

- AWS託管 Microsoft 活動目錄：隨著AWS Directory Service你可以運行 Microsoft 活動目錄（AD）作為託管服務。 AWS Directory Service for Microsoft Active Directory也稱為AWS Managed

Microsoft AD，是由視窗服務器 2019 年供電。使用AWS Managed Microsoft AD，您可以在中執行目錄感知工作負載AWS 雲端，包括 Microsoft SharePoint 和自訂 .NET 和 SQL 伺服器型應用程式。

- 使用中的目錄連接器：AD Connector 是一個目錄閘道，可以將目錄要求重新導向至您的內部部署 Microsoft Active Directory，而不會快取雲端中的任何資訊。AD Connector 支援連線到 Amazon EC2 上託管的網域

> **Note**
>
> 使用 AD 的連接器時，不支援註冊網域控制站。AWS Managed Microsoft AD

服務帳戶

使用 Directory Service AD Connector 時，您需要委派其他權限給服務帳戶。在服務帳戶上設置訪問控制列表（ACL）以允許以下功能：

- 新增及移除服務主要名稱 (SPN) 至本身
- 在以下容器中建立並更新憑證授權機構：

```
#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
```

- 建立並更新 NT AuthCertificates 憑證授權單位 (CA) 物件。附註：如果 NT AuthCertificates CA 物件存在，您必須委派它的權限。如果物件不存在，您必須委派在公開金鑰服務容器上建立子物件的能力。

```
#objects
CN=NTAuthCertificates,CN=Public Key Services,CN=Services,CN=Configuration
```

> **Note**
>
> 如果您正在使用，AWS Managed Microsoft AD則當您使用目錄授權 AD 服務的連接器時，會自動委派其他權限。您可以略過此先決條件步驟。

您可以使用此 PowerShell 指令碼委派其他權限。它會建立 NT AuthCertifiates 憑證授權單位物件。將「我的連接器帳戶」替換為服務帳戶名稱。

```
$AccountName = 'myconnectoraccount'
# DO NOT modify anything below this comment.
# Getting Active Directory information.
Import-Module -Name 'ActiveDirectory'
$RootDSE = Get-ADRootDSE

# Getting AD Connector service account Information
$AccountProperties = Get-ADUser -Identity $AccountName
$AccountSid = New-Object -TypeName 'System.Security.Principal.SecurityIdentifier'
 $AccountProperties.SID.Value
[System.GUID]$ServicePrincipalNameGuid = (Get-ADObject -SearchBase
 $RootDse.SchemaNamingContext -Filter { lDAPDisplayName -eq 'servicePrincipalName' } -
Properties 'schemaIDGUID').schemaIDGUID
$AccountAclPath = $AccountProperties.DistinguishedName

# Getting ACL settings for AD Connector service account.
$AccountAcl = Get-ACL -Path "AD:\$AccountAclPath"

# Setting ACL allowing the AD Connector service account the ability to add and remove a
 Service Principal Name (SPN) to itself
$AccountAccessRule = New-Object -TypeName
 'System.DirectoryServices.ActiveDirectoryAccessRule' $AccountSid, 'WriteProperty',
 'Allow', $ServicePrincipalNameGuid, 'None'
$AccountAcl.AddAccessRule($AccountAccessRule)
Set-ACL -AclObject $AccountAcl -Path "AD:\$AccountAclPath"

# Add ACLs allowing AD Connector service account the ability to create certification
 authorities
[System.GUID]$CertificationAuthorityGuid = (Get-ADObject -SearchBase
 $RootDse.SchemaNamingContext -Filter { lDAPDisplayName -eq 'certificationAuthority' }
 -Properties 'schemaIDGUID').schemaIDGUID
$CAAccessRule = New-Object -TypeName
 'System.DirectoryServices.ActiveDirectoryAccessRule' $AccountSid,
 'ReadProperty,WriteProperty,CreateChild,DeleteChild', 'Allow',
 $CertificationAuthorityGuid, 'None'
$PKSDN = "CN=Public Key Services,CN=Services,CN=Configuration,
$($RootDSE.rootDomainNamingContext)"
$PKSACL = Get-ACL -Path "AD:\$PKSDN"
$PKSACL.AddAccessRule($CAAccessRule)
Set-ACL -AclObject $PKSACL -Path "AD:\$PKSDN"
```

```
$AIADN = "CN=AIA,CN=Public Key Services,CN=Services,CN=Configuration,
$($RootDSE.rootDomainNamingContext)"
$AIAACL = Get-ACL -Path "AD:\$AIADN"
$AIAACL.AddAccessRule($CAAccessRule)
Set-ACL -AclObject $AIAACL -Path "AD:\$AIADN"

$CertificationAuthoritiesDN = "CN=Certification Authorities,CN=Public Key
 Services,CN=Services,CN=Configuration,$($RootDSE.rootDomainNamingContext)"
$CertificationAuthoritiesACL = Get-ACL -Path "AD:\$CertificationAuthoritiesDN"
$CertificationAuthoritiesACL.AddAccessRule($CAAccessRule)
Set-ACL -AclObject $CertificationAuthoritiesACL -Path "AD:\$CertificationAuthoritiesDN"

$NTAuthCertificatesDN = "CN=NTAuthCertificates,CN=Public Key
 Services,CN=Services,CN=Configuration,$($RootDSE.rootDomainNamingContext)"
If (-Not (Test-Path -Path "AD:\$NTAuthCertificatesDN")) {
New-ADObject -Name 'NTAuthCertificates' -Type 'certificationAuthority' -OtherAttributes
 @{certificateRevocationList=[byte[]]'00';authorityRevocationList=[byte[]]'00';cACertificate=[b
 -Path "CN=Public Key Services,CN=Services,CN=Configuration,
$($RootDSE.rootDomainNamingContext)" }

$NTAuthCertificatesACL = Get-ACL -Path "AD:\$NTAuthCertificatesDN"
$NullGuid = [System.GUID]'00000000-0000-0000-0000-000000000000'
$NTAuthAccessRule = New-Object -TypeName
 'System.DirectoryServices.ActiveDirectoryAccessRule' $AccountSid,
 'ReadProperty,WriteProperty', 'Allow', $NullGuid, 'None'
$NTAuthCertificatesACL.AddAccessRule($NTAuthAccessRule)
Set-ACL -AclObject $NTAuthCertificatesACL -Path "AD:\$NTAuthCertificatesDN"
```

IAM 政策

若要建立 AD 連接器，您需要可讓您建立連接器資源的 IAM 政策、與 AD 服務連接器共用私有 CA，以及授權 AD 服務的連接器與目錄中的 AD 服務連接器。

以下是使用者管理策略的範例：

```
{
    "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/*"
        }
    ]
}
```

AD 的連接器需要額外的AWS RAM權限，主控台和命令列使用。

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

```
                "ram:ListResourceSharePermissions",
                "ram:ListResourceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 私有 CA與連接器共用 AD

您將需要使用服務主體共用AWS Private CA與連接器AWS Resource Access Manager服務共用。

當您在AWS主控台中建立連接器時，會自動為您建立資源共用。

當您使用建立資源共用時AWS CLI，您將使用AWS RAMcreate-resource-share指令。

下列指令會建立資源共用：

```
$  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
```

呼叫的服務主體 CreateConnector 具有 PCA 的憑證發行權限。若要防止使用 AD Connector 的服務主體具有AWS 私有 CA資源的一般存取權，請使`CalledVia`用來限制其權限。

使用您的目錄授權 AD 的連接器

您可以使用目錄授權 AD 服務的連接器，以便連接器可以與您的目錄進行通訊。若要授權 AD 服務的連接器，請建立目錄註冊。如需建立目錄註冊的詳細資訊，請參閱 管理目錄註冊

安全群組

您的 VPC 和 AD 連接器連接器之間的通訊是通過的AWS PrivateLink，因此需要具有輸入規則的安全群組，該安全群組會在 VPC 上開啟連接埠 443 TCP 和 UDP。當您建立連接器時，系統會要求您提供此安全性群組。您可以將來源指定為自訂，然後選取 VPC 的 CIDR 區塊。您可以選擇進一步限制此項目 (例如 IP、CIDR 和安全群組識別碼)。

建立連接器

如需指示，請參閱程序 建立連接器

設定 AD 政策

AD 的連接器無法檢視或管理客戶的群組原則物件 (GPO) 設定。GPO 會控制 AD 要求傳送至客戶或其他驗證AWS 私有 CA或憑證自動販賣伺服器的路由。無效的 GPO 組態可能會導致您的要求路由不正確。客戶需要針對 AD 組態設定和測試連接器。

群組原則與連接器相關聯，您可以選擇為單一 AD 建立多個連接器。如果每個連接器的群組原則組態不同，您可以自行決定管理其存取控制。

資料層呼叫的安全性取決於 Kerberos 和您的 VPC 組態。任何具有 VPC 存取權的人都可以進行資料平面呼叫，只要對應 AD 進行驗證即可。這存在於範圍之外， AWSAuth 而且管理授權和身份驗證取決於您，即客戶。

在使用中目錄中，依照下列步驟建立指向建立連接器時所產生之 URI 的 GPO。若要從主控台或命令列使用 AD 的連接器，必須執行此步驟。

設定 GPO。

1. 在 DC 上開啟伺服器管理員
2. 移至 [工具]，然後選擇主控台右上角的 [群組原則管理]。
3. 移至樹系 > 網域。選擇您的域名，然後右鍵單擊您的域名。選取 [在此網域中建立 GPO]，然後在此連結... 並輸入PCA GP0名稱。
4. 新建立的 GPO 現在會列在您的網域名稱下方。
5. 選擇 PCA GPO，然後選取編輯。如果開啟一個對話方塊，並顯示警示訊息這是一個連結，且變更將會全域傳播，請確認訊息是否繼續。群組原則管理編輯器應該會開啟。
6. 在 [群組原則管理編輯器] 中，移至 [電腦設定] > [原則] > [Windows 設定] > [安全性設定] > [公用金鑰原則] (選擇資料夾)。
7. 移至物件類型，然後選擇憑證服務用戶端-憑證註冊原則
8. 在選項中，將「組態模型」變更為「啟用」。
9. 確認已核取並啟用使用中目錄註冊原則。選擇新增。
10. 「憑證註冊原則伺服器」視窗應該會開啟。
11. 在 [輸入註冊伺服器原則 URI] 欄位中，輸入您建立連接器時所產生的憑證註冊原則伺服器端點。
12. 將「驗證類型」保留為「Windows 整合式」。

13. 選擇「驗證」。驗證成功後，選取 [新增]。對話方塊將關閉。
14. 返回憑證服務用戶端-憑證註冊原則，並核取新建立的連接器旁邊的方塊，以確保連接器是預設的註冊原則
15. 選擇使用中目錄註冊原則，然後選取移除。
16. 在確認撥號方塊中，選擇 [是] 以刪除 LDAP 型驗證。
17. 在憑證服務用戶端 > 憑證註冊原則視窗上選擇套用和確定，然後關閉它。
18. 移至 [公用金鑰原則] 資料夾，然後選擇 [憑證服務用戶端-自動註冊]。
19. 將「組態模型」選項變更為「啟用」。
20. 確認已核取「更新過期憑證」和「更新憑證」。保留其他設置，因為它們。
21. 選擇「套用」，然後選擇 「確定」，然後關閉對話方塊。

接下來設定使用者組態的公開金鑰原則。移至「使用者設定 > 原則 > Windows 設定 > 安全性設定 > 公開金鑰原則」。遵循步驟 6 到步驟 21 概述的程序，設定使用者組態的公開金鑰原則。

完成 GPO 和公開金鑰原則的設定之後，網域中的物件會從 AD 的AWS 私有 CA連接器要求憑證，並取得由AWS 私有 CA發行的憑證。

建立範本

如需指示，請參閱程序 建立連接器範本

管理 AD 群組權限

如需指示，請參閱程序 管理範本的 AD 群組和權限

AWS 私有 CA作用中目錄程序的連接器

本節中的程序說明如何建立連接器、設定範本，以及與AWS 私有 CA和 Active Directory 整合。您可以從 AD 用AWS 私有 CA連接器主控台、使用的 AD 連接器區段AWS CLI，或使用 AD API 的AWS 私有 CA連接器來執行這些作業。

> **Note**
>
> 雖然 AD 的AWS 私有 CA連接器與密切整合AWS 私有 CA，但這兩個服務有不同的 API。如需詳細資訊，請參閱 AWS Private Certificate AuthorityAPI 參考和作用中目錄 API 參考的AWS 私有 CA連接器。

程序

- 建立連接器
- 建立連接器範本
- 列出作用中目錄的連接器
- 列出連接器範本
- 檢視連接器詳情
- 檢視連接器範本詳細
- 管理目錄註冊
- 管理範本的 AD 群組和權限
- 設定服務主體名稱
- 標記 AD 資源的連接器

建立連接器

使用下列程序，使用主控台、命令列或 API 來建立連接器作用中目錄的AWS Private CA連接器。

建立連接器 (主控台)

完成下列程序，以使用AWS主控台建立和設定連接器。

任務

- 打開控制台
- 開啟建立連接器
- 選擇或建立目錄
- 選擇私人 CA
- 設定標記
- 檢閱和建立

打開控制台

登入您的AWS帳戶，然後開啟適用於作用中目錄的AWS Private CA連接器主控台，位置為**https://console.aws.amazon.com/pca-connector-ad/home**。

開啟建立連接器

在 [第一次服務登陸頁面] 或 [Active Directory 的連接器] 頁面上，選擇 [建立連接器]。

選擇或建立目錄

在 [建立作用中目錄的專用 CA 連接器] 頁面上，提供 [作用中目錄] 段落的資訊。

- 在 [選取您的使用中目錄類型] 下，選擇下列兩種可用類型之一：
  - AWS Directory Service for Microsoft Active Directory— 指定由管理的作用中目錄AWS Directory Service。
  - 具有 AWS AD Connector 的內部部署作用中目錄 — 使用 AD Connector 存取您主控內部部署的作用中目錄。
- 在「選取您的目錄」下，從清單中選擇您的目錄。

  或者，您也可以選擇 [建立目錄]，在新視窗中開啟AWS Directory Service主控台。當您完成建立新目錄時，請返回 Active Directory 主控台的AWS Private CA連接器，並重新整理目錄清單。您的新目錄應該可供選擇。

  > **Note**
  >
  > 建立目錄時，請注意，AD 的 Connector 僅支援AWS Directory Service主控台提供的下列目錄類型：
  >
  > - AWS Managed Microsoft AD
  > - AD Connector

- 在為 VPC 端點選取安全群組下，從清單中選擇安全群組。

  或者，您可以選擇建立安全群組，在新視窗中開啟 Amazon EC2 主控台至建立安全群組頁面。當您完成建立安全性群組時，請返回 Active Directory 的AWS Private CA連接器主控台，並重新整理安全性群組清單。您的新安全性群組應該可供選取。

選擇私人 CA

在私人憑證授權單位區段中，從清單中選擇私有 CA。

或者，您可以選擇建立私人 CA，在新視窗中開啟「私人憑證授權單位」頁面的AWS 私有 CA主控台。當您完成建立 CA 時，請返回作用中目錄主控台的AWS Private CA連接器，並重新整理 CA 清單。您的新 CA 應該可供選擇。

設定標記

在 [標籤 — 選用] 窗格中，您可以在 AD 資源上套用和移除中繼資料。標籤是索引鍵值字串配對，其中索引鍵必須是資源唯一的，而且值是選用的。窗格會在表格中顯示資源的任何現有標籤。支援以下動作。

- 選擇「管理標籤」以開啟「管理標籤」頁面。
- 選擇「新增標籤」以建立標籤。填寫「關鍵字」欄位，並選擇性地填入「值」欄位。選擇「儲存變更」以套用標籤。
- 選擇標記旁邊的「移除」按鈕，將其標示為要刪除，然後選擇「儲存變更」以確認。

檢閱和建立

提供必要資訊並檢閱您的選擇之後，請選擇 [建立連接器]。這會開啟 Active Directory 的連接器詳細資料頁面，您可以在其中檢視連接器建立時的進度。

建立連接器的程序完成之後，請為其指派服務主體名稱。

為作用中目錄建立連接器 (AWS CLI)

若要使用 CLI 建立 Active Directory 的連接器，請使用的作用中目錄的連接器區段中的建立AWS Private CA連接器命令。AWS CLI

為使用中目錄 (API) 建立連接器

若要使用 API 建立作用中目錄的連接器，請使用 CreateConnector作用中目錄 API AWS Private CA 連接器中的動作。

建立連接器範本

建立連接器範本 (主控台)

完成下列程序，以使用AWS控制台。

任務

- 打開控制台
- 選擇連接器
- 尋找範本區段

- 模板創建方法
- 範本設定
- 設定憑證設定
- 設定要求處理和註冊設定
- 設定金鑰使用量延伸
- 指派應用程式原
- 配置自定義應用程序
- 設定密碼編譯設定
- 設定群組和權限
- 設定取代範本
- 設定標記
- 檢閱和建立

打開控制台

登入您的AWS帳戶並打開AWS Private CA作用中的目錄主控台的連接器**https://console.aws.amazon.com/pca-connector-ad/home**。

選擇連接器

從中選擇連接器作用中目錄的連接器列出，然後選擇查看詳情。

尋找範本區段

在連接器的詳細資料頁面上，找到模板部分，然後選擇建立範本。

模板創建方法

在「」建立範本頁面，在」模板創建方法」區段中，選擇其中一個方法選項。

- 從預先定義的樣板開始(預設值) — 從 AD 應用程式的預先定義範本清單中選擇：
  - 程式碼簽章
  - 计算机
  - 網域控制站驗證
  - EFS 復原代理
  - 註冊代理

- 註冊代理程式 (電腦)
- IPSec
- 凱伯洛斯身份驗證
- RAS 和 IAS 伺服器
- 智慧卡登入
- 信任清單簽署
- 使用者簽名
- 工作站驗證

- 從您建立的現有範本開始— 從您之前創建的自定義模板列表中進行選擇。
- 從空白範本開始— 選擇此選項可開始建立全新的範本。

範本設定

在範本設定」區段中，提供下列資訊：

- 範本名稱— 範本的名稱
- 模板架構版本— 範本的結構描述版本。結構描述版本會影響範本選項的可用性，如下所示：

  結構描述版本 2

  - 支援視窗 XP /視窗伺服器2003 及更高版本的用戶端相容性。
  - 僅支援傳統密碼編譯服務提供者。

  結構描述版本 3

  - 支援視窗 Vista /視窗伺服器2008 及更高版本的用戶端相容性。
  - 支援允許要求者使用現有金鑰續約。
  - 僅支援金鑰儲存區提供者。

  結構描述版本 4

  - 支持視窗 8 /視窗服務器2012 及更高版本的客戶端兼容性。
  - 支援允許要求者使用現有金鑰續約。
  - 支援傳統加密服務提供者和金鑰儲存區提供者。
- 用戶端相容— 與範本相容的最低作業系統層級。選擇其中一個列出的選項：
  - 視窗 XP /視窗伺服器 2003
  - 視窗遠景/視窗伺服器 2008

- 視窗 7/ 視窗伺服器 2008 R2
- 視窗 8 及更高版本/視窗伺服器 2012
- 視窗 8 及以上/視窗伺服器 2012 R2
- 視窗 8 及更高版本/視窗伺服器 2016 及以上

設定憑證設定

在憑證設定」區段中，針對以此範本為基礎的憑證定義下列設定。

- 證書類型— 指定是否要建立使用者或者计算机證書。
- 自動註冊— 選擇是否根據此範本啟用憑證的自動註冊。
- 有效期— 將憑證有效期間指定為小時、天、週、月或年的整數值。最小值為 2 小時。
- 更新期— 將憑證續約期間指定為小時、天、週、月或年的整數值。續期必須不超過有效期的 75%。
- 主旨名稱— 根據 Active Directory 中包含的資訊，選擇要包含在主旨名稱中的一個或多個選項。

> **Note**
>
> 至少必須指定一個主體名稱或主體替代名稱選項。

  - 通用名稱
  - DNS 作為一般名稱
  - 目錄路徑
  - 電子郵件
- 主體替代名稱— 根據 Active Directory 中包含的資訊，選擇要包含在主旨替代名稱中的一或多個選項。

> **Note**
>
> 至少必須指定一個主體名稱或主體替代名稱選項。

  - 目錄指南
  - DNS 名稱
  - 域名 DNS 服務

- 電子郵件
- 服務主要名稱 (SPN)
- 使用者主體名稱 (UPN)

設定要求處理和註冊設定

在憑證要求處理和註冊選項」區段中，根據範本指定憑證的用途，然後選擇下列其中一個選項。

- Signature
- 加密
- 簽名和加密
- 簽名和智慧卡登入

接下來，選擇要啟動的下列功能之一。選項會根據憑證用途而有所不同。

- 刪除無效的憑證 (不封存)
- 包括對稱演算法
- 可匯出私密金鑰

最後，選擇憑證註冊選項。選項會根據憑證用途而有所不同。

- 無需使用者輸入
- 註冊期間提示使用者
- 註冊期間提示使用者並要求使用者輸入

設定金鑰使用量延伸

在金鑰使用率延伸設定部分中，選擇使用簽名和加密金鑰使用的選項。

簽章金鑰用法

- 數位簽章
- 簽名是原產地證明（不可否認）

加密金鑰使用

- 允許在不加密金鑰的情況下交換金鑰 (金鑰合約
- 僅允許使用金鑰加密進行金鑰交換 (金鑰加密)
- 允許加密使用者資料 (資料加密)

您也可以選擇將金鑰使用擴充功能設定對於這兩種類型的密鑰。

指派應用程式原

在申請政策」段落中，選擇所有套用的應用程式原則。可用的策略會列在數個頁面中。某些策略可能會因為先前的設定而被預先選取。

配置自定義應用程序

在自定義應用程序區段中，您可以將自訂 OID 新增至範本，並指定應用程式原則延伸是否為重要。

設定密碼編譯設定

在密碼編譯設定」段落中，針對以此樣板為基礎的憑證選擇下列密碼編譯設定類別。

1. 區段頂端的內容由模板創建方法 和範本設定 您以前選擇的。

   - 如果您接受預設值模板版本 2在範本設定 ，則會在此處顯示下列狀態訊息：
     - 密碼編譯提供者類別
     - 傳統密碼編譯服務提供者

     在這種情況下，沒有要配置的設置，您可以繼續進行下一步。
   - 如果您指定模板版本 3在範本設定 ，則會在此處顯示下列狀態訊息：
     - 密碼編譯提供者類別
     - 金鑰儲存提供者

     您還必須選擇一個金鑰演算法從列出的選項EDH_P256,EDH_P384,EDH_P521，以及RSA。
   - 如果您指定模板版本 4在範本設定 ，那麼你必須在一個之間進行選擇金鑰儲存提供者和一個傳統密碼編譯服務提供者。如果您選擇密鑰存儲提供，然後是金鑰演算法還必須從列出的選項中選擇EDH_P256,EDH_P384,EDH_P521，以及RSA。
2. 金鑰大小下限 (位元)— 指定最小金鑰大小。此設定會影響可用的密碼編譯提供者。
3. 選擇要求可使用的密碼編譯提供者— 選擇兩個可用選項之一：

   - 請求可以使用主題計算機上可用的任何提供程序

- 請求必須使用下列其中一個選取的提供者

  選擇此選項會開啟一個密碼編譯提供者列表。您可以使用中的按鈕選取提供者並排定優先順序訂單欄。支援下列提供者：

  - 微軟基礎加密提供程序 V1.0
  - 微軟基地 DSS 和迪菲-赫爾曼密碼提供商
  - 微軟基礎智能卡加密提供商
  - 微軟 DH 通道密碼提供商
  - 微軟增強型加密提供程序 V1.0
  - 微軟增強型 DSS 和迪菲-赫爾曼密碼編譯提供商
  - 微軟增強型 RSA 和 AES 密碼編譯提供商
  - 微軟 RSA 通道密碼編譯提供商

設定群組和權限

在群組和權限區段中，您可以檢視範本現有的群組和註冊權限，或者您可以選擇新增群組和權限按鈕添加一個新的。按鈕會開啟需要下列資訊的表單：

- 顯示名稱
- 安全標識符(西德)
- 注册，具有選項允許 | 拒絕 | 未設置
- 自動註冊，具有選項允許 | 拒絕 | 未設置

設定取代範本

在取代模板部分中，您可以通知活動目錄當前模板取代 AD 中創建的一個或多個模板。選擇以套用取代範本從作用中目錄新增範本以取代並指定取代範本的一般名稱。

設定標記

在標籤-可選窗格中，您可以在 AD 資源上應用和刪除元數據。標籤是索引鍵值字串配對，其中索引鍵必須是資源唯一的，而且值是選用的。窗格會在表格中顯示資源的任何現有標籤。支援以下動作。

- 選擇管理標籤以開啟管理標籤頁面。
- 選擇「新增標籤」以建立標籤。填寫「」關鍵欄位和 (選擇性)價值欄位。選擇儲存變更以套用標籤。

- 選擇合適的移除標籤旁邊的按鈕將其標記為要刪除，然後選擇儲存變更以確認。

檢閱和建立

提供必要資訊並檢閱您的選擇後，請選擇建立範本。這將打開範本詳情，您可以在其中檢閱新範本的設定、編輯或刪除範本、管理群組和權限、管理已取代的範本、管理標籤，以及為憑證持有者設定自動重新註冊。

建立連接器範本 (CLI)

使用創建模板中的指令AWS Private CA的「作用中目錄」段落的連接器AWS CLI。

建立連接器範本 (API)

使用 CreateTemplate中的動作AWS Private CA活動目錄 API 的連接器。

列出作用中目錄的連接器

您可以使用AWS Private CA作用中目錄主控台的連接器或AWS CLI以列出您擁有的連接器。

使用主控台列出連接器

1. 登入您的AWS帳戶並打開AWS Private CA作用中的目錄主控台的連接器`https://console.aws.amazon.com/pca-connector-ad/home`。
2. 檢閱中的資訊作用中目錄的連接器列表。您可以使用右上角的頁碼瀏覽連接器的多個頁面。依預設，每個連接器都會佔用一列，顯示下列資訊欄。

- 接頭識別碼— 連接器的唯一 ID。
- 目錄名稱— 與連接器相關聯的作用中目錄資源。
- 連接器狀態— 連接器狀態。可能的值為：創建|作用中|刪除|失敗。
- 服務主體名稱狀態— 與連接器相關聯的服務主體名稱 (SPN) 狀態。可能的值為：創建|作用中|刪除|失敗。
- 目錄註冊狀態— 副董事的註冊狀況。可能的值為：創建|作用中|刪除|失敗。
- 建立於— 建立連接器時的時間戳記。

選擇主機右上角的齒輪圖示，您可以使用頁面大小偏好。

若要使用列出您的連接器AWS CLI

使用列表連接器命令列出您的連接器。

若要使用 API 列出您的連接器

使用ListConnectors中的動作AWS Private CA作用中目錄 API 的連接器。

列出連接器範本

您可以使用AWS Private CA作用中目錄主控台的連接器或AWS CLI以列出您擁有之連接器的樣板。連接器樣板是以AWS Private CA BlankEndEntityCertificate_API通過/V1範本。

使用主控台刊登範本

1. 登入您的AWS帳戶並打開AWS Private CA作用中的目錄主控台的連接器**https://console.aws.amazon.com/pca-connector-ad/home**。
2. 選擇一個連接器作用中目錄的連接器列出，然後選擇查看詳情。
3. 在連接器詳細資料頁面上，檢閱模板部分。您可以使用右上角的頁碼瀏覽多個範本頁面。每個範本都會佔用一列，顯示下列資訊欄。

- 範本名稱— 使用者可讀取的範本名稱。
- 範本狀態— 範本的狀態。可能的值為：作用中|刪除。
- 範本識別碼— 範本的唯一識別元。

若要使用AWS CLI

使用列表模板指令以列示指定連接器的樣板。

若要使用 API 刊登範本

使用 ListTemplates中的動作AWS Private CA用於列出指定連接器之範本的作用中目錄 API 的連接器。

檢視連接器詳情

使用下列程序在主控台、命令列或 API 中檢視連接器的組態詳細資料AWS Private CA作用中目錄的連接器。

查看連接器（控制台）

若要檢視連接器 (主控台) 的詳細資料

1. 登入您的AWS帳戶並打開AWS Private CA作用中目錄主控台的連接器，**https://console.aws.amazon.com/pca-connector-ad/home**。
2. 從中選擇連接器作用中目錄的連接器列出，然後選擇查看詳情。
3. 在連接器詳細資料頁面上，檢閱 [連接器詳細資料] 窗格中的資訊，其中包括下列項目：
   - 接頭識別碼
   - 連接器狀態
   - 其他狀態詳情
   - ARN 連接器
   - 憑證註冊原則伺服器端點
   - 目錄名稱
   - 目錄識別碼
   - AWS 私有 CA主題
   - AWS 私有 CA狀態
   - VPC 端點和安全群組
4. 在模板窗格中，您可以建立或管理與連接器相關聯的範本。
5. 從服務主要名稱 (SPN)」窗格中，您可以檢視與連接器相關聯的服務原則名稱。
6. 從目錄註冊」窗格中，您可以檢視或變更與連接器相關聯的目錄註冊。
7. 從標籤 —可選的」窗格中，您可以建立或管理與連接器相關聯的標籤。

檢視連接器 (CLI)

使用獲取連接器中的指令AWS Private CA的作用中目錄連接器段落AWS CLI。

檢視連接器 (API)

使用GetConnector在「」中的動作AWS Private CA活動目錄 API 的連接器。

檢視連接器範本詳細

使用下列程序，使用主控台、命令列或 API 來檢視連接器範本的組態詳細資料AWS Private CA作用中目錄的連接器

視圖模板 (控制台)

若要檢視連接器範本 (主控台) 的詳細資料

1. 登入您的AWS帳戶並打開AWS Private CA作用中的目錄主控台的連接器**https://console.aws.amazon.com/pca-connector-ad/home**。
2. 選擇連接器作用中目錄的連接器列出，然後選擇查看詳情。
3. 在連接器詳細資料頁面上，檢閱模板區段，然後選取您要檢查的範本。然後選擇查看詳情。
4. 在詳細資訊頁面上，範本詳情窗格會顯示有關範本的下列資訊：

   - 範本名稱
   - 範本識別碼
   - 範本狀態
   - 模板架構版本
   - 模板版本
   - ARN 模板
   - 證書類型
   - 自動註冊已開啟
   - 有效期
   - 更新期
   - 主題名稱要求
   - 主體替代名稱要求
   - 憑證要求和註冊設定
   - 密碼編譯提供者類別
   - 金鑰演算法
   - 金鑰大小下限 (位元)
   - 雜湊演算法
   - 密碼編譯提供者
   - 金鑰使用率延伸設定

   在此窗格中，您也可以使用編輯,刪除，以及動作按鈕。

   - Edit (編輯)

- 刪除
- 管理群組和權限— 如需詳細資訊，請參閱設定群組和權限。
- 管理已取代的範本— 如需詳細資訊，請參閱檢閱和建立。
- 管理標籤— 如需詳細資訊，請參閱標記 AD 資源的連接器。
- 重新註冊所有憑證持有人— 此設定可讓範本的主要版本自動增加。允許使用範本註冊的 Active Directory 群組的所有成員都會收到使用該範本發行的新憑證。如需詳細資訊，請參閱 UpdateTemplate API。

5. 下方窗格會顯示一列標籤，可讓您變更範本的組態。

- 群組和權限— 檢視和管理使用此範本註冊憑證的使用中目錄群組的權限。如需詳細資訊，請參閱設定群組和權限
- 申請政策— 檢視和管理範本應用程式原則。如需詳細資訊，請參閱指派應用程式原。
- 已取代的範本— 檢視和管理已取代的範本。如需詳細資訊，請參閱檢閱和建立。
- 標籤可選的— 查看和管理此模板上的標籤。如需詳細資訊，請參閱標記 AD 資源的連接器。

若要檢視連接器範本的詳細資料 (AWS CLI)

檢視範本

使用獲取範本中的指令AWS Private CA的作用中目錄連接器段落AWS CLI。

檢視範本

若要檢視連接器範本 (API) 的詳細資料

使用 GetTemplate中的動作AWS Private CA活動目錄 API 的連接器。

管理目錄註冊

管理目錄註冊 (主控台)

連接器的目錄註冊可以從的最上層管理AWS Private CA作用中目錄主控台的連接器。本主題逐步介紹可用的管理選項。

1. 登入您的AWS帳戶並打開AWS Private CA作用中目錄主控台的連接器，**https://console.aws.amazon.com/pca-connector-ad/home**。
2. 在左側導覽區域中，選擇目錄註冊。
3. 該目錄註冊頁面會顯示已註冊目錄的表格，其中包含下列欄位：

- 目錄識別碼— 目錄的唯一 ID
- 目錄名稱— 目錄域名網站名稱
- 目錄類型
- 已註冊— 註冊狀態。支持的值是「創建」|「活動」|「刪除」|「失敗
- 目錄狀態— 目錄的狀態

使用可以使用註冊目錄創建一個新的註冊。

4. 您可以選取其中一個列出的註冊以進行管理。這使得查看註冊詳情和取消註冊目錄按鈕。該查看註冊詳情按鈕打開註冊的詳細信息頁面。
5. 該目錄註冊詳情窗格會顯示下列資訊：

- 目錄網域網站名稱
- 目錄識別碼— 目錄的唯一 ID。選擇鏈接將帶您到AWS Directory Service控制台。
- 目錄類型
- 狀態— 目錄的狀態
- 目錄註冊 ARN— 目錄註冊的亞馬遜資源名稱
- 其他狀態資訊

6. 在連接器和服務主要名稱 (SPN)窗格中，您可以管理連接器的 SPN。如需詳細資訊，請參閱〈檢視連接器詳情。
7. 在標籤-可選窗格中，您可以在 AD 資源上應用和刪除元數據。標籤是索引鍵值字串配對，其中索引鍵必須是資源唯一的，而且值是選用的。窗格會在表格中顯示資源的任何現有標籤。支援以下動作。

- 選擇管理標籤打開管理標籤頁面。
- 選擇「新增標籤」以建立標籤。填寫「」鑰匙欄位和 (選擇性)價值欄位。選擇儲存變更以套用標籤。
- 選擇合適的移除標籤旁邊的按鈕將其標記為要刪除，然後選擇儲存變更以確認。

若要管理目錄註冊 (CLI)

創建：使用create-directory-registration中的指令AWS Private CA的作用中目錄連接器段落AWS CLI。

擷取:get-directory-registration中的指令AWS Private CA的作用中目錄連接器段落AWS CLI。

清單:list-directory-registrations中的指令AWS Private CA的作用中目錄連接器段落AWS CLI。

刪除:delete-directory-registration中的指令AWS Private CA的作用中目錄連接器段落AWS CLI。

若要管理目錄註冊 (API)

創建:CreateDirectoryRegistration在「」中的動作AWS Private CA活動目錄 API 的連接器。

擷取:GetDirectoryRegistration在「」中的動作AWS Private CA活動目錄 API 的連接器。

清單:ListDirectoryRegistrations在「」中的動作AWS Private CA活動目錄 API 的連接器。

刪除:DeleteDirectoryRegistration在「」中的動作AWS Private CA活動目錄 API 的連接器。

管理範本的 AD 群組和權限

若要管理範本群組和權限 (主控台)

您可以從範本的詳細資訊頁面管理現有範本的群組和權限。如需詳細資訊，請參閱檢視連接器範本詳細資料

設定群組可以或不能為特定範本註冊憑證的權限。您可以提供群組的安全性識別碼 (SID)。然後，您可以設定群組的註冊和自動註冊權限。對於自動註冊，註冊和自動註冊都必須設定為「允許」。

在活動目錄中查找組安全標識符

您可以使用下面的腳本來查找活動目錄中的組安全標識符。

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

若要管理範本群組和權限 (CLI)

在的 [作用中目錄的AWS Private CA連接器] 區段中建立: create-template-group-access-control- entry 命令。AWS CLI

在的作用中目錄的AWS Private CA連接器區段中更新： update-template-group-access-control- entry 命令。AWS CLI

的 [作用中目錄的AWS Private CA連接器] 區段中的擷取: get-template-group-access-control- entry 命令。AWS CLI

的 [作用中目錄的AWS Private CA連接器] 區段中的清單： list-template-group-access-control- entries 命令。AWS CLI

刪除的 [作用中目錄的AWS Private CA連接器] 區段中的 [delete-template-group-access控制項] 命令。AWS CLI

若要管理範本群組和權限 (API)

在 CreateTemplateGroupAccessControlEntry作用中目錄 API 的AWS Private CA連接器中建立:動作。

更新： UpdateTemplateGroupAccessControlEntry作用中目錄 API AWS Private CA 連接器中的動作。

擷取： GetTemplateGroupAccessControlEntry作用中目錄 API AWS Private CA 連接器中的動作。

清單： ListTemplateGroupAccessControlEntries作用中目錄 API AWS Private CA 連接器中的動作。

刪除： DeleteTemplateGroupAccessControlEntry作用中目錄 API AWS Private CA 連接器中的動作。

設定服務主體名稱

管理服務主體名稱 (主控台)

您可以從連接器的詳細資料頁面管理現有 AD 連接器的服務主體名稱 (SPN)。如需詳細資訊，請參閱管理目錄註冊檢視連接器詳情

管理服務主體名稱 (CLI)

創建: create-service-principal-name」中的指令AWS Private CA的「作用中目錄」段落的連接器AWS CLI。

擷取: get-service-principal-name」中的指令AWS Private CA的「作用中目錄」段落的連接器AWS CLI。

清單: list-service-principal-names」中的指令AWS Private CA的「作用中目錄」段落的連接器AWS CLI。

刪除: delete-service-principal-name」中的指令AWS Private CA的「作用中目錄」段落的連接器AWS CLI。

管理服務主體名稱 (API)

創建: CreateServicePrincipalName中的動作AWS Private CA作用中目錄 API 的連接器。

擷取: GetServicePrincipalName中的動作AWS Private CA作用中目錄 API 的連接器。

清單: ListServicePrincipalNames中的動作AWS Private CA作用中目錄 API 的連接器。

刪除: DeleteServicePrincipalName中的動作AWS Private CA作用中目錄 API 的連接器。

標記 AD 資源的連接器

您可以將標籤套用至連接器、範本和目錄註冊。標記可將中繼資料新增至可協助組織和管理的資源。

若要管理資源標記 (主控台)

現有資源的標記是在資源的詳細資訊頁面上進行管理。如需詳細資訊，請參閱下列程序：

- 檢視連接器範本詳細
- 管理目錄註冊

若要管理資源標記 (CLI)

標籤:標籤資源」中的指令AWS Private CA的「作用中目錄」段落的連接器AWS CLI。

清單標籤: list-tags-for-resource」中的指令AWS Private CA的「作用中目錄」段落的連接器AWS CLI。

取消標記:無標籤資源」中的指令AWS Private CA的「作用中目錄」段落的連接器AWS CLI。

若要管理資源標記 (API)

標籤: TagResource中的動作AWS Private CA作用中目錄 API 的連接器。

清單標籤: ListTagsForResource中的動作AWS Private CA作用中目錄 API 的連接器。

取消標記: UntagResource中的動作AWS Private CA作用中目錄 API 的連接器。

重要提示-使用標籤來標記包含機密數據的對象是可以接受的。但是，標籤本身不應包含任何個人身份信息（PII），敏感或機密信息。

AWS 私有 CA 簡易憑證註冊通訊協定 (SCEP) 的連接器 (預覽版)

SCEP 的連接器正在的預覽版本中， AWS Private CA 且可能會變更。

什麼是適用於 SCEP 的連接器？

簡易憑證註冊通訊協定 (SCEP) 的連接器會連結 AWS Private Certificate Authority 至已啟用 SCEP 的行動裝置和網路設備。使用適用於 SCEP 的連接器，您可以用 AWS Private CA 來發行憑證和註冊 SCEP 裝置。SCEP 的連接器可搭配常用的行動裝置管理 (MDM) 系統使用，其設計可與支援 SCEP 的用戶端或端點搭配使用。

主題

- 用於 SCEP 的連接器的特點
- 如何開始使用適用於 SCEP 的連接器
- 相關服務
- 存取用於 SCEP 的連接器
- 適用於 SCEP 的連接器定價

用於 SCEP 的連接器的特點

Sup@@ port SCEP 通訊協定-SCEP 是一種廣泛採用的通訊協定，可從憑證授權單位 (CA) 取得數位身分憑證，並將其分發至行動裝置和網路設備。您可以使用適用於 SCEP 的連接器來協助您使用 SCEP 註冊端點。

行動裝置註冊-您可以將 SCEP 連接器與熱門 MDM 系統 (包括 Microsoft Intune 和 Jamf Pro) 搭配使用。

大規模發行憑證-將啟用 SCEP 的裝置設定為透過連接器的 SCEP 端點要求憑證後，您的用戶端可以自動從中要求憑證。 AWS Private CA

如何開始使用適用於 SCEP 的連接器

若要開始使用，請從 SCEP 管理主控台的連接器啟動引導式精靈，以協助您建立連接器，並指定要與連接器搭配使用的私有 CA。完成這些步驟後，SCEP 的連接器會提供端點和其他組態參數，供您輸入 MDM 系統或網路設備。設定 MDM 系統或網路設備後，您的用戶端會自動向其要求憑證。AWS Private CA若要深入瞭解如何開始使用適用於 SCEP 的連接器，請參閱開始使用適用於 SC AWS Private Certificate Authority EP 的連接器。

相關服務

用於 SCEP 的連接器與以下 AWS 服務有關。

- AWS Private Certificate Authority- AWS Private CA 為您提供高可用性的私有 CA 服務，而無需支付操作自己的私有 CA 的前期投資和持續維護成本。
- AWS Private CA 作用中目錄的連接器-AD 連接器將您的作用中目錄 (AD) 連結到 AWS Private CA。連接器會代理由 AD 管理的 AWS Private CA 使用者和機器的憑證交換。

存取用於 SCEP 的連接器

您可以使用下列任何介面建立、存取及管理 SCEP 連接器的連接器：

- AWS Management Console-提供可用來存取 SCEP 連接器的 Web 介面。請參閱 SCEP 管理主控台的連接器。
- AWS Command Line Interface-提供一組廣泛 AWS 服務的命令，包括用於 SCEP 的連接器。在視窗、macOS 和 Linux 上支援。 AWS CLI 如需詳細資訊，請參閱 AWS Command Line Interface。
- AWS SDK-提供特定於語言的 API 並處理許多連接詳細信息，例如計算簽名，處理請求重試和錯誤處理。如需詳細資訊，請參閱 AWS Command Line Interface。
- 適用於 SCEP API 的連接器-提供您使用 HTTPS 要求呼叫的低階 API 動作。使用 SCEP API 的連接器是存取服務的最直接方式。不過，SCEP API 的連接器會要求您的應用程式處理低階詳細資料，例如產生雜湊以簽署要求，以及處理錯誤。如需詳細資訊，請參閱 SCEP API 參考的連接器。

適用於 SCEP 的連接器定價

SCEP 連接器的 AWS 私有 CA 功能提供，無需額外費用。您只需支付用於建立和更新連接器的 AWS Private Certificate Authority 作業和憑證費用。

如需最新 AWS 私有 CA 定價資訊，請參閱AWS Private Certificate Authority 定價。您也可以使用定AWS 價計算器來估算成本。

適用於 SCEP 概念的連接器

SCEP 的連接器正在的預覽版本中， AWS Private CA 且可能會變更。

SCEP 連接器是的附加功能。 AWS Private Certificate Authority

以下是 SCEP 連接器的主要概念：

憑證簽署要求

為了獲發數碼證書而向通訊局提供的必要信息。此信息包含一個公鑰以及一個標識。

挑戰密碼

SCEP 通訊協定會使用挑戰密碼來驗證要求，然後再從 CA 發出憑證。SCEP 的連接器會根據連接器類型來處理 SCEP 挑戰密碼。如需詳細資訊，請參閱 SCEP 連接器的運作方式。

證書撤銷

憑證撤銷是指在已發行憑證到期日之前撤銷憑證的程序。您可以在 API、 AWS SDK 或RevokeCertificate中呼叫來撤銷與連接器相關聯的私有 CA 憑證 AWS CloudFormation。 AWS Command Line Interface

連接器應用於 SCEP

用於 SCEP 連結 AWS Private CA 至啟用 SCEP 之裝置的連接器。

行動裝置管理

行動裝置管理 (MDM) 可讓 IT 管理員控制、保護和執行智慧型手機、平板電腦和其他端點或裝置上的政策。許多 MDM 系統為基於 SCEP 的證書註冊提供內置集成。

哭泣

SCEP 是用來自動散發憑證的標準化通訊協定 (RFC 8894)。通訊協定為裝置提供端點，以向 CA 要求憑證。SCEP 使用挑戰密碼授權憑證發行給裝置。SCEP 通常應用於移動設備管理（MDM）系統和網絡設備。MDM 解決方案可讓 IT 管理員控制、保護及執行智慧型手機、平板電腦及其他實體 (例如 Apple 工作站) 的政策。大多數 MDM 解決方案都支持 SCEP，例如 Microsoft Intune，蘋果

MDM 和 Jamf 專業版。大多數網路設備 (例如路由器、負載平衡器、Wi-Fi 中樞、VPN 裝置和防火牆) 都使用 SCEP 進行自動化憑證註冊。

SCEP 設定檔

SCEP 設定檔包含用來定義憑證設定檔的組態參數。這包括憑證有效期間、金鑰大小、SCEP 組態名稱、挑戰密碼、失敗的嘗試重試次數和重試間隔次數，以及其他與發行憑證相關的資訊。MDM 系統和憑證管理平台通常會將 SCEP 設定檔傳送至要求驗證憑證的用戶端。

SCEP 連接器的運作方式

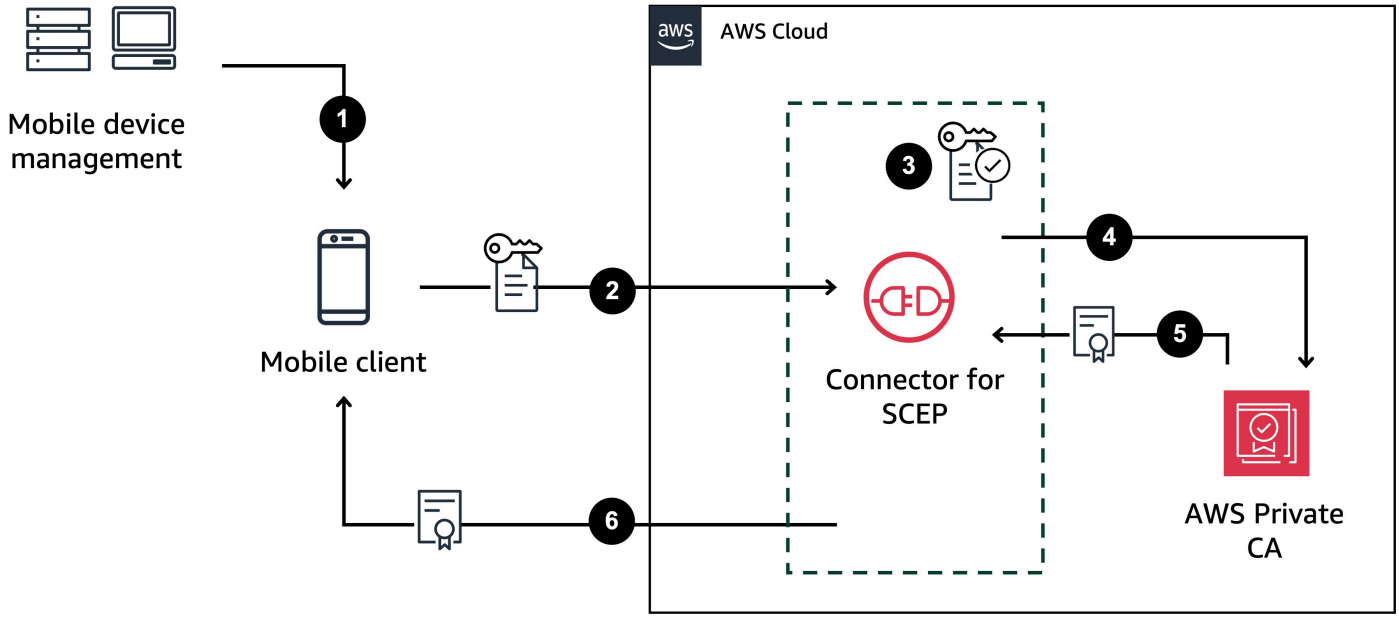
SCEP 的連接器正在的預覽版本中， AWS Private CA 且可能會變更。

簡易憑證註冊通訊協定 (SCEP) 是用於憑證註冊和續訂的標準通訊協定。適用於 SCEP 的連接器是以 RFC 8894 為基礎的 SCEP 伺服器，可自動將憑證從 AWS Private Certificate Authority 您的 SCEP 用戶端發行。當您建立連接器時，SCEP 的連接器會提供 HTTPS 端點，供 SCEP 用戶端要求憑證。用戶端會使用在服務憑證簽署要求 (CSR) 中包含的挑戰密碼來進行驗證。您可以將適用於 SCEP 的連接器與流行的行動裝置管理 (MDM) 解決方案 (包括 Microsoft Intune 和 Jamf Pro) 搭配使用來註冊行動裝置。它適用於任何支援 SCEP 的用戶端或端點。

適用於 SCEP 的連接器提供兩種類型的連接器 — 一般用途和連接器適用於 Microsoft Intune 的 SCEP。以下各節將說明它們的運作方式。

一般用途

一般用途連接器的設計是為了與支援 SCEP 的端點搭配使用 (Microsoft Intune 除外)，我們有一個特殊的連接器。使用一般用途連接器，您可以管理 SCEP 挑戰密碼。下圖以行動裝置管理 (MDM) 系統為例，但如果您使用的是啟用 SCEP 的類似系統或裝置，也會套用相同的功能。

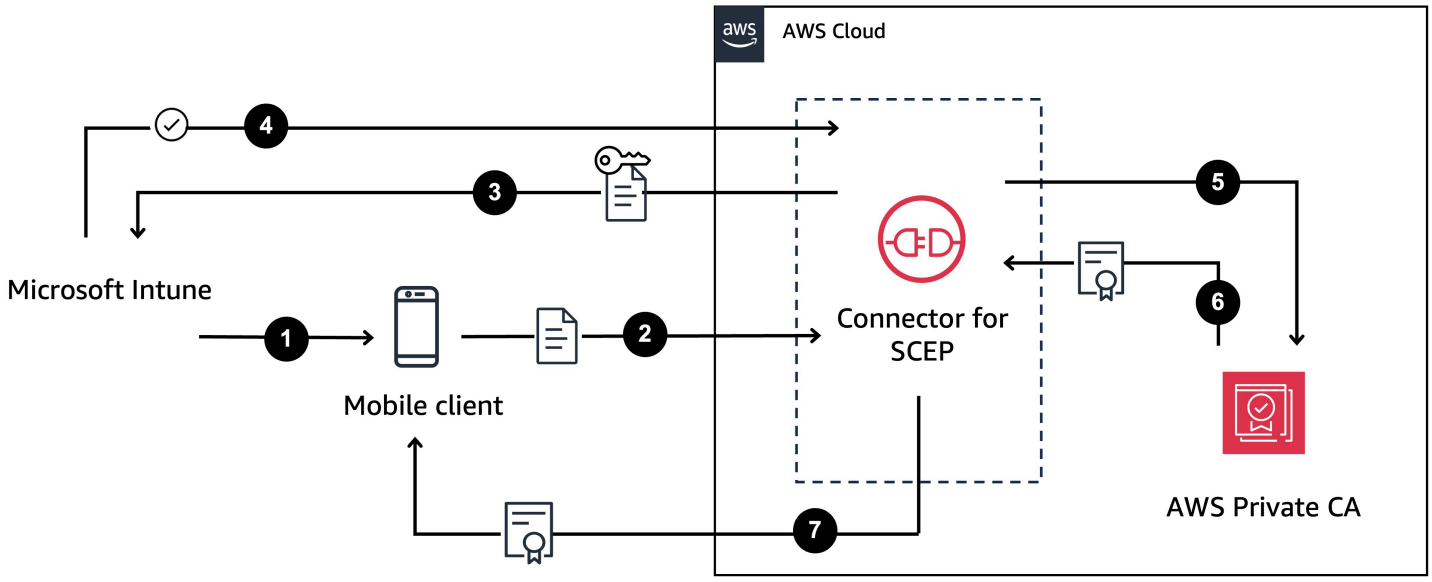


1. MDM 系統 (或類似裝置或系統) 會將 SCEP 設定檔傳送至行動用戶端。SCEP 設定檔包含用來定義憑證設定檔的組態參數，包括憑證有效期間、金鑰大小、SCEP 組態名稱、挑戰密碼、失敗嘗試次數和重試間隔，以及其他與發行憑證相關的資訊。
2. 行動用戶端要求憑證，並傳送包含挑戰密碼的憑證簽署要求 (CSR)。
3. SCEP 的連接器會驗證挑戰密碼。如果它是有效的，則服務會要求代表 AWS Private CA 行動用戶端的憑證。
4. AWS Private CA 發出憑證並將其傳送至 SCEP 的連接器。
5. SCEP 的連接器會將已核發的憑證傳送至行動用戶端。

AWS Private Certificate Authority 適用於 Microsoft Intune 的 SCEP 連接器

AWS Private CA 連接器為 SCEP Microsoft Intune 的設計與 Microsoft Intune 一起使用。使用適用於 Microsoft Intune 連接器類型的 SCEP 連接器，您將使用 Microsoft Intune 來管理您的 SCEP 挑戰密碼。如需將連接器與 Microsoft Intune 搭配使用 SCEP 的相關資訊，請參閱。使用連接器適用於 Microsoft Intune 的 SCEP

當您將連接器用於 SCEP 與 Microsoft Intune 時，某些功能會透過 Microsoft API 存取 Microsoft Intune 來啟用。您使用 SCEP 連接器及隨附的 AWS 服務並不會讓您需要擁有 Microsoft Intune 服務的有效授權。您也應該檢閱 Microsoft Intune® 應用程式保護原則。



1. Microsoft Intune 會將 SCEP 設定檔傳送至行動用戶端。設定檔包含行動用戶端放入 CSR 的加密挑戰密碼。
2. 行動用戶端要求憑證，並將 CSR 傳送至用於 SCEP 的連接器。
3. 用於 SCEP 的連接器會將 CSR 傳送至 Microsoft Intune 以進行授權。
4. Microsoft Intune 解密 CSR 中的挑戰密碼。如果有效，Microsoft Intune 會將核准傳送至 SCEP 的連接器，以便將憑證發行至行動用戶端。
5. SCEP 的連接器代表 AWS Private CA 行動用戶端要求憑證。
6. AWS Private CA 發出憑證並將其傳送至 SCEP 的連接器。
7. SCEP 的連接器會將已核發的憑證傳送至行動用戶端。

使用 SCEP 的連接器時的注意事項和限制

考量事項

CA 操作模式

您只能將連接器用於具有使用一般用途操作模式的私有 CA 的 SCEP。SCEP 的連接器預設為簽發有效期為一年的憑證。使用短期憑證模式的私有 CA 不支援簽發有效期超過七天的憑證。如需有關操作模式的資訊，請參閱憑證機構模式。

挑戰密碼

- 非常小心地分發您的挑戰密碼，並且僅與高度信任的個人和客戶共享。單一挑戰密碼可用來核發任何憑證，包含任何主體和 SAN，這會造成安全風險。
- 如果使用一般用途連接器，建議您經常手動輪換您的挑戰密碼。

對 RFC 8894 的一致性

SCEP 的連接器透過提供 HTTPS 端點而非 HTTP 端點，從而不是 RFC 8894 通訊協定偏離。

社会责任

- 如果傳送至 SCEP 連接器的憑證簽署要求 (CSR) 不包含延伸金鑰使用量 (EKU) 延伸模組，我們會將 EKU 值設定為。`clientAuthentication`如需詳細資訊，請參閱 4.2.1.12。RFC 5280 中的延伸金鑰使用方式。
- 我們在 CSR 中支持`ValidityPeriod`和`ValidityPeriodUnits`自定義屬性。如果您的 CSR 不包含`ValidityPeriod`，我們會頒發一份有效期為一年的憑證。請記住，您可能無法在 MDM 系統中設定這些屬性。但是，如果您可以設置它們，我們將支持它們。如需有關這些屬性的資訊，請參閱 < 名稱 _ 值對 > 。

端點共享

僅將連接器的端點散發給受信任的方。將端點視為秘密，因為任何能夠找到您唯一的完整網域名稱和路徑的人都可以擷取您的 CA 憑證。

限制

下列限制適用於 SCEP 的連接器。

動態挑戰密碼

您只能使用一般用途連接器建立靜態挑戰密碼。若要搭配一般用途連接器使用動態密碼，您必須建置自己的循環機制，以使用連接器的靜態密碼。適用於 Microsoft Intune 連接器類型的 SCEP 連接器提供動態密碼的支援，您可以使用 Microsoft Intune 來管理。

HTTP

適用於 SCEP 的連接器僅支援 HTTPS，並建立 HTTP 呼叫的重新導向。如果您的系統依賴 HTTP，請確定它可以容納 SCEP 連接器所提供的 HTTP 重新導向。

共用私有 CA

您只能將連接器用於具有您身為擁有者的私人 CA 的 SCEP。

設定用於 SCEP 的連接器

SCEP 的連接器正在的預覽版本中， AWS Private CA 且可能會變更。

本節中的程序可協助您開始使用 SCEP 的連接器。它假設您已經創建了一個 AWS 帳戶。完成此頁面中的步驟後，您可以繼續為 SCEP 建立連接器。

主題

- 步驟 1：建立 AWS Identity and Access Management 策略
- 步驟 2：建立私有 CA
- 步驟 3：使用建立資源共用 AWS Resource Access Manager

步驟 1：建立 AWS Identity and Access Management 策略

若要為 SCEP 建立連接器，您需要建立 IAM 政策，讓 SCEP 的 Connector 能夠建立和管理連接器所需的資源，以及代表您發行憑證。如需 IAM 的詳細資訊，請參閱什麼是 IAM？ 在 IAM 使用者指南中。

下列範例是可用於 SCEP 連接器的客戶管理原則。

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

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "acm-pca:IssueCertificate",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/
BlankEndEntityCertificate_APICSRPassthrough/V*"
                },
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "pca-connector-scep.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:CreateResourceShare",
                "ram:GetResourcePolicies",
                "ram:GetResourceShareAssociations",
                "ram:GetResourceShares",
                "ram:ListPrincipals",
                "ram:ListResources",
                "ram:ListResourceSharePermissions",
                "ram:ListResourceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

步驟 2：建立私有 CA

若要使用 SCEP 的連接器，您需要將私人 CA 從 AWS Private Certificate Authority 與連接器相關聯。由於 SCEP 通訊協定中存在固有的安全性弱點，因此建議您使用僅適用於連接器的私有 CA。

私有 CA 必須符合下列要求：

- 它必須處於作用中狀態，並使用一般用途的操作模式。
- 您必須擁有私有 CA。您無法使用透過跨帳戶共用與您共用的私有 CA。

將私有 CA 設定為搭配 SCEP 的連接器使用時，請注意下列考量事項：

- DNS 名稱限制 — 請考慮使用 DNS 名稱限制來控制針對 SCEP 裝置發行的憑證中允許或禁止哪些網域。如需詳細資訊，請參閱如何在 AWS Private Certificate Authority.
- 撤銷 — 在您的私有 CA 上啟用 OCSP 或 CRL 以允許撤銷。如需詳細資訊，請參閱 設定憑證撤銷方法。
- PII — 我們建議您不要在 CA 憑證中新增個人識別資訊 (PII) 或其他機密或敏感資訊。如果發生安全漏洞，這有助於限制敏感信息的暴露。
- 將根憑證儲存在信任存放區 — 將根 CA 憑證儲存在裝置信任存放區中，以便您可以驗證憑證和的傳回值GetCertificateAuthorityCertificate。如需信任存放區相關的資訊 AWS Private CA，請參閱根 CA 。

如需如何建立私有 CA 的詳細資訊，請參閱建立私有 CA。

步驟 3：使用建立資源共用 AWS Resource Access Manager

如果您以程式設計方式使用 SCEP 的連接器 AWS Command Line Interface，以 AWS 程式設計方式使用 SCEP 的連接器，則需要使用 AWS Resource Access Manager 服務主體共用與 SCEP 的連接器共用私有 CA。這可讓 SCEP 的連接器共用存取您的私有 CA。當您在 AWS 主控台中建立連接器時，我們會自動為您建立資源共用。如需有關資源共用的資訊，請參閱AWS RAM 使用指南中的建立資源共用。

若要使用建立資源共用 AWS CLI，您可以使用 AWS RAM create-resource-share指令。下面的命令創建一個資源共享。指定您要共用的私有 CA 的 ARN 作為資# arns 的值。

```
$  aws ram create-resource-share \
--region us-east-1 \
--name MyPcaConnectorScepResourceShare \
--permission-arns arn:aws:ram::aws:permission/
AWSRAMBlankEndEntityCertificateAPICSRPassthroughIssuanceCertificateAuthority \
--resource-arns arn:aws:acm-pca:Region:account:certificate-authority/CA_ID \
--principals pca-connector-scep.amazonaws.com \
--sources account
```

呼叫的服務主體在私CreateConnector有 CA 上具有憑證發行權限。若要防止使用 SCEP Connector 的服務主體具有 AWS 私有 CA 資源的一般存取權，請使用來限制其權限。CalledVia

開始使用適用於 SC AWS Private Certificate Authority EP 的連接器

SCEP 的連接器正在的預覽版本中， AWS Private CA 且可能會變更。

使用 SCEP AWS Private Certificate Authority 連接器，您可以將憑證從私有 CA 發行至啟用 SCEP 的裝置和行動裝置管理 (MDM) 系統。當您建立連接器時， AWS Private Certificate Authority 會建立公用 SCEP URL 供您要求憑證，並提供您可用來整合至 MDM 系統的資訊。

若要發行憑證，您必須建立 AWS Private Certificate Authority 私有 CA、建立連接器，然後設定啟用 SCEP 的 MDM 系統和裝置，以向連接器要求憑證。

主題

- 開始之前
- 步驟 1：建立連接器
- 步驟 2：將連接器詳細資料複製到 MDM 系統

開始之前

接下來的教學課程將引導您完成為 SCEP 建立連接器的過程。

要按照本教程進行操作，您將需要一個私有 CA 和啟用了 SCEP 的設備。您也必須先符合設定用於 SCEP 的連接器區段中列出的先決條件。

下列程序會引導您如何使用 AWS 控制台建立連接器。

任務

- 步驟 1：建立連接器
- 步驟 2：將連接器詳細資料複製到 MDM 系統

步驟 1：建立連接器

您將建立一個連接器以供一般用途使用，或為 Microsoft Intune 的 SCEP 建立連接器。一般用途連接器是專為與具有 SCEP 功能的端點搭配使用而設計，而且您可以管理 SCEP 挑戰密碼。Microsoft Intune 的 SCEP 連接器是與 Microsoft Intune 一起使用，並且您管理挑戰密碼使用 Microsoft Intune。

General-purpose

若要建立一般用途的連接器

登入您的 AWS 帳戶，然後在以下位**https://console.aws.amazon.com/pca-connector-scep/home** 置開啟 SCEP 主控台的連接器。

1. 選擇 Create connector (建立連接器)。
2. 在 [建立連接器] 頁面中，選擇性地在 [名稱標籤] 欄位中為連接器指定好記名稱。名稱將顯示在連接器清單中。如果需要，您可以選取 [新增更多標籤]，將更多標籤新增至連接器。標籤是指派給 AWS 資源的標籤。每個標籤皆包含索引鍵與選用值。您可以使用標籤來搜尋和篩選資源或追蹤 AWS 成本。
3. 在「連接器類型」下，選擇「一般用途」。
4. 在「私人 CA」下，選擇要與此連接器搭配使用的私有 CA。或者，透過選取建立私有 CA 來建立新的 CA。由於 SCEP 通訊協定中的固有弱點，我們建議您使用專用於此連接器的私有 CA。如果您已建立新 CA，則在中完成建立後 AWS Private CA，請返回 SCEP 主控台的連接器，並重新整理私有 CA 清單。您的新私有 CA 應該可供選擇。
5. 在 [問題密碼] 下方，選取 [自動產生挑戰 當我們建立此連接器時，我們會為您產生靜態挑戰密碼。
6. 選取建立連接器。

Microsoft Intune

若要建立適用於 Microsoft Intune 的 SCEP 連接器

登入您的 AWS 帳戶，然後在以下位**https://console.aws.amazon.com/pca-connector-scep/home** 置開啟 SCEP 主控台的連接器。

1. 選擇 Create connector (建立連接器)。
2. 在 [建立連接器] 頁面上，選擇性地在 [名稱標籤] 欄位中為連接器指定好記名稱。名稱將顯示在連接器清單中。如果需要，您可以選取 [新增更多標籤]，將更多標籤新增至連接器。標籤是指派給 AWS 資源的標籤。每個標籤皆包含索引鍵與選用值。您可以使用標籤來搜尋和篩選資源或追蹤 AWS 成本。
3. 在 [連接器類型] 下，選擇 [Microsoft Intune]。

a. 針對應用程式 (用戶端) 識別碼，請輸入 Microsoft Entra ID 應用程式註冊中的應用程式 (用戶端) 識別碼。如需有關針對 SCEP 使用 Microsoft Intune 與連接器的資訊，請參閱。將連接器用於搭配 MDM 系統的 SCEP

b. 針對目錄 (租用戶) 識別碼或主要網域，請輸入 Microsoft Entra ID 應用程式註冊中的目錄 (租用戶) 識別碼或主要網域。

4. 在「私人 CA」下，選擇要與此連接器搭配使用的私有 CA。或者，透過選取建立私有 CA 來建立新的 CA。由於 SCEP 通訊協定中的固有弱點，我們建議您使用專用於此連接器的私有 CA。如果您已建立新 CA，則在中完成建立後 AWS Private CA，請返回 SCEP 主控台的連接器，並重新整理私有 CA 清單。您的新私有 CA 應該可供選擇。

5. 選取建立連接器。

步驟 2：將連接器詳細資料複製到 MDM 系統

建立連接器後，您需要將下列詳細資料從連接器複製到 MDM 系統中。若要使用主控台檢視連接器的詳細資料，請從 SCEP 主控台的連接器頁面上的清單中選取連接器。

- 公用 SCEP URL-這是您的 SCEP 用戶端要求憑證的連接器端點。請注意僅將此端點提供給受信任的實體。
- (一般用途) 挑戰密碼-在 [挑戰密碼] 下，選取您在上述程序中自動產生的密碼，然後選取 [檢視密碼] 以檢視密碼。若要建立其他密碼，請選取 [建立密碼]。小心謹慎地將密碼分發給高度信任的個人和客戶。單一挑戰密碼可用來核發任何憑證，包含任何主旨和 SAN，因此應小心處理。
- (Microsoft Intune) 開放識別碼值-如果您要與 Microsoft Intune 整合，您必須將開放識別碼發行者、開放識別碼主旨和開放識別碼受眾複製到您的 Microsoft 應用程式註冊的 OpenID Connect (OIDC) 認證中。如需更多詳細資訊，請參閱 將連接器用於搭配 MDM 系統的 SCEP。

將連接器用於搭配 MDM 系統的 SCEP

SCEP 的連接器正在的預覽版本中， AWS Private Certificate Authority 且可能會變更。

下列各節說明如何將連接器用於 SCEP 與特定行動裝置管理 (MDM) 系統搭配使用。

主題

- 使用連接器用於 SCEP 與 Jamf 專業版

- 使用連接器適用於 Microsoft Intune 的 SCEP

使用連接器用於 SCEP 與 Jamf 專業版

您可 AWS Private CA 以將 Jamf Pro 行動裝置管理 (MDM) 解決方案用作外部憑證授權單位 (CA)。本指南提供有關如何在創建 SCEP AWS Private Certificate Authority 連接器後將 Jamf Pro 配置為 SCEP 代理的說明。

賈姆夫專業版要求

您的實施 Jamf Pro 必須滿足以下要求。

- 您必須使用賈姆夫臨 10.0.0 或更高版本。
- 您必須在 Jamf Pro 中啟用啟用憑證型驗證設定。您可以在 Jamf Pro 文檔中的 Jamf Pro 安全設置頁面上找到有關此設置的詳細信息。

必要條件

若要將連接器用於 SCEP 與 Jamf Pro 搭配使用，您必須先為 SCEP 建立私有 CA 和一般用途連接器。如需說明，請參閱設定用於 SCEP 的連接器。

在 Jamf Pro 中配置 AWS Private CA 為外部 CA

建立 SCEP 的連接器之後，您必須在 Jamf Pro 中設定 AWS Private CA 為外部 CA。您可以設定 AWS Private CA 為全域外部 CA。或者，您可以使用 Jamf Pro 設定描述檔， AWS Private CA 針對不同的使用案例發行不同的憑證，例如向組織中的裝置子集頒發憑證。有關實施 Jamf Pro 配置文件的指導超出了本文檔的範圍。

若要在 Jamf Pro 中設定 AWS Private CA 為外部 CA

1. 在 Jamf Pro 主控台中，前往 「設定」>「全域」>「PKI 憑證」，進入 PK I 憑證設定頁面。
2. 選取管理憑證範本索引標籤。
3. 選取 [外部 CA]。
4. 選擇 Edit (編輯)。
5. (選擇性) 針對組態設定檔選取啟用 Jamf Pro 作為 SCEP 代理伺服器。您可以使用 Jamf Pro 設定描述檔來發行針對特定使用案例量身打造的不同憑證。有關如何在 Jamf Pro 中使用配置文件的指導，請參閱 J amf Pro 文檔中的配置文件中啟用 Jamf Pro 作為 SCEP 代理配置文件。

6. 選取 [使用具有 SCEP 功能的外部 CA 進行電腦和行動裝置註冊]。
7. (選擇性) 選取使用 Jamf Pro 作為 SCEP 代理伺服器以進行電腦和行動裝置註冊。如果您遇到設定檔安裝失敗，請參閱排解設定檔安裝失敗。
8. 將連接器的詳細資料中的 SCEP 公用 SCEP URL 的連接器複製並貼到 Jamf Pro 中的 URL 欄位。若要檢視連接器的詳細資料，請從 SCEP 的連接器清單中選擇連接器。或者，您可以通過調用GetConnector並從響應中復制`Endpoint`值來獲取 URL。
9. (選擇性) 在「名稱」欄位中輸入執行處理的名稱。例如，您可以將其命名AWS Private CA。
10. 選取 [靜態] 做為挑戰類型。
11. 複製連接器的挑戰密碼，並將其貼到「挑戰」欄位中。若要檢視連接器的挑戰密碼，請導覽至 AWS 主控台中連接器的詳細資料頁面，然後選取 [檢視密碼] 按鈕。或者，您也可以呼叫 Pass GetChallengeword 並複製回應中的`Password`值，以取得連接器的挑戰密碼。
12. 將挑戰密碼貼到「驗證挑戰」欄位中。
13. 選擇金鑰大小。我們建議使用 2048 或更高的金鑰大小。
14. (選擇性) 選取「用作數位簽章」。選取此選項以進行驗證，以授予裝置安全存取 Wi-Fi 和 VPN 等資源的權限。
15. (選擇性) 選取用於金鑰加密。
16. (選擇性) 在指紋欄位中輸入十六進位字串。如需如何建立私人 CA 指紋的指示，請參閱(選擇性) 新增 CA 指紋。
17. 選取 Save (儲存)。

建立並上傳設定檔簽署憑證

若要在 Jamf Pro 中使用適用於 SCEP 的連接器，您必須為與連接器相關聯的私有 CA 提供簽署和 CA 憑證。您可以通過將配置文件簽名證書密鑰庫上傳到包含兩個證書的 Jamf Pro 來執行此操作。您需要使用內部程序產生憑證簽署要求 (CSR)，並由其簽署 AWS Private Certificate Authority。以下說明將說明如何創建證書密鑰庫並將其上傳到 Jamf Pro。下列範例使用 OpenSSL，但您可以使用偏好的方法產生憑證簽署要求。

1. 使用 OpenSSL，執行下列命令來產生私密金鑰：

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

2. 產生憑證簽署要求 (CSR)：

```
openssl req -new -key local.key -sha512 -out local.csr -
subj "/CN=MySigningCertificate/O=MyOrganization" -addext
 keyUsage=critical,digitalSignature,nonRepudiation
```

3. 使用 AWS CLI，使用您在步驟二中產生的 CSR 發行簽署憑證。執行下列命令，並在回應中記下憑證 ARN：

```
aws acm-pca issue-certificate --certificate-authority-arn <SAME CA AS USED ABOVE,
 SO IT'S TRUSTED> --csr fileb://local.csr --signing-algorithm SHA512WITHRSA --
validity Value=365,Type=DAYS
```

4. 使用步驟 3 中的憑證 ARN 執行下列命令，以取得簽署憑證：

```
aws acm-pca get-certificate --certificate-authority-arn <SAME CA AS USED ABOVE, SO
 IT'S TRUSTED> --certificate-arn <ARN OF NEW CERTIFICATE> | jq -r '.Certificate'
 >local.crt
```

5. 執行下列命令以取得 CA 憑證：

```
aws acm-pca get-certificate-authority-certificate --certificate-authority-arn <SAME
 CA AS USED ABOVE, SO IT'S TRUSTED> | jq -r '.Certificate' > ca.crt
```

6. 使用 OpenSSL，以 p12 格式輸出簽署憑證金鑰儲存庫。您將使用在步驟四和第五步中產生的`crt`檔案。執行以下命令：

```
openssl pkcs12 -export -in local.crt -inkey local.key -certfile ca.crt -name "CA
 Chain" -out local.p12
```

7. 出現提示時，請輸入匯出密碼。此密碼是您的密鑰庫密碼，您稍後需要使用它。
8. 在 Jamf Pro 中，導航到管理證書模板，然後轉到外部 CA 窗格。
9. 在「外部 CA」窗格底部，選取「變更簽署和 CA 憑證」。
10. 遵循畫面上的指示，上傳外部 CA 的簽署憑證和 CA 憑證。

(選擇性) 新增 CA 指紋

新增 CA 指紋可讓受管理裝置驗證 CA，並僅向 CA 要求憑證。

1. 從 AWS Private CA 主控台或使用取得私有 CA 憑證GetCertificateAuthorityCertificate。將其另存為`ca.pem`文件。

2. 在 OpenSSL 中，執行下列命令：

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

3. 將輸出複製並貼到前述程序中所述的「指紋」欄位中。

(選用) 在使用者啟動的註冊期間安裝憑證

若要在使用者啟動的註冊期間將連接器的私人 CA 憑證安裝到用戶端或裝置，請設定 Jamf Pro 使用者啟動的註冊設定。這有助於 Jamf Pro 在客戶端或設備請求 AWS Private CA 證書時將證書安裝到客戶端或設備上。您有責任測試您的組態，以確保其與用於 SCEP 實作的連接器相容。如需 Jamf Pro 使用者啟動的註冊設定的相關資訊，請參閱 Jamf Pro 文件中的使用者啟動註冊設定。

排解設定檔安裝失敗

如果您在啟用電腦和行動裝置註冊時使用 Jamf Pro 做為 SCEP Proxy 後遇到設定檔安裝失敗，請嘗試下列動作。

錯誤訊息	緩解
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:15001>`	如果您在嘗試註冊時收到此錯誤訊息，請重試註冊。註冊成功之前，可能需要多次嘗試。
`Profile installation failed. Unable to obtain certificate from SCEP server at "<your-jamf-endpoint>.jamfcloud.com". <MDM-SCEP:14006>`	您的挑戰密碼可能設定錯誤。請確認 Jamf Pro 中的挑戰密碼與連接器的挑戰密碼相符。

使用連接器適用於 Microsoft Intune 的 SCEP

您可 AWS Private CA 以將外部憑證授權單位 (CA) 與 Microsoft Intune 行動裝置管理 (MDM) 系統搭配使用。本指南提供有關如何在建立適用於 Microsoft Intune 的 SCEP 連接器之後設定 Microsoft Intune 的指示。

必要條件

建立適用於 Microsoft Intune 的 SCEP 連接器之前，您必須完成下列必要條件。

- 創建一個項目 ID。
- 建立 Microsoft 的 Intune 租用戶。
- 在您的 Microsoft 項目 ID 中創建一個應用程序註冊。如需有關如何管理應用程式層級權限的詳細資訊，請參閱 Microsoft Entra ID 中的更新應用程式所要求的權限。應用程序註冊必須具有以下權限：
  - 在 [Intune] 下設定偵測挑戰提供者。
  - 對於 Microsoft 圖形設置應用程序. 閱讀全部和用戶.
- 您必須在應用程式註冊管理員同意中授予應用程式。如需詳細資訊，請參閱 Microsoft Entra 文件中的授予承租人範圍的管理員同意給應用程式。

> **Tip**
>
> 建立應用程式註冊時，請記下應用程式 (用戶端) ID 和目錄 (租用戶) 識別碼或主要網域。當您建立適用於 Microsoft Intune 的 SCEP 連接器時，您將輸入這些值。如需有關如何取得這些值的詳細資訊，請參閱建立可存取資源的 Microsoft Entra 應用程式和服務主體 (位於 Microsoft Entra 文件中)。

授 AWS Private CA 予使用 Microsoft 項目 ID 應用程式的權限

建立適用於 Microsoft Intune 的 SCEP 連接器之後，您必須在 Microsoft 應用程式登錄下建立同盟認證，以便 SCEP 的連接器可以與 Microsoft Intune 通訊。

若要在 Microsoft Intune 中設定 AWS Private CA 為外部 CA

1. 在 Microsoft Entra ID 控制台中，導航到應用程序註冊。
2. 選擇您建立要與 SCEP 連接器搭配使用的應用程式。您按一下之應用程式的應用程式 (用戶端) ID 必須符合您在建立連接器時指定的識別碼。
3. 從受管理的下拉式功能表中選取憑證和密碼。
4. 選取 [同盟認證] 索引標籤。
5. 選取 [新增認證]。
6. 從同盟認證案例下拉式功能表中，選擇其他發行者。

7. 將 OpenID 發行者值從適用於 Microsoft Intune 詳細資料的 SCEP 連接器複製並貼到 [發行者] 欄位中。若要檢視連接器的詳細資料，請從 AWS 主控台的 SCEP 連接器清單中選擇連接器。或者，您可以通過調用獲取 URL，GetConnector然後從響應中復制`Issuer`值。
8. 將 OpenID 對象值從適用於 Microsoft Intune 詳細資料的 SCEP 連接器複製並貼到「對象」欄位中。若要檢視連接器的詳細資料，請從 AWS 主控台的 SCEP 連接器清單中選擇連接器。或者，您可以通過調用獲取 URL，GetConnector然後從響應中復制`Subject`值。
9. (選擇性) 在「名稱」欄位中輸入執行處理的名稱。例如，您可以將其命名AWS Private CA。
10. (選擇性) 在「說明」欄位中輸入說明。
11. 選取「對象」欄位下的「編輯」(選用)。將 OpenID 主題值從連接器複製並粘貼到「主題」字段中。您可以在 AWS 主控台的連接器詳細資料頁面中檢視 OpenID 簽發者值。或者，您可以通過調用獲取 URL，GetConnector然後從響應中復制`Audience`值。
12. 選取新增。

設定 Microsoft 的 Intune 設定設定檔

在您授 AWS Private CA 與呼叫 Microsoft Intune 的權限之後，您必須使用 Microsoft Intune 建立 Microsoft Intune 組態設定設定檔，以指示裝置連接至用於 SCEP 的連接器以進行憑證發行。

1. 建立信任的憑證組態設定檔。您必須將與 SCEP 連接器搭配使用的鏈結的根 CA 憑證上傳至 Microsoft Intune，才能建立信任。如需如何建立受信任憑證組態設定檔的詳細資訊，請參閱 Microsoft Intune 說明文件中的受信任的根憑證設定檔。
2. 建立 SCEP 憑證組態設定設定檔，以便在裝置需要新憑證時將裝置指向連接器。組態設定檔的設定檔類型應該是 SCEP 憑證。對於組態設定檔的根憑證，請確定您使用在上一個步驟中建立的信任憑證。

   對於 SCEP 伺服器 URL，請將公用 SCEP URL 從連接器的詳細資料複製並貼到「SCEP 伺服器 URL」欄位中。若要檢視連接器的詳細資料，請從 SCEP 的連接器清單中選擇連接器。或者，您可以通過調用獲取 URL GetConnector，然後從響應中復制`Endpoint`值。如需在 Microsoft Intune 中建立組態設定檔的指引，請參閱 Microsoft Intune 說明文件中的建立和指派 SCEP 憑證設定檔。

   > **Note**
   >
   > 對於非 Mac OS 和 iOS 裝置，如果您未在設定描述檔中設定有效期，SCEP 的連接器會核發有效期為一年的憑證。如果您未在組態設定檔中設定延伸金鑰使用量 (EKU) 值，SCEP 的連接器會發出 EKU 設定為的憑證。`Client Authentication (Object Identifier: 1.3.6.1.5.5.7.3.2)`對於 macOS 或 iOS 裝置，Microsoft Intune 不會

尊重您的設定描述檔`ExtendedKeyUsage`或`Validity`參數。對於這些裝置，用於 SCEP 的連接器會透過用戶端驗證向這些裝置發行一年有效期的憑證。

驗證與 SCEP 連接器的連線

建立指向 SCEP 端點連接器的 Microsoft Intune 組態設定檔之後，請確認註冊的裝置可以要求憑證。若要確認，請確定沒有任何原則指派失敗。若要確認，請在 Intune 入口網站中瀏覽至 [裝置] > [管理裝置] > [組態設定] > [組態原則指派失敗] 下方沒有列出任何內容。如果有，請使用上述程序的資訊確認您的設定。如果您的設定正確且仍有失敗，請參閱從行動裝置收集可用資料。

如需裝置註冊的相關資訊，請參閱什麼是裝置註冊？ 在 Microsoft 的 Intune 文檔中。

故障診斷

如果您在使用 AWS 私有 CA時遇到問題，請參閱下列主題。

主題

- 建立和簽署私有 CA 憑證
- OCSP 回應中的延遲
- 設定 Amazon S3 以允許建立 CRL 儲存貯體
- 撤銷自我簽署的 CA 憑證
- 處理例外狀況
- 使用物質標準
- AD 錯誤和失敗的連接器
- AD 連接器建立失敗錯誤的連接器

建立和簽署私有 CA 憑證

建立私有 CA 後，您必須擷取 CSR 並提交到 X.509 基礎設施中的中繼 CA 或根 CA。您的 CA 使用 CSR 來建立私有 CA 憑證，然後簽署憑證，再傳回給您。

遺憾的是，目前無法針對建立及簽署私有 CA 憑證的相關問題提供特定建議。X.509 基礎結構及其中 CA 階層的詳細資料已超出本 AWS 私有 CA 文件的範圍。

OCSP 回應中的延遲

如果呼叫者在地理位置上遠離區域邊緣快取或發行 CA 的區域，OCSP 回應速度可能會較慢。如需區域邊緣快取可用性的詳細資訊，請參閱全球邊緣網路。我們建議您在使用憑證附近的區域核發憑證。

設定 Amazon S3 以允許建立 CRL 儲存貯體

如果您的帳戶強制執行 Amazon S3 區塊公開存取 (儲存貯體設定)，您的私有 CA 可能無法建立 CRL 儲存貯體。如果發生這種情況，請檢查您的 Amazon S3 設定。如需詳細資訊，請參閱使用 Amazon S3 封鎖公開存取。

撤銷自我簽署的 CA 憑證

您無法撤銷自我簽署的 CA 憑證。反之，您必須刪除 CA。

處理例外狀況

AWS 私有 CA 命令失敗可能有幾個原因。如需每個例外狀況及解決這些例外狀況的建議相關資訊，請參閱下表。

AWS 私有 CA 例外

傳回的例外狀況 AWS 私有 CA	描述	修補
AccessDeniedException	使用指定命令所需的許可，尚未由私有 CA 委派給呼叫帳戶。	如需中委派權限的資訊 AWS 私有 CA，請參閱將憑證續訂權限指派給 ACM。
InvalidArgsException	使用了無效參數來發出建立或更新憑證的請求。	請檢查命令的個別文件，以確認您的輸入參數有效。如果您正在建立新憑證，請確認請求的簽署演算法可與 CA 的金鑰類型搭配使用。
InvalidStateException	因為相關聯的私有 CA 未處於 ACTIVE 狀態，所以無法更新憑證。	嘗試還原私有 CA。如果私有 CA 不在其還原期間內，則無法還原 CA，也無法更新憑證。
LimitExceededException	每個憑證授權機構 (CA) 都有可發行的憑證配額。與指定憑證相關聯的私有 CA 已達到其配額上限。如需詳細資訊，請參閱 AWS 一般參考 指南中的 Service Quotas。	請與中AWS Support 心聯絡以申請提高配額。
MalformedCSRException	AWS 私有 CA 無法確認或驗證提交的憑證簽署請求 (CSR)。	請確認您的 CSR 已正確產生及設定。

傳回的例外狀況 AWS 私有 CA	描述	修補
`OtherException`	發生內部錯誤而導致請求失敗。	請嘗試重新執行命令。如果問題仍然存在，請聯繫中AWS Support 心。
`RequestFailedException`	您 AWS 環境中的網路問題造成要求失敗。	重試 請求。如果故障仍然存在，請檢查您的 Amazon VPC (VPC) 組態。
`ResourceNotFoundException`	發出憑證的私有 CA 已刪除，且已不存在。	向另一個有效的 CA 請求新憑證。
`ThrottlingException`	請求的 API 動作因超過配額而失敗。	確認您發出的通話數量不超過允許的數量 AWS 私有 CA。 您遇到暫時性狀況 (而非超過配額) 時，也可能導致發生 `ThrottlingException` 錯誤。如果您發生該錯誤，且發出的呼叫數未超過配額，請重試請求。 如果您即將達到配額上限，您可能可以請求提高配額。如需詳細資訊，請參閱 AWS 一般參考 指南中的 Service Quotas。
`ValidationException`	請求的輸入參數格式不正確，或根憑證有效期的結束時間早於請求的憑證有效期結束時間。	請檢查命令的輸入參數語法要求，以及 CA 根憑證的有效期。如需如何變更有效期的資訊，請參閱更新您的私有 CA。

使用物質標準

Mat ter 連線標準指定了可改善物聯網 (IoT) 裝置安全性和一致性的憑證組態。您可以在以下位置找到建立符合 Matter 標準的根 CA、中繼 CA 和最終實體憑證的 Java 範例。使用 AWS 私有 CA API 來實現問題標準（Java 示例）

為了協助進行故障排除，Matter 開發人員提供了一個名為晶片證書的證書驗證工具。下表列出了工具報告的錯誤，其中包含修正。

錯誤代碼	意義	修補
0x000003 5	`BasicCons traints KeyUsage`、和`ExtensionKeyUsage` 副檔名必須標示為重要。	請確定您已針對您的使用案例選取正確的範本。
0x000000 0	授權金鑰識別碼副檔名必須存在。	AWS 私有 CA 不會在根憑證上設定授權金鑰識別碼延伸。您必須用 CSR 產生 Base64 編碼的AuthorityKeyIdentifier 值，然後透過CustomExtension 如需詳細資訊，請參閱 為節點操作憑證 (NOC CA。 及 啟動產品驗證授權單位 (PAA)。
0x000000 E	憑證已過期。	請確定您使用的憑證尚未過期。
0x000000 4	憑證鏈結驗證失敗。	如果您嘗試建立符合 Matter 標準的最終實體憑證，而不使用提供範例 (使用 AWS 私有 CA API 傳遞正確設定)，則可能會遇到此錯KeyUsage 依預設， AWS 私有 CA 會產生九位元 KeyUsage 延伸值，第九產生額外的位元組。在格式轉換期間，Metter 忽略額外的字節，證失敗。但是，`APIPassthrough` 模板CustomExtension中的設置`KeyUsage`值中的確切字節數。如需範例，請參閱建立節點(NOC)。 如果您修改範例程式碼或使用替代 X.509 公用程式 (例如 OpenS需要執行手動驗證，以避免鏈結驗證錯誤。

錯誤代碼	意義	修補

		驗證轉換是否為無損失
		
		1. 使用 openssl 驗證節點 (終端實體) 憑證是否包含有效鏈結。例子中，rcac.pem是根 CA 證書，icac.pem是中繼 CA 證且noc.pem是節點證書。
		
		`openssl verify -verbose -CAfile <(cat rcac.pem icac noc.pem`
		
		2. 使用晶片證書將 PEM 格式的節點證書轉換為 TLV（標籤，長值）格式，然後再次返回。
		
		`./chip-cert convert-cert noc.pem noc.chip -c`
		`./chip-cert convert-cert noc.chip noc_converted.pem`
		
		這些文件noc.pem和noc_converted.pem 應該與字符串確認的完全相同。

AD 錯誤和失敗的連接器

使用此處的資訊可協助您在使用 AD 的連接器時診斷及修正錯誤及建立失敗。

主題

- 錯誤
- 連接器建立失敗
- SPN 建立失敗

錯誤

AD 的連接器會傳送錯誤訊息的原因有幾個。如需每個錯誤的相關資訊以及解決錯誤的建議，請參閱下表。您可以透過訂閱 Amazon EventBridge 排程器事件 (事件來源:`aws.pca-connector-ad`) 或在 Windows 中使用手動註冊來收到這些錯誤。

錯誤代碼	意義	修補
FF000	Kerberos 驗證失敗。	請確定您的目錄可連線，而且用戶端是使用者或電腦。如果您使用的是自動註冊，請修正您的 AWS 資源服務主體。如果您正在使用活動目錄 UI 來獲取證書，請運行`gpupdate /force`。
FF001	SOAP 消息必須包含一個操作標頭。	新增動作標頭。
FF002	連接器無法存取其所連線的私有 CA。	透過建立 AWS Resource Access Manager (RAM)，以在私有 CA 和 AD 服務的連接器之間共用，以便與連接器共用您的私有 CA。
FF003	此連接器的私人 CA 未處於作用中狀態。	將私有 CA 移至作用中狀態。如果您的私有 CA 處於擱置憑證狀態，請安裝 CA 憑證。
FF004	此連接器的私人 CA 不存在。	如果憑證授權單位處於「已刪除」狀態，請將其移至「作用中」狀態。如果您的私人 CA 已永久刪除，請使用不同的 CA 建立新的連接器。
FF005	範本指定了`directoryGuid` 憑證主體或主體替代名稱的屬性，但在要求者的 AD 物件中找不到該屬性。	活動目錄沒有`directoryGuid` 為您的目錄生成一個。活動目錄中的疑難排解。

錯誤代碼	意義	修補
FF006	範本指定了dnsHostName 憑證主體或主體替代名稱的屬性，但在要求者的 AD 物件中找不到該屬性。	將dnsHostName 屬性添加到您的 AD 對象。
FF007	範本指定要包含在憑證主旨或主旨替代名稱中的電子郵件屬性，但在要求者的 AD 物件中找不到該屬性。	將電子郵件屬性添加到您的 AD 對象
FF008	SOAP 訊息必須具有http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy/IPolicy/GetPolicies 或的動作標頭http://schemas.microsoft.com/windows/pki/2009/01/enrollment/RST/wstep 。	更新動作標頭以使用其中一個指定值。
FF009	必 BinarySecurityToken 須在中編碼http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary 。	更新二進位安全性權杖類型。
氟氯乙烯	BinarySecurityToken 無效。	檢查 CSR 是否正確產生。

錯誤代碼	意義	修補
FFA00B	值類型 BinarySecurityToken 必須為http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#PKCS7 或http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10 。	將二進位安全性權杖值類型更新為有效值。
氟氯乙烯	包 BinarySecurityToken 含的無效 CMS。	Base64 有效，但密碼編譯訊息語法 (CMS) 無效。檢閱 CMS 語法。
FF00D	包 BinarySecurityToken 含無效的 CSR。	檢查 CSR 是否已正確產生。
FF00E	私有 CA 無法使用特定範本發行憑證。	複查來源的驗證例外 AWS Private CA。您可以在 Amazon EventBridge 或中查看驗證異常 AWS CloudTrail。
FFF	SOAP 訊息必須具有的要求類型http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue 。	將請求類型設定為http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue 。
FF010	SOAP 訊息必須具有連接器 CertificateEnrollmentPolicyServerEndpoint 欄位或 XCEP 回應中的 URI 欄位的 To 標頭。	將請求安全性權杖的標頭設定為 XCEP 回應中的欄位或 URI 欄位。CertificateEnrollmentPolicyServerEndpoint

錯誤代碼	意義	修補
FF011	SOAP 消息必須只有一個操作標題。	檢閱要求安全性權杖的 SOAP 訊息標頭，並正確設定標頭。
FF012	SOAP 消息必須只有一個`messageId` 標題。	檢閱要求安全性權杖的 SOAP 訊息標頭，並正確設定標頭。
FF013	SOAP 消息必須只有一個頭。	檢閱要求安全性權杖的 SOAP 訊息標頭，並正確設定標頭。
0FFA	請求者無法存取要求的範本。	建立存取控制項目，允許要求者的群組使用要求的範本註冊。
FF15	`CertificateTemplateInformation` 或`CertificateTemplateName` 副檔名必須存在於 BinarySecurityToken.	將安全性延伸模組新增至您的 CSR。
FF016	找不到指定連接器的要求範本。	範本是每個連接器的子資源。使用建立連接器的範本`createTemplate` 。
FF017	由於請求調節，因此請求遭到拒絕。	減慢請求的速率。
FF018	SOAP 消息必須包含一個`to`標題。	檢閱 SOAP 訊息的標頭。
FFA19	由於無法識別的標頭，無法處理 SOAP 消息。	檢閱 SOAP 訊息的標頭。

錯誤代碼	意義	修補
FFAA	範本指定要包含在憑證主體或主體替代名稱中的 UPN 屬性，但在要求者的 AD 物件中找不到該屬性。	將 UPN 新增至作用中目錄物件。

連接器建立失敗

連接器建立可能會因各種原因而失敗。連接器建立失敗時，您會在 API 回應中收到失敗原因。如果您使用主控台，則失敗原因會顯示在連接器詳細資料容器中 [其他狀態詳細資料] 欄位下的 [連接器詳細資料] 頁面中。下表說明失敗原因和建議的解決步驟。

失敗狀態	描述	修補
CA_CERTIFICATE_REGISTRATION_FAILED	AD 的連接器無法將 CA 憑證匯入您的目錄。	檢閱 [必要條件] 頁面，並檢查您的服務帳戶是否具有正確的權限。將正確的權限委派給您的服務帳戶後，請刪除失敗的連接器並建立新的連接器。如需委派權限的相關資訊，請參閱《AWS Directory Service 管理指南》中的將權限委派給您的服務帳戶。
DIRECTORY_ACCESS_DENIED	AD 的連接器無法存取您的目錄。	您必須授與連接器，才能存取目錄的 AD。檢閱本IAM 政策節，確保您的 AWS 帳戶相關聯的 IAM 政策可讓您存取和描述目錄。授與 AWS 角色的正確權限後，請刪除失敗的連接器並建立新的連接器。
INTERNAL_FAILURE	AD 的連接器發生內部故障。	請稍後再試。刪除失敗的連接器並建立新的連接器。

失敗狀態	描述	修補
PRIVATECA_ACCESS_DENIED	AD 的連接器無法存取您的私有 CA。	檢閱 [必要條件] 頁面，並檢查您是否具有建立連接器的權限。如需相關資訊，請參閱IAM 政策。 如果您要透過 AWS CLI 或 API 建立連接器，請檢閱 [必要條件] 頁面，並檢查您是否已使用連接器與 AD 共用私有 CA AWS Resource Access Manager。 檢查並修復 IAM 許可和 AWS RAM 資源共用後，請刪除失敗的連接器並建立新的連接器。
PRIVATECA_RESOURCE_NOT_FOUND	AD 的連接器找不到指定的私有 CA。	請確定您指定正確的私有 CA Amazon 資源名稱 (ARN)，然後刪除失敗的連接器，並使用預期的私有 CA ARN 建立新的連接器。
SECURITY_GROUP_NOT_IN_VPC	安全性群組不在託管您目錄的 VPC 中。	使用託管目錄的 VPC 中的安全群組。如需詳細資訊，請參閱安全群組。刪除失敗的連接器，並使用 VPC 中的安全群組建立新的連接器。
VPC_ACCESS_DENIED	AD 的連接器無法存取託管您目錄的 Amazon VPC。	檢查您的 IAM 許可。刪除失敗的連接器並建立新的連接器。如需包含存取權限的 IAM 政策範例，請參閱 IAM 政策

失敗狀態	描述	修補
VPC_ENDPOINT_LIMIT_EXCEEDED	AD 連接器無法在您的 Amazon VPC 中建立端點。您已達到可為帳戶建立的 VPC 端點上限。	刪除 Amazon VPC 人雲端節點，或要求提高限制。完成上述兩個步驟之一後，請刪除失敗的連接器並建立新的連接器。如需配額的相關資訊，請參閱 Amazon Virtual Private Cloud 服務配額。
VPC_RESOURCE_NOT_FOUND	AD 的連接器找不到指定的 VPC。	請確定您指定了正確的 VPC，且 VPC 存在。然後刪除失敗的連接器，並使用正確的 VPC ID 建立新的連接器。

SPN 建立失敗

服務主體名稱 (SPN) 建立可能會因各種原因而失敗。SPN 建立失敗時，您會在 API 回應中收到失敗原因。如果您使用主控台，則失敗原因會顯示在 [連接器詳細資料] 頁面中 [服務主體名稱 (SPN)] 容器內 [其他狀態詳細資料] 欄位下方。下表說明失敗原因和建議的解決步驟。

失敗狀態	描述	修補
DIRECTORY_ACCESS_DENIED	AD 的連接器無法存取您的目錄。	授與連接器，以便 AD 存取您的目錄。如需包含授與目錄存取權的許可的 IAM 政策範例，請參閱IAM 政策。
DIRECTORY_NOT_REACHABLE	AD 的連接器無法存取您的目錄。	檢查 AWS 和目錄之間的網路，然後嘗試再次建立 SPN。
DIRECTORY_RESOURCE_NOT_FOUND	AD 的連接器找不到指定的目錄。	請確定您指定正確的目錄 ID，然後刪除失敗的連接器，並使

失敗狀態	描述	修補
		用預期的目錄 ID 建立新的連接器。
INTERNAL_FAILURE	AD 的連接器發生內部故障。	請稍後再試。
SPN_EXISTS_ON_DIFFERENT_AD_OBJECT	服務主體名稱 (SPN) 存在於不同的「作用中目錄」物件上。	從「作用中目錄」物件刪除 SPN，然後嘗試再次建立 SPN。
SPN_LIMIT_EXCEEDED	AD 的連接器無法建立 SPN，因為您已達到每個目錄 SPN 的限制。每個目錄的 SPN 數目上限為 10。	從您的帳戶中刪除一或多個 SPN，然後嘗試再次建立 SPN。

AD 連接器建立失敗錯誤的連接器

建立 AD 連接器的連接器可能會因各種原因而失敗。連接器建立失敗時，您會在 API 回應中收到失敗原因。如果您使用主控台，則失敗原因會顯示在 [連接器詳細資料] 容器中 [其他狀態詳細資料] 欄位下的 [連接器詳細資料] 頁面中。下表說明失敗原因和建議的解決步驟。

術語和概念

下列術語和概念可以協助您使用 AWS Private Certificate Authority (AWS 私有 CA)。

主題

- 信任
- TLS 伺服器憑證
- 憑證簽章
- 憑證授權單位
- 根 CA
- 加拿大證書
- 根 CA 憑證
- 終端實體證書
- 自我簽署的憑證
- 私人證書
- 憑證路徑
- 路徑長度約束

信任

為了讓 Web 瀏覽器信任網站的身分，瀏覽器必須能驗證網站的憑證。不過，瀏覽器只信任少數稱為 CA 根憑證的憑證。稱為憑證授權機構 (CA) 的信任第三方會驗證網站的身分，並將已簽署的數位憑證發給網站的營運商。然後，瀏覽器便可檢查數位簽章，以驗證網站的身分。如果驗證成功，瀏覽器會在網址列中顯示鎖定圖示。

TLS 伺服器憑證

HTTPS 交易需要伺服器憑證，才能對伺服器進行身分驗證。伺服器憑證是 X.509 v3 資料結構，將憑證中的公有金鑰繫結至憑證主體。TLS 憑證由憑證授權單位 (CA) 簽署。它包含服務器的名稱，有效期，公鑰，簽名算法等。

憑證簽章

數位簽章是透過憑證的加密雜湊。簽章是用於確認憑證資料的完整性。私有 CA 會在變數大小的憑證內容上使用加密雜湊函數 (例如 SHA256) 來建立簽章。這個雜湊函數會產生有效不可偽造的固定大小資料字串。此字串稱為雜湊。然後，CA 會使用其私密金鑰來加密雜湊值，並使用憑證串連加密雜湊。

憑證授權單位

憑證授權機構 (CA) 發行及 (在必要時) 撤銷數位憑證。最常見的憑證類型是根據 ISO X.509 標準。X.509 憑證會確認憑證主體的身分，並將該身分繫結至公開金鑰。主體可以是使用者、應用程式、電腦或其他裝置。CA 透過雜湊內容來簽署憑證，然後使用憑證中與公有金鑰相關的私有金鑰來加密雜湊。Web 瀏覽器等需要確認主體身分的用戶端應用程式使用公開金鑰來解密憑證簽章。然後，用戶端應用程式會雜湊憑證內容，並將雜湊值與解密簽章進行比較，以判斷是否相符。如需憑證簽署的相關資訊，請參閱 憑證簽章。

您可以使用 AWS 私有 CA 建立私有 CA，然後使用私有 CA 發行憑證。私有 CA 只發行私有 SSL/TLS 憑證，以供組織內使用。如需詳細資訊，請參閱 私人證書。您的私有 CA 還需要憑證才能使用。如需詳細資訊，請參閱 加拿大證書。

根 CA

是一種加密建置區塊和信任根，可做為發行憑證的根據。其包含私有金鑰，用於簽署 (發行) 憑證和根憑證，這些憑證可識別根 CA 並將私有金鑰繫結至 CA 名稱。根憑證會散佈至環境中每個實體的信任存放區。管理員會建構信任存放區以僅包含其信任的 CA。系統管理員將信任存放區更新或建置到其環境中實體的作業系統、執行個體和主機機器映像中。當資源嘗試互相連線時，會檢查每個實體呈現的憑證狀態。用戶端會檢查憑證是否有效，以及從憑證到信任存放區中安裝的根憑證是否存在鏈結。如果滿足這些條件，則會在資源之間完成「握手」。此交握會以加密方式來互相證實各實體的身分，然後在這些實體之間建立加密的通訊通道 (TLS/SSL)。

加拿大證書

憑證授權機構 (CA) 憑證確認 CA 的身分，並將該身分繫結至憑證所含的公開金鑰。

您可以使用 AWS 私有 CA 來建立私有根 CA 或私有次級 CA (每個都由 CA 憑證支援)。次級 CA 憑證由信任鏈結中較高層的另一個 CA 憑證所簽署。但是在根 CA 的情況下，憑證會自我簽署。您也可以建立外部根授權機構 (例如裝載在內部部署)。然後您就可以使用根授權機構來簽署由 AWS 私有 CA 裝載的次級根 CA 憑證。

下列範例示範 AWS 私有 CA X.509 CA 憑證所含的一般欄位。請注意，針對 CA 憑證，CA: 欄位中的 Basic Constraints 值設為 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
```

根 CA 憑證

憑證授權單位 (CA) 通常存在於階層式結構內，其中包含多個其他 CA，其間具有明確定義的父子關係。下層 CA 或次級 CA 是由上層 CA 認證，並形成憑證鏈。階層頂端的 CA 稱為根 CA，其憑證稱為根憑證。此憑證通常為自我簽署。

終端實體證書

終端實體憑證可識別資源，例如伺服器、執行個體、容器或裝置。與 CA 憑證不同，終端實體憑證無法用於發行憑證。終端實體憑證的其他一般術語為「用戶端」或「分葉」憑證。

自我簽署的憑證

由發行者 (而不是較高的 CA) 簽署的憑證。與 CA 維護的安全根憑證發行的憑證不同，自我簽署憑證充當自己的根憑證，因此它們具有重大限制：它們可用於提供線路加密，但不能驗證身分，因此無法撤銷。從安全的角度來看，它們是不可接受的。但組織仍會使用這種憑證，因為易於產生、不需要專門技術或基礎設施，而且許多應用程式都能接受。自我簽署憑證的發行無法控管。由於無法追蹤這種憑證的過期日期，所以使用自我簽署憑證的組織，因憑證過期導致服務中斷時間的風險較高。

私人證書

AWS 私有 CA憑證是私有 SSL/TLS 憑證，您可以在組織內使用，但在公用網際網路上不受信任。您可以使用這些憑證來識別資源，例如用戶端、伺服器、應用程式、服務、裝置和使用者。建立安全加密通訊通道時，每個資源皆會使用如下所示的憑證和加密技術來證明其對另一個資源的身分。內部 API 端點、Web 伺服器、VPN 使用者、IoT 裝置和其他多種類型的應用程式皆使用私有憑證來建立安全操作需要的加密通訊通道。在預設情況下，私有憑證不受公開信任。內部管理員必須明確設定應用程式，以信任私有憑證及分配憑證。

```
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
```

憑證路徑

依賴憑證的用戶端會驗證從最終實體憑證 (可能透過中繼憑證鏈結) 存在至受信任的根目錄的路徑。用戶端會檢查路徑上的每個憑證是否有效 (未遭撤銷)。它也會檢查終端實體憑證是否尚未過期、具有完整性 (尚未遭竄改或修改)，以及是否強制執行憑證中的限制。

路徑長度約束

CA 憑證的基本限制pathLenConstraint條件會設定其下方鏈結中可能存在的從屬 CA 憑證數目。例如，路徑長度限制為零的 CA 憑證不能有任何從屬 CA。路徑長度限制條件為 1 的 CA 下方，最多可以有一個層級的次級 CA。RFC 5280 將其定義為「在有效憑證路徑中可能遵循此 non-self-issued 憑證的中繼憑證數目上限」。路徑長度值不包括最終實體證書，儘管關於驗證鏈「長度」或「深度」的非正式語言可能包含它... 導致混淆。

文件歷史記錄

下表說明此文件自 2018 年 1 月後的重大變更。除了這裡所列的主要變更外，我們也會經常更新文件以改進說明內容和範例，並且反映您傳送給我們的意見回饋。如要接收重大變更的通知，請使用右上角的連結來訂閱 RSS 摘要。

變更	描述	日期
更新 AD 疑難排解指引的連接器	如果您收到 AD 的 Connector 執行個體發`Kerberos authentication failed`生錯誤，請確定您的目錄可連線，而且用戶端是使用者或電腦。	2024年7月3日
新增稽核報告的限制	AWS Private CA 不支援將 Amazon S3 物件鎖定與用於稽核報告的儲存貯體搭配使用。	2024年7月3日
現在支援中國地區的 SM2	AWS Private CA 現在支援 SM2 簽名演算法，僅適用於中國地區。	2024年6月27日
AWS Private CA 現在支援用於 SCEP 的連接器 (預覽)	使用適用於 SCEP 的連接器來連結 AWS Private CA 至已啟用 SCEP 的用戶端和裝置。	2024年6月11日
新的連接器疑難排解	已新增連接器疑難排解和 SPN 建立失敗的兩個新章節。	2024年4月4日
新增物件的 CDP 副檔名	添加對物質的憑證撤銷清單發佈點 (CDP) 延伸的支援。	2024年1月25日
AWS Private CA 針對 MDL 的 API 支援	新增 API 支援，可建立符合行動駕駛執照 (MDL) ISO/IEC 標準的憑證。	2024年1月16日

AWS Private CA 作用中目錄的連接器	AD 連接器的使用者指南、API 和 CLI 支援。如需詳細資訊，請參閱 AD 說明文件的連接器。	2023 年 8 月 24 日
變更安全性原則名稱以符合新的服務名稱	針對指定標準許可的 AWS 受管 IAM 政策採用新名稱 AWS 私有 CA。如需詳細資訊，請參閱AWS 受管理的策略。	2023 年 2 月 13 日
新增 AWS 受管理策略的變更追蹤器	已新增用來追蹤指定標準許可的 AWS 受管 IAM 政策變更的文件 AWS 私有 CA。如需詳細資訊，請參閱的受 AWS 管理原則更新 AWS 私有 CA。	2022 年 11 月 11 日
針對發行短期憑證的 CA 提供 API 和 CLI 支援	隨著 CA 使用模式的引入，CA 可以設定為發行一般用途或專用的短期憑證。如需詳細資訊，請參閱憑證授權單位模式。	2022 年 10 月 24 日
服務品牌重塑和主控台更新	服務已重新命名為 AWS Private Certificate Authority (AWS 私有 CA)。 AWS 私有 CA 控制台獲得可用性改進，包括鏈接到完整文檔的集成幫助面板。	2022 年 9 月 27 日
符合事項規範的憑證支援	三個新的憑證範本新增了對符合 Matter 標準的 CA 和最終實體憑證的支援。如需詳細資訊，請參閱瞭解憑證範本。	2022 年 7 月 20 日

新區域支援	為亞太區域 (雅加達) 新增端點。如需 AWS Private CA 端點的完整清單，請參閱 ACM 私有憑證授權單位端點和配額。	2022 年 5 月 4 日
Support 自訂屬性和擴充功能	使用CustomAttribute物件來設定自訂 CA 和憑證，並使用CustomExtension 物件來設定自訂憑證。	2022 年 3 月 16 日
Support 管理式 OCSP	請參閱針對包括 OCSP 在內的撤銷選項設定憑證撤銷方法。	2021 年 8 月 18 日
Support 適用於 CRL 的 S3 區塊公開存取功能	請參閱啟用 S3 封鎖公用存取功能。	2021 年 5 月 27 日
新的和更新的 Java 實現示例	請參閱使用 ACM 私有 CA API (Java 範例)。	2020 年 9 月 9 日
新區域支援	新增了非洲 (開普敦) 和歐洲 (米蘭) 的端點。如需 AWS 私有 CA 端點的完整清單，請參閱AWS Certificate Manager 私有憑證授權單位端點和配額。	2020 年 8 月 27 日
支援跨帳戶私有 CA 存取	AWS Certificate Manager 使用者可以獲得授權，使用他們不擁有的私有 CA 發行憑證。如需詳細資訊，請參閱私人 CA 的跨帳戶存取。	2020 年 8 月 17 日
VPC 端端點 (PrivateLink) 支援	已新增對使用 VPC 端點 (AWS PrivateLink) 的支援，以增強網路安全性。如需詳細資訊，請參閱 ACM 私有 CA VPC 端點 ()AWS PrivateLink。	2020 年 3 月 26 日

增加了專用安全部分	的安全性文件 AWS 已整合到專用的安全性區段中。如需安全性的相關資訊，請參閱AWS Certificate Manager 私有憑證授權單位中的安	2020 年 3 月 26 日
範本 ARN 已新增至稽核報告。	如需詳細資訊，請參閱為您的私有 CA 建立稽核報告。	2020 年 3 月 6 日
CloudFormation 支持	Support 添加了 AWS CloudFormation. 如需詳細資訊，請參閱《 AWS CloudFormation 使用指南》中的《ACMPCA 資源類型參考》。	2020 年 1 月 22 日
CloudWatch 活動整合	與非同步 CloudWatch 事件的事件整合，包括建立 CA、憑證發行和 CRL 建立。如需詳細資訊，請參閱使用 CloudWatch 事件。	2019 年 12 月 23 日
FIPS 端點	為 AWS GovCloud (美國東部) 和 AWS GovCloud (美國西部) 新增 FIPS 端點。如需 AWS 私有 CA 端點的完整清單，請參閱AWS Certificate Manager 私有憑證授權單位端點和配額。	2019 年 12 月 13 日
基於標籤的權限	標籤式許可支援使用 `TagResource` 、`UntagResource` 和 `ListTagsForResource` 等新的 API。如需標籤式控制的一般資訊，請參閱使用 IAM 資源標籤控制 IAM 使用者和角色的存取權。	2019 年 11 月 5 日

名稱限制強制	新增了對所匯入 CA 憑證實施主體名稱限制條件的支援。如需詳細資訊，請參閱對私有 CA 實施名稱限制條件。	2019 年 10 月 28 日
新的憑證範本	添加了新的證書模板，包括用於代碼簽名的模板 AWS Signer。如需詳細資訊，請參閱使用範本。	2019 年 10 月 1 日
規劃您的 CA	新增了使用 AWS 私有 CA規劃 PKI 的新章節。如需詳細資訊，請參閱規劃您的 ACM 私有 CA 部署。	2019 年 9 月 30 日
已新增的區域支援	新增 AWS 亞太區域 (香港) 區域支援。如需支援區域的完整清單，請參閱AWS Certificate Manager 私有憑證授權單位端點和配額。	2019 年 7 月 24 日
新增完整的私有 CA 階層支援	建立和託管根 CA 的 Support 會消除對外部父項的需求。	2019 年 6 月 20 日
已新增的區域支援	新增對 AWS GovCloud (美國西部和美國東部) 區域的區域支援。如需支援區域的完整清單，請參閱 Cer AWS tificate Manager 私人憑證授權單位端點和配額。	2019 年 5 月 8 日
已新增的區域支援	新增 AWS 亞太區域 (孟買和首爾)、美國西部 (加利佛尼亞北部) 和歐洲 (巴黎和斯德哥爾摩) 區域的區域支援。如需支援區域的完整清單，請參閱AWS Certificate Manager 私有憑證授權單位端點和配額。	2019 年 4 月 4 日

測試憑證更新工作流	客戶現在可以手動測試 ACM 受管續約工作流程的組態。如需更多資訊，請參閱測試 ACM 受管續約的組態。	2019 年 3 月 14 日
已新增的區域支援	新增對 AWS 歐盟 (倫敦) 區域的區域支援。如需支援區域的完整清單，請參閱 Cer AWS tificate Manager 私人憑證授權單位端點和配額。	2018 年 8 月 1 日
還原已刪除的 CA	私有 CA 還原允許客戶在憑證授權單位 (CA) 被刪除之後高達 30 天內，還能夠將其還原。如需詳細資訊，請參閱還原您的私有 CA。	2018 年 6 月 20 日

舊版更新

下表說明 2018 年 6 月 AWS Private Certificate Authority 之前的文件發行歷史記錄。

變更	描述	日期
新指南	此版本推出 AWS Private Certificate Authority。	2018 年 4 月 04 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。