AWS Certificate Manager
Private Certificate Authority

사용 설명서
버전 latest
AWS Certificate Manager Private Certificate Authority: 사용 설명서
Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.
# Table of Contents

AWS Certificate Manager Private
Certificate Authority 사용 설명서

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM Private CA이란 무엇입니까?</td>
<td>1</td>
</tr>
<tr>
<td>내 요구에 가장 적합한 인증서 서비스는 무엇입니까?</td>
<td>1</td>
</tr>
<tr>
<td>Regions</td>
<td>2</td>
</tr>
<tr>
<td>통합 서비스</td>
<td>2</td>
</tr>
<tr>
<td>할당량</td>
<td>2</td>
</tr>
<tr>
<td>인증서에 대한 할당량</td>
<td>2</td>
</tr>
<tr>
<td>API 요청에 대한 할당량</td>
<td>3</td>
</tr>
<tr>
<td>RFC 규정 준수</td>
<td>4</td>
</tr>
<tr>
<td>요금</td>
<td>5</td>
</tr>
<tr>
<td>보안</td>
<td>6</td>
</tr>
<tr>
<td>데이터 보호</td>
<td>6</td>
</tr>
<tr>
<td>ACM Private CA 프라이빗 키 저장</td>
<td>7</td>
</tr>
<tr>
<td>ID 및 액세스 관리</td>
<td>7</td>
</tr>
<tr>
<td>인증</td>
<td>7</td>
</tr>
<tr>
<td>리소스, 소유권 및 관리 정책 이해</td>
<td>8</td>
</tr>
<tr>
<td>인라인 정책</td>
<td>8</td>
</tr>
<tr>
<td>로깅 및 모니터링</td>
<td>14</td>
</tr>
<tr>
<td>CloudWatch 지표</td>
<td>16</td>
</tr>
<tr>
<td>CloudWatch 이벤트 사용</td>
<td>16</td>
</tr>
<tr>
<td>사용 CloudTrail</td>
<td>21</td>
</tr>
<tr>
<td>규정 준수 확인</td>
<td>32</td>
</tr>
<tr>
<td>감사 보고서 생성</td>
<td>32</td>
</tr>
<tr>
<td>인프라 보완</td>
<td>36</td>
</tr>
<tr>
<td>VPC 엔드포인트(AWS PrivateLink)</td>
<td>36</td>
</tr>
<tr>
<td>ACM Private CA 모범 사례</td>
<td>38</td>
</tr>
<tr>
<td>CA 구조 및 정책 문서화</td>
<td>39</td>
</tr>
<tr>
<td>가능한 경우 루트 CA 사용 최소화</td>
<td>39</td>
</tr>
<tr>
<td>루트 CA에 자체 AWS 계정 제공</td>
<td>39</td>
</tr>
<tr>
<td>AWS CloudTrail 켜기</td>
<td>39</td>
</tr>
<tr>
<td>CA 프라이빗 키 업데이트</td>
<td>40</td>
</tr>
<tr>
<td>미사용 CA 삭제</td>
<td>40</td>
</tr>
<tr>
<td>사설 CA 계획</td>
<td>41</td>
</tr>
<tr>
<td>CA 계층 구조 설계</td>
<td>41</td>
</tr>
<tr>
<td>최종 엔터티 인증서 효율성 검사</td>
<td>43</td>
</tr>
<tr>
<td>CA 계층 구조 계획</td>
<td>44</td>
</tr>
<tr>
<td>인증 계층에서 길이 제약 조건 설정</td>
<td>45</td>
</tr>
<tr>
<td>CA 수명 주기 관리</td>
<td>47</td>
</tr>
<tr>
<td>유효 기간 선택</td>
<td>47</td>
</tr>
<tr>
<td>CA 승계 관리</td>
<td>48</td>
</tr>
<tr>
<td>CA 해지</td>
<td>49</td>
</tr>
<tr>
<td>복원성</td>
<td>49</td>
</tr>
<tr>
<td>이중화 및 재해 복구</td>
<td>49</td>
</tr>
<tr>
<td>사설 CA 생성 및 관리</td>
<td>50</td>
</tr>
<tr>
<td>설정</td>
<td>50</td>
</tr>
<tr>
<td>AWS에 가입</td>
<td>50</td>
</tr>
<tr>
<td>CLI 설치(선택 사항)</td>
<td>51</td>
</tr>
<tr>
<td>사설 CA 생성</td>
<td>51</td>
</tr>
<tr>
<td>CRL 암호화</td>
<td>55</td>
</tr>
<tr>
<td>CA 인증서 설치</td>
<td>57</td>
</tr>
<tr>
<td>루트 CA 인증서 설치 중인 경우</td>
<td>57</td>
</tr>
<tr>
<td>ACM Private CA에서 호스팅하는 하위 CA 인증서를 설치하는 경우</td>
<td>57</td>
</tr>
<tr>
<td>외부 상위 CA에서 서명한 하위 CA 인증서를 설치하는 경우</td>
<td>58</td>
</tr>
<tr>
<td>CA 업데이트</td>
<td>59</td>
</tr>
</tbody>
</table>

버전 latest

iii
ACM Private CA이란 무엇입니까?

ACM Private CA를 사용하면 온프레미스 CA를 운영하는 데 드는 투자 및 유지 관리 비용 없이 루트 및 하위 CA를 비롯한 사설 CA 계층을 생성할 수 있습니다. 사설 CA는 다음과 같은 시나리오에서 유용한 최종 엔터티 X.509 인증서를 발급할 수 있습니다.

- 암호화된 TLS 통신 채널 생성
- 사용자, 컴퓨터, API 엔드포인트 및 IoT 디바이스 인증
- 암호화 서명 코드
- 인증서 해지 상태를 얻기 위한 온라인 인증서 상태 프로토콜(OCSP) 구현

ACM Private CA 작업은 AWS Management 콘솔에서 ACM Private CA API 또는 AWS CLI를 사용하여 액세스할 수 있습니다.

내 요구에 가장 적합한 인증서 서비스는 무엇입니까?

1. ACM Private CA—이 서비스는 AWS 클라우드 내부에 PKI(퍼블릭 키 인프라)를 구축하는 기업 고객을 대상으로 하며 조직 내에서 공유로 사용할 수 있도록 고안되었습니다. ACM Private CA를 사용하여 고유한 CA 계층을 생성하고 사용자, 컴퓨터, 애플리케이션, 서비스, 서버 및 기타 디바이스를 인증하기 위해 인증서를 발급할 수 있습니다. 사설 CA에서 발급한 인증서는 인터넷이 아닌 조직 내에서만 신뢰할 수 있습니다.

사설 CA를 생성하면 외부 CA에서 유효성 검사를 받지 않고 직접 인증서를 발급하고 조직의 내부 요구 사항에 맞게 사용자 지정할 수 있습니다. 예를 들어 다음을 수행할 수 있습니다.

- 보안 주체 이름으로 인증서를 생성합니다.
- 만료 날짜가 있는 인증서를 생성합니다.
- 지원되는 모든 프라이빗 키 알고리즘과 키 길이 사용 가능.
- 지원되는 모든 서명 알고리즘 사용 가능.
- 템플릿을 사용하여 인증서 발급을 제어합니다.


2. AWS Certificate Manager(ACM)—이 서비스는 TLS를 사용하여 공개적으로 신뢰할 수 있는 보안 웹이 필요한 엔터프라이즈 고객을 위해 인증서를 관리합니다. AWS Elastic Load Balancing, Amazon CloudFront, Amazon API Gateway 및 기타 통합 서비스에 ACM 인증서를 배포할 수 있습니다. 이러한 종류의 가장 일반적인 애플리케이션은 중요한 트래픽 요구 사항을 가진 안전한 공개 웹사이트입니다.

이 서비스에서는 ACM에서 제공하는 공인 인증서(ACM 인증서) 또는 ACM으로 가져온 인증서를 사용할 수 있습니다. ACM Private CA를 사용하여 CA를 생성하는 경우, ACM은 해당되는 사설 CA에서 인증서 발급을 관리하고 인증서 형식을 자동화할 수 있습니다.

자세한 내용은 AWS Certificate Manager 사용 설명서를 참조하십시오.
통합 서비스 (p. 2)
할당량 (p. 2)
RFC 규정 준수 (p. 4)
요금 (p. 5)

Regions

대부분의 AWS 리소스와 마찬가지로 사설 CA는 리전별 리소스입니다. 하나 이상의 리전에서 사설 CA를 사용하려면 이러한 리전에서 CA를 생성해야 합니다. 리전 간 사설 CA를 복사할 수 없습니다. ACM Private CA에 대한 리전별 가능성을 확인하려면 AWS General Reference의 AWS 리전 및 엔드포인트 또는 AWS 리전 표를 참조하십시오.

Note
ACM은 현재 일부 리전(ACM Private CA가 아님)에서 사용 가능합니다.

AWS Certificate Manager Private Certificate Authority에 통합되는 서비스

AWS Certificate Manager를 사용하여 사설 인증서를 요청하는 경우, 해당 인증서를 ACM에 통합된 모든 서비스와 연결할 수 있습니다. 이는 ACM Private CA 루트에 연결된 인증서와 외부 루트에 연결된 인증서에 모두 적용됩니다. 자세한 내용은 AWS Certificate Manager 사용 설명서의 통합 서비스를 참조하십시오.

ACM Private CA API 또는 AWS CLI를 사용하여 인증서를 발급하거나 ACM에서 사설 인증서를 내보내는 경우에는 원하는 위치에 인증서를 설치할 수 있습니다.

할당량

AWS Certificate Manager Private Certificate Authority는 허용된 인증서 및 인증 기관 수 및 API 속도에 할당량을 지정합니다.

주제
• 인증서에 대한 할당량 (p. 2)
• API 요청에 대한 할당량 (p. 3)

인증서에 대한 할당량

각 리전 및 계정에 다음과 같은 ACM Private CA 인증서 할당량이 적용됩니다. 할당량 증가를 요청하려면 AWS Support 센터에서 사례를 생성합니다.

<table>
<thead>
<tr>
<th>항목</th>
<th>기본 할당량</th>
</tr>
</thead>
<tbody>
<tr>
<td>사설 CA의 수</td>
<td>10*</td>
</tr>
<tr>
<td>사설 CA별 사설 인증서 수</td>
<td>1,000,000*</td>
</tr>
<tr>
<td>CA당 만료되지 않고 해지된 사설 인증서 수**</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>
* 이러한 항목에 대한 할당량 증가를 요청할 수 있습니다. AWS Support 센터를 방문하여 사례 생성을 선택한 다음 Service limit increase(서비스 한도 증가)를 선택합니다.

** 이 할당량은 CRL을 사용하는 클라이언트가 처리할 수 있는 최대 CRL 크기를 기준으로 인증서 해지 목록(CRL)에 포함시킬 수 있는 만료되지 않은 인증서 수를 반영합니다. 이 할당량은 늘릴 수 없습니다.

Note

삭제된 사설 CA는 복원 기간이 종료될 때까지 인증서 할당량에 집계됩니다. 자세한 내용은 사설 CA 삭제를 참조하십시오.

ACM Private CA는 ACM에 통합됩니다. ACM 콘솔, AWS CLI 또는 ACM API를 사용하여 기존 사설 인증기관(CA)에서 사설 인증서를 요청할 수 있습니다. 이 인증서는 ACM에서 관리하며, 제한 사항은 ACM에서 발행한 공용 인증서와 동일합니다. 제한 사항 목록은 사설 인증서 요청을 참조하십시오. 또한 ACM Private CA API 또는 AWS CLI를 사용하여 사설 인증서를 발급할 수도 있습니다. 자세한 내용은 사설 최종 엔터티 인증서 발급(p. 68) 참조하십시오. 사용하는 방법에 관계없이 사설 CA를 10개 생성하고, 사설 인증서를 1,000,000개 요청하고, 리전별로 계정당 1,000,000개의 인증서를 해지할 수 있습니다. ACM는 공용 인증서와 가져온 인증서에 할당량을 적용합니다. 자세한 내용은 ACM 할당량을 참조하십시오.

API 요청에 대한 할당량

각 리전 및 계정의 ACM Private CA API에 다음 할당량이 적용됩니다. ACM Private CA API 작업에 따른 할당량은 서로 달리하여 API 요청을 조절합니다. 조절이란 요청이 초당 요청 수에 대한 해당 작업의 할당량을 초과하기 때문에 다른 경우라면 유효한 요청을 ACM Private CA가 거부하는 것을 의미합니다. 요청이 초과되면 ACM Private CA가 ThrottlingException 오류를 반환합니다. 아래 표에는 각 API 작업과 ACM Private CA API에 해당 작업에 대한 요청을 초과하는 비율이 나와 있습니다. ACM Private CA API에 대한 최소 요청 비율을 보장하지 않습니다.

Note

ThrottlingException 오류가 발생하는 경우, 지원 팀에 문의하기 전에 작업을 다시 시도해보는 것이 좋습니다.

각 ACM Private CA API 작업에 대한 초당 요청 할당량

<table>
<thead>
<tr>
<th>API 작업</th>
<th>초당 요청</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateCertificateAuthority</td>
<td>1</td>
</tr>
<tr>
<td>CreateCertificateAuthorityAuditReport</td>
<td>1</td>
</tr>
<tr>
<td>CreatePermission</td>
<td>1</td>
</tr>
<tr>
<td>DeleteCertificateAuthority</td>
<td>10</td>
</tr>
<tr>
<td>DeletePermission</td>
<td>1</td>
</tr>
<tr>
<td>DescribeCertificateAuthority</td>
<td>20</td>
</tr>
<tr>
<td>DescribeCertificateAuthorityAuditReport</td>
<td>20</td>
</tr>
<tr>
<td>GetCertificate</td>
<td>75*</td>
</tr>
<tr>
<td>GetCertificateAuthorityCertificate</td>
<td>20</td>
</tr>
<tr>
<td>GetCertificateAuthorityCsr</td>
<td>10</td>
</tr>
<tr>
<td>ImportCertificateAuthorityCertificate</td>
<td>10</td>
</tr>
<tr>
<td>IssueCertificate</td>
<td>25*</td>
</tr>
</tbody>
</table>
### RFC 규정 준수

ACM Private CA는 RFC 5280에 정의된 특정 제약 조건을 적용하지 않습니다. 반대의 상황도 마찬가지입니다. 사설 CA에 해당되는 특정한 추가 제약 조건이 적용됩니다.

**적용**

- **날짜 이후에 적용되지 않음** ACM Private CA는 RFC 5280에 Not After 날짜보다 늦은 인증서는 발급되지 않도록 합니다.
- **기본 제약 조건** ACM Private CA는 가져온 CA 인증서에 기본 제약 조건과 경로 길이를 적용합니다.

  기본 제약 조건은 인증서에 의해 식별된 CA 인증서의 인증 경로에서 CA 인증서가 아닌 경우에는 인증서가 아닌 경우를 포함합니다. ACM Private CA는 다음과 같은 이유로 인한 유효성 검사 예외 실패를 통해 기본 제약 조건을 적용합니다.

  - **확장** AC 인증서에 포함되지 않습니다.
  - **확장** critical로 표시되어 있지 않습니다.

경로 길이는 유효성 검사에서 가져온 CA 인증서 아래에 있는 유효한 인증 경로의 최대 길이를 결정합니다. ACM Private CA는 다음과 같은 이유로 인한 유효성 검사 예외 실패를 통해 경로 길이를 적용합니다.

- **인증서를 가져오면** CA 인증서 또는 체인의 CA 인증서에서 경로 길이 제약 조건을 위반할 수 있습니다.
- **인증서를 발급하면** 경로 길이 제약 조건을 위반할 수 있습니다.

- **이름 제약 조건** CA에 대한 이러한 제약 조건에 따라 다음과 같은 이유로 인한 유효성 검사 예외 실패를 통해 이름 제약 조건을 적용합니다.

적용되지 않음

- **정책 제약 조건** 이러한 제약 조건은 CA의 하위 CA 인증서 발급 능력을 제한합니다.
- **보안 주체 키 식별자(SKI) 및 인증 기관 키 식별자(AKI)**, RFC는 CA 인증서의 SKI 확장이 포함되도록 요구합니다. CA에서 발급한 인증서는 CA 인증서의 SKI와 일치하는 AKI 확장이 포함되어야 합니다. AWS는 이러한 요구 사항을 적용하지 않습니다. CA 인증서의 SKI가 포함되지 않으면 발급된 최종 인터페이스 또는 하위 CA 인증서 AKI가 대신하여 발급자 퍼블릭 키의 SHA-1 해시가 됩니다.
요금

계정에는 사설 CA를 생성한 시점부터 각 사설 CA에 대한 월별 요금이 청구됩니다. 발급한 각 인증서에 대해 서도 요금이 청구됩니다. 여기에는 ACM에서 내보낸 인증서와 ACM Private CA API 또는 ACM Private CA CLI에서 생성한 인증서가 포함됩니다. 삭제된 사설 CA에 대해서도 요금이 부과되지 않습니다. 그러나 사설 CA를 복원하면 삭제부터 복원까지의 기간에 대한 요금이 청구됩니다. 사용자가 프라이빗 키에 액세스할 수 없는 사설 인증서는 무료입니다. 여기에는 Elastic Load Balancing, CloudFront, API 게이트웨이 등 통합 서비스와 함께 사용되는 인증서가 포함됩니다. ACM Private CA 요금에 대한 최신 정보는 AWS 웹 사이트의 ACM 요금 페이지를 참조하십시오.
AWS Certificate Manager Private Certificate Authority의 보안

AWS에서는 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객은 보안에 가장 관련된 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와 귀하의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

• 클라우드의 보안 – AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호합니다. AWS는 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램의 일환으로 보안 효과를 정기적으로 테스트하고 검증합니다. AWS Certificate Manager Private Certificate Authority에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하십시오.

• 클라우드 내 보안 – 귀하의 책임은 귀하가 사용하는 AWS 서비스에 의해 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 ACM Private CA 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목록에 맞게 ACM Private CA를 구성하는 방법을 보여줍니다. 또한 ACM Private CA 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 배우게 됩니다.

주제
• AWS Certificate Manager Private Certificate Authority에서 데이터 보호 (p. 6)
• AWS Certificate Manager Private Certificate Authority의 Identity and Access Management (p. 7)
• AWS Certificate Manager Private Certificate Authority의 로깅 및 모니터링 (p. 16)
• AWS Certificate Manager Private Certificate Authority의 규정 준수 확인 (p. 32)
• AWS Certificate Manager Private Certificate Authority의 인프라 보안 (p. 36)
• ACM Private CA 모범 사례 (p. 38)

AWS Certificate Manager Private Certificate Authority에서 데이터 보호

AWS Certificate Manager Private Certificate Authority는 AWS 공동 책임 모델을 준수하며, 여기에는 데이터 보호 관련 규정 및 지침이 포함됩니다. AWS는 모든 AWS 서비스를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. AWS는 고객 콘텐츠 및 개인 데이터의 처리를 위한 보안 구성 제어 등이 인프라에서 호스팅되는 데이터에 대한 제어권을 유지합니다. 데이터 제어자 또는 데이터 처리자의 역할을 담당하는 AWS 고객과 APN 파트너는 AWS Cloud에 올린 모든 개인 데이터에 대해 책임을 집니다.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 개별 사용자 계정을 설정하여 AWS Identity and Access Management(IAM) (p. 7)를 사용하여 ACM Private CA API 함수에 대한 액세스를 제어하는 것이 좋습니다. 이렇게 하면 각 사용자에게 자신의 직무를 이행하는 데 필요한 권한만 부여할 수 있습니다. 또한 다음과 같은 방법으로 데이터 보호하는 것이 좋습니다.

• 각 계정마다 멀티 팩터 인증(MFA)을 사용합니다.
• SSL/TLS를 사용하여 AWS 리소스와 통신합니다.

• AWS CloudTrail (p. 21)로 API 및 사용자 활동 로깅을 설정합니다.
AWS Certificate Manager Private Certificate Authority의 Identity and Access Management

ACM Private CA에 액세스하려면 AWS가 요청을 인증하는 데 사용할 수 있는 자격 증명이 필요합니다. 다음 주제에서는 사설 CA에 액세스할 수 있는 대상을 제어하여 사설 CA를 보호할 수 있도록 AWS Identity and Access Management(IAM)을 사용하는 방법에 대한 세부 정보를 제공합니다.

인증

다음과 같은 ID 유형으로 AWS에 액세스할 수 있습니다.

• AWS 계정 루트 사용자 – AWS 계정을 처음 생성할 때 해당 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 SSO(Single Sign-In) ID로 시작합니다. 이 자격 증명은 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일반적인 작업은 물론 관리 작업에도 루트 사용자에게 사용하지 않는 것이 좋습니다. 대신 IAM 사용자를 처음 생성할 때 루트 사용자 액세스 권한은 전부 계정에 대한 관리 권한을 관리 할 수 있는 IAM 사용자로 전부 전달해야 합니다. 그런 다음 루트 사용자 자격 증명을 안전하게 보관해 두고 몇 가지 계정 및 서비스 관리 작업을 수행할 때만 해당 자격 증명을 사용합니다.

• IAM 사용자 – IAM 사용자는 특정 사용자 지정 권한(예: ACM Private CA에서 a directory을 생성할 권한)이 있는 AWS 계정 내 자격 증명입니다. IAM 사용자 이름과 암호를 사용하여 AWS Management 콘솔, AWS 토큰 포럼 또는 AWS Support Center 같은 보안 AWS 웹 페이지에 로그인할 수 있습니다.

사용자 이름과 암호 외에도 각 사용자에 대해 액세스 키를 생성할 수 있습니다. 여러 SDK 중 하나를 통해 또는 AWS Command Line Interface(CLI)를 사용하여 AWS 제품에 프로그래밍 방식으로 액세스할 수 있으며 이러한 키를 사용할 수 있습니다. SDK 및 CLI 도구는 액세스 키를 사용하여 암호화 방식으로 요청에 서명합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. ACM Private CA supports는 인바운드 API 요청을 인증하기 위한 포트폴리온 서명 버전 4를 지원합니다. 요청 인증에 대한 자세한 내용은 AWS General Reference의 서명 버전 4 서명 프로세스 단원을 참조하십시오.

• IAM 역할 – IAM 역할은 특정 권한을 가진 계정에서 생성할 수 있는 IAM 자격 증명입니다. IAM 역할은 AWS에서 자격 증명으로 할 수 있는 것과 할 수 없는 것을 결정하는 권한 정책을 포함하는 AWS 자격 증명이라는 점에서 IAM 사용자와 유사합니다. 그러나 역할은 한 사용자만 연관되지 않고 해당 역할이 필요한 사람이면 누구든지 만들 수 있어야 합니다. 또한 역할에는 그와 연관된 암호 또는 액세스 키와 같은 표준
장기 자격증명이 없습니다. 대신에 역할을 수행할 사람에게는 해당 역할 재선을 위한 임시 보안 자격증명이 제공됩니다. 임시 자격증명이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 연합된 사용자 역할 - IAM 사용자를 만드는 대신 AWS Directory Service의 기존 자격증명, 엔터프라이즈 사용자 디렉터리 또는 웹 자격증명 공급자를 사용할 수 있습니다. 이 사용자를 연합된 사용자라고 합니다. AWS에서는 이 중 ID 공급자를 통해 역할을 추천되며 연합된 사용자에게 역할을 할당합니다. 연합된 사용자에 대한 자세한 내용은 IAM 사용 설명서의 연합된 사용자 및 역할 단원을 참조하십시오.

- AWS 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 사용하는 IAM 역할입니다. 서비스 역할은 해당 계정 내에서만 역할을 제공하며 다른 계정의 서비스에 대한 액세스를 부여하는 데 사용할 수 없습니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 역 할을 만들어 AWS 서비스에 권한 위임 단원을 참조하십시오.

- Amazon EC2에서 실행 중인 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 자격증명을 읽을 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여 단원을 참조하십시오.

리소스, 소유권 및 권한 정책 이해

ACM Private CA에서 사용되는 기본 리소스는 사설 인증 기관(PCA)입니다. 사용자가 소유 또는 제어하는 모든 사설 CA는 다음과 같은 형식을 가진 Amazon 리소스 이름(ARN)으로 식별됩니다.

```
```

리소스 소유자는 AWS 리소스가 생성되는 AWS 계정의 보안 주체 개체입니다. 다음 예에서는 이 계정의 작동 방식을 설명합니다.

- AWS 계정 루트 사용자의 자격증명을 사용하여 사설 CA를 생성하면 AWS 계정이 CA를 소유하게 됩니다.
- 자신의 AWS 계정에서 IAM 사용자를 만들면 이 사용자에게 사설 CA를 생성할 수 있는 권한을 부여할 수 있습니다. 하지만 해당 사용자가 속한 계정이 CA의 소유자가 됩니다.
- AWS 계정에서 IAM 역할을 생성하고 이 역할이 사설 CA를 생성할 수 있는 권한을 부여한 경우, 해당 역할을 담당할 수 있는 사람은 누구나 CA를 생성할 수 있습니다. 하지만 해당 역할이 속한 계정이 사설 CA의 소유자가 됩니다.

권한 정책은 누가 무엇에 액세스 할 수 있는지를 나타냅니다. 다음 단원에서는 권한 정책을 생성하는 데 사용할 수 있는 옵션에 대해 설명합니다.

Note

이 설명서에서는 ACM Private CA의 맥락에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 정보는 다루지 않습니다. IAM 설명서 전체 내용은 IAM 사용 설명서 참조하십시오. IAM 정책 구문과 설명에 대한 자세한 내용은 AWS IAM Policy Reference를 참조하십시오.

IAM 자격증명에 연결할 액세스 제어 및 권한 정책(자격증명 기반 정책)을 설정할 때 다음 표를 참조로 사용합니다. 표의 첫 번째 열에는 각각의 ACM Private CA API 작업이 나열되어 있습니다. 정책의 Action 요소에 작업을 지정합니다. 남은 열에서 정보를 추가로 제공합니다.
<table>
<thead>
<tr>
<th>ACM Private CA API 작업</th>
<th>필요한 권한</th>
<th>리소스</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreatePermission</td>
<td>acm-pca:CreatePermission</td>
<td>arn:aws:acm-pca:AWS_Region:AWS_Account:certificate-authority/certificate_ID</td>
</tr>
<tr>
<td>ListCertificateAuthorities</td>
<td>acm-pca:ListCertificateAuthorities</td>
<td>해당 사항 없음</td>
</tr>
<tr>
<td>ListTags</td>
<td>acm-pca:ListTags</td>
<td>해당 사항 없음</td>
</tr>
</tbody>
</table>

버전 latest

9
IAM을 사용하여 IAM 사용자, 그룹 또는 역할에 권한을 적용하는 정책을 만들 수 있습니다. 이를 자격 증명 기반 정책이라고 합니다. IAM은 다음과 같은 유형의 자격 증명 기반 정책을 제공합니다.

- AWS 관리형 정책 (p. 10)는 기본적으로 ACM Private CA에서 사용할 수 있는 정책입니다. 이러한 정책은 기본 사용자 역할을 다룹니다.
- 고객 관리형 정책 (p. 13)은 AWS 계정에서 생성 및 관리하는 정책으로, 여러 사용자, 그룹 및 역할에 연결할 수 있습니다. 고객 관리형 정책을 사용할 경우에는 AWS 관리형 정책을 사용할 때보다 더욱 정밀한 정책 제어가 가능합니다.
- 인라인 정책 (p. 14)는 사용자가 생성 및 관리하고 단일 사용자, 그룹 또는 역할에 직접 포함되는 정책입니다.

**AWS 관리형 정책**

ACM Private CA에는 관리자, 사용자 및 감사자를 위해 사전 정의된 AWS 관리형 정책의 집합이 포함되어 있습니다. 이러한 정책을 이해하면 고객 관리형 정책 (p. 13)을 구현하는 데 도움이 될 수 있습니다.

- FullAccess – 무제한 관리 제어.

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

- ReadOnly – 읽기 전용 API 작업으로 액세스 제한.

```json
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "acm-pca:DescribeCertificateAuthority"
        ],
        "Resource": "*"
    }
}
```
• PrivilegedUser – CA 인증서를 발급 및 해지하는 기능. 이 정책에는 다른 관리 기능이 없으며 최종 엔터티 인증서를 발급할 수 있는 기능이 없습니다. 권한은 사용자 정책과 상호 배타적입니다.

```json
{
  "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",
      "Resource": "arn:aws:acm-pca:*:*:certificate-authority/*"
    },
    {
      "Effect": "Allow",
      "Action": [ "acm-pca:ListCertificate Authorities" ],
      "Resource": "*"
    }
  ]
}
```
리소스, 소유권 및 권한 정책 이해

• 사용자 – 최종 엔터티 인증서를 발급하고 해지할 수 있는 기능. 이 정책에는 관리 기능이 없으며 CA 인증서를 발급할 수 있는 기능이 없습니다. 권한은 PrivilegedUser 정책과 상호 배타적입니다.

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

• 감사자 – 읽기 전용 API 작업에 대한 액세스 및 CA 감사 보고서 생성 권한.

```json
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "acm-pca:CreateCertificateAuthorityAuditReport",
            "acm-pca:DescribeCertificateAuthority",
            "acm-pca:DescribeCertificateAuthorityAuditReport",
```
고객 관리형 정책

모범 사례로서 ACM Private CA를 포함하여 AWS와 상호 작용하는 데 AWS 계정 루트 사용자를 사용하지 않는 것이 바람직합니다. IAM 사용자, IAM 역할 또는 연합된 사용자를 생성할 때는 그 대신 AWS Identity and Access Management(IAM)를 사용하십시오. 관리자 그룹을 생성하고 여기에 자신을 추가합니다. 그런 다음 관리자로 로그인합니다. 필요에 따라 그룹에 사용자를 더 추가합니다.

사용자에게 할당할 수 있는 고객 관리형 IAM 정책을 생성하는 것도 좋은 방법입니다. 고객 관리형 정책은 AWS 계정에서 만들어서 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립적인 자격 증명 기반 정책입니다. 이러한 정책은 사용자가 지정한 ACM Private CA 작업만 수행하도록 제한합니다.

사용자는 아래의 고객 관리형 정책 예를 사용하여 CA 감사 보고서를 생성할 수 있습니다. 이 정책은 예제이며, 원하는 ACM Private CA 작업을 선택할 수 있습니다. 더 많은 예제는 인라인 정책 (p. 14)를 참조하십시오.

고객 관리형 정책을 생성하는 방법

1. AWS 관리자의 자격 증명을 사용하여 IAM 콘솔에 로그인합니다.
2. 콘솔의 탐색 창에서 정책을 선택합니다.
3. [Create policy]를 선택합니다.
4. [JSON] 탭을 선택합니다.
5. 다음 정책을 복사하여 편집기에 붙여 넣습니다.

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

7. 이름에 PcaListPolicy를 입력합니다.
8. (선택 사항) 설명을 입력합니다.
9. [Create policy]를 선택합니다.
관리자는 모든 IAM 사용자에게 정책을 연결하여 해당 사용자가 수행할 수 있는 ACM Private CA 작업을 제한할 수 있습니다. 권한 정책을 적용하는 방법은 IAM 사용 설명서의 IAM 사용자 권한 변경을 참조하십시오.

인라인 정책

인라인 정책은 자신이 생성 및 관리하며, 사용자, 그룹 또는 역할에 직접 포함되는 정책입니다. 다음 정책 예에서는 ACM Private CA 작업을 수행할 수 있는 권한을 할당하는 방법을 보여줍니다. 인라인 정책에 대한 일반적인 내용은 IAM 사용 설명서의 인라인 정책 작업을 참조하십시오. AWS Management 콘솔, AWS Command Line Interface(AWS CLI) 또는 IAM API를 사용하여 인라인 정책을 만들고 포함할 수 있습니다.

주제
• 사설 CA 나열 (p. 14)
• 사설 CA 인증서 검색 (p. 14)
• 사설 CA 인증서 가져오기 (p. 14)
• 사설 CA 인증서 삭제 (p. 15)
• ACM Private CA에 대한 읽기 전용 액세스 (p. 15)
• ACM Private CA에 대한 전체 액세스 (p. 15)
• 모든 AWS 리소스에 대한 관리자 액세스 (p. 16)

사설 CA 나열
다음 정책은 사용자가 계정에 모든 사설 CA를 나열하도록 허용합니다.

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

사설 CA 인증서 검색
다음 정책은 사용자가 사설 CA 인증서를 검색하도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "acm-pca:GetCertificateAuthorityCertificate",
    }
}
```

사설 CA 인증서 가져오기
다음 정책은 사용자가 사설 CA 인증서를 가져오도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "acm-pca:GetCertificate",
    }
}
```
인라인 정책

"Effect":"Allow",
"Action":"acm-pca:ImportCertificateAuthorityCertificate",
}
}

사설 CA 인증서 삭제

다음 정책은 사용자가 특정한 사설 CA 인증서를 삭제하도록 허용합니다.

{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":"acm-pca:DeleteCertificateAuthority",
   }
}

ACM Private CA에 대한 읽기 전용 액세스

다음 정책은 사용자가 사설 CA를 설명 및 나열하고 사설 CA 인증서 및 인증서 체인을 검색하도록 허용합니다.

{
   "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":"
   }
}

ACM Private CA에 대한 전체 액세스

다음 정책은 사용자가 ACM Private CA 작업을 수행할 수 있도록 허용합니다.

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

모든 AWS 리소스에 대한 관리자 액세스

다음 정책은 사용자가 모든 AWS 리소스에 대한 작업을 수행할 수 있도록 허용합니다.

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

AWS Certificate Manager Private Certificate Authority의 로깅 및 모니터링

모니터링은 AWS Certificate Manager Private Certificate Authority 및 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 역할을 합니다. 발생하는 다중 지점 실패를 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다.

다음 주제에서는 ACM Private CA에 사용할 수 있는 AWS 클라우드 모니터링 도구에 대해 설명합니다.

주제
- 지원되는 CloudWatch 지표 (p. 16)
- CloudWatch 이벤트 사용 (p. 17)
- 사용 CloudTrail (p. 21)

지원되는 CloudWatch 지표

Amazon CloudWatch는 AWS 리소스에 대한 모니터링 서비스입니다. CloudWatch를 사용하여 지표를 수집 및 추적하고, AWS 리소스의 변경 사항에 자동으로 대응할 수 있습니다. ACM Private CA는 다음과 같은 CloudWatch 지표를 지원합니다.

<table>
<thead>
<tr>
<th>지표</th>
<th>설명</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRLGenerated</td>
<td>인증서 취소 목록(CRL)이 생성되었습니다. 이 지표는 사설 CA에만 적용됩니다.</td>
</tr>
<tr>
<td>MisconfiguredCRLBucket</td>
<td>CRL에 대해 지정된 S3 버킷이 올바르게 구성되어 있지 않습니다. 버킷 정책을 확인합니다. 이 지표는 사설 CA에만 적용됩니다.</td>
</tr>
<tr>
<td>Time</td>
<td>이 지표는 인증서가 발급된 시간을 지정합니다. 이 지표는 IssueCertificate 작업에만 적용됩니다.</td>
</tr>
<tr>
<td>Success</td>
<td>인증서가 성공적으로 발급되었습니다. 이 지표는 IssueCertificate 작업에만 적용됩니다.</td>
</tr>
<tr>
<td>Failure</td>
<td>작업이 실패했습니다. 이 지표는 IssueCertificate 작업에만 적용됩니다.</td>
</tr>
</tbody>
</table>
CloudWatch 지표에 대한 자세한 내용은 다음 주제를 참조하십시오.

- Amazon CloudWatch 지표 사용
- Amazon CloudWatch 경보 만들기

## CloudWatch 이벤트 사용

Amazon CloudWatch Events를 사용하여 AWS 서비스를 자동화하고 애플리케이션 사양 문제나 리소스 변경 같은 시스템 이벤트에 자동으로 응답할 수 있습니다. AWS 서비스의 이벤트는 거의 실시간으로 CloudWatch 이벤트로 전송됩니다. 관심이 있는 이벤트와 이벤트 발생 시 실행할 자동 작업을 표시하도록 간단한 규칙을 작성할 수 있습니다. 자세한 내용은 이벤트에서 트리거되는 CloudWatch 이벤트 규칙 생성을 참조하십시오.

Amazon EventBridge를 사용하여 CloudWatch 이벤트를 작업으로 바꿀 수 있습니다. EventBridge에서는 이벤트를 사용하여 AWS Lambda 함수, AWS Batch 작업, Amazon SNS 주제 및 기타 여러 대상을 트리거할 수 있습니다. 자세한 내용은 Amazon EventBridge란 무엇입니까?를 참조하십시오.

### 사설 CA를 생성할 때 성공 또는 실패

이러한 이벤트는 CreateCertificateAuthority 작업에 의해 트리거됩니다.

**성공**

성공하면 작업은 새 CA의 ARN을 반환합니다.

```json
{
  "version": "0",
  "id": "93c0a8a5-3879-ee03-597f-5e2example18",
  "detail-type": "ACM Private CA Creation",
  "source": "aws.acm-pca",
  "account": "111111111111",
  "time": "2019-11-04T19:14:56Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:acm-pca:us-west-2:111111111111:certificate-authority/d543408e-0f41-4a3f-a0e0-84dEXAMPL51"
  ],
  "detail": {
    "result": "success"
  }
}
```

**결함**

실패 시 작업은 CA에 대한 ARN을 반환합니다. ARN을 사용하여 DescribeCertificateAuthority를 호출하여 CA의 상태를 확인할 수 있습니다.

```json
{
  "version": "0",
  "id": "93c0a8a5-3879-ee03-597f-5e2example18",
  "detail-type": "ACM Private CA Creation",
  "source": "aws.acm-pca",
  "account": "111111111111",
  "time": "2019-11-04T19:14:56Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:acm-pca:us-west-2:111111111111:certificate-authority/d543408e-0f41-4a3f-a0e0-84dEXAMPL51"
  ],
}
```
인증서 발급 시 성공 또는 실패

이러한 이벤트는 IssueCertificate 작업에 의해 트리거됩니다.

성공

성공하면 작업은 CA와 새 인증서의 ARN을 반환합니다.

```json
{
   "version": "0",
   "id": "dba9ac68-e917-adb7-b5fa-071examplea7",
   "detail-type": "ACM Private CA Certificate Issuance",
   "source": "aws.acm-pca",
   "account": "111111111111",
   "time": "2019-11-04T19:57:46Z",
   "region": "us-east-1",
   "resources": [
      "arn:aws:acm-pca:us-west-2:111111111111:certificate-authority/d543408e-0f41-4a3f-a0e0-84dEXAMPL51",
      "arn:aws:acm-pca:us-west-2:111111111111:certificate-authority/d543408e-0f41-4a3f-a0e0-84dEXAMPL51/certificate/b845c374b495cexample4c81cc4043"
   ],
   "detail": {
      "result": "success"
   }
}
```

결함

오류가 발생하면 작업은 인증서 ARN과 CA의 ARN을 반환합니다. 인증서 ARN을 사용하면 GetCertificate를 호출하여 실패 원인을 확인할 수 있습니다.

```json
{
   "version": "0",
   "id": "dba9ac68-e917-adb7-b5fa-071examplea7",
   "detail-type": "ACM Private CA Certificate Issuance",
   "source": "aws.acm-pca",
   "account": "111111111111",
   "time": "2019-11-04T19:57:46Z",
   "region": "us-east-1",
   "resources": [
      "arn:aws:acm-pca:us-west-2:111111111111:certificate-authority/d543408e-0f41-4a3f-a0e0-84dEXAMPL51",
      "arn:aws:acm-pca:us-west-2:111111111111:certificate-authority/d543408e-0f41-4a3f-a0e0-84dEXAMPL51/certificate/b845c374b495cexample4c81cc4043"
   ],
   "detail": {
      "result": "failure"
   }
}
```

인증서 해지 시 성공

이 이벤트는 RevokeCertificate 작업에 의해 트리거됩니다.
해지가 실패하거나 인증서가 이미 해지된 경우에는 이벤트가 전송되지 않습니다.

Success

성공하면 작업은 CA 및 해지된 인증서의 ARN을 반환합니다.

```json
{
  "version":"0",
  "id":"247b9dcb-1f62-b23a-2195-790example7b",
  "detail-type":"ACM Private CA Certificate Revocation",
  "source":"aws.acm-pca",
  "account":"
111111111111",
  "time":"2019-11-05T20:25:19Z",
  "region":"us-east-1",
  "resources":[
    "arn:aws:acm-pca:us-east-1:111111111111:certificate-authority/
d87e9a0a-75cb-44ba-bf83-44cEXAMPLE92",
    "arn:aws:acm-pca:us-east-1:111111111111:certificate-authority/
d87e9a0a-75cb-44ba-bf83-44cEXAMPLE92/certificate/d2196bfeef09b5b2088b205EXAMPLEd4"
  ],
  "detail":{
    "result":"success"
  }
}
```

CRL 생성 시 성공 또는 실패

이러한 이벤트는 RevokeCertificate 작업에 의해 트리거되며, 이로 인해 인증서 해지 목록(CRL)이 생성됩니다.

성공

성공하면 이 작업은 CRL과 연결된 CA의 ARN을 반환합니다.

```json
{
  "version":"0",
  "id":"fefaffcc-9579-8e7b-0565-f11examplead",
  "detail-type":"ACM Private CA CRL Generation",
  "source":"aws.acm-pca",
  "account":"
111111111111",
  "time":"2019-11-04T21:07:08Z",
  "region":"us-east-1",
  "resources":[
    "arn:aws:acm-pca:ap-southeast-1:111111111111:certificate-authority/b58e1229-
f656-453e-bd11-109EXAMPLE1a"
  ],
  "detail":{
    "result":"success"
  }
}
```

실패 1 - 권한 오류로 인해 CRL이 Amazon S3에 저장되지 않았을 수 있음

이 오류가 발생하면 Amazon S3 버킷 권한을 확인합니다.
실패 2 - 내부 오류로 인해 CRL이 Amazon S3에 저장되지 않았을 수 있음
이 오류가 발생하면 작업을 다시 시도하십시오.

실패 3 - ACM PCA에서 CRL을 생성하지 못함
이 오류를 해결하려면 CloudWatch 지표를 확인하십시오.

CA 감사 보고서 생성 시 성공 또는 실패
이러한 이벤트는 CreateCertificateAuthorityAuditReport 작업에 의해 트리거됩니다.
성공

성공하면 작업은 CA의 ARN과 감사 보고서의 ID를 반환합니다.

```
{
    "version":"0",
    "id":"f4f561f4-c045-e0a3-cb06-bf7exampled8",
    "detail-type":"ACM Private CA Audit Report Generation",
    "source":"aws.acm-pca",
    "account":"111111111111",
    "time":"2019-11-04T21:54:20Z",
    "region":"us-east-1",
    "resources":[
        "arn:aws:acm-pca:us-west-2:111111111111:certificate-authority/d543408e-0f41-4a3f-a0e0-84dEXAMPLB51",
        "5903194d-0df7-4733-a8a0-cefexampleb9"
    ],
    "detail":{
        "result":"success"
    }
}
```

결함

ACM PCA에서 Amazon S3 버킷에 대한 PUT 권한이 없거나 버킷에서 암호화가 활성화된 경우, 또는 기타 이유로 감사 보고서가 실패할 수 있습니다.

```
{
    "version":"0",
    "id":"f4f561f4-c045-e0a3-cb06-bf7exampled8",
    "detail-type":"ACM Private CA Audit Report Generation",
    "source":"aws.acm-pca",
    "account":"111111111111",
    "time":"2019-11-04T21:54:20Z",
    "region":"us-east-1",
    "resources":[
        "arn:aws:acm-pca:us-west-2:111111111111:certificate-authority/d543408e-0f41-4a3f-a0e0-84dEXAMPLB51",
        "5903194d-0df7-4733-a8a0-cefexampleb9"
    ],
    "detail":{
        "result":"failure"
    }
}
```

사용 CloudTrail

AWS CloudTrail을 사용하여 AWS Certificate Manager Private Certificate Authority에서 생성되는 API 호출을 기록할 수 있습니다. 자세한 내용은 다음 항목을 참조하십시오.

주제

- CA 생성 (p. 22)
- 감사 보고서 생성 (p. 23)
- CA 삭제 (p. 23)
- 인증 기관 복원 (p. 24)
- CA 설명 (p. 24)
- CA 인증서 검색 (p. 25)
- 인증 기관 서명 요청 검색 (p. 25)
CA 생성

다음 CloudTrail 예제에서는 CreateCertificateAuthority 작업을 호출한 결과를 보여 줍니다.

```json
{
    eventVersion: "1.05",
    userIdentity: {
        type: "IAMUser",
        principalId: "account",
        arn: "arn:aws:iam::account:user/name",
        accountId: "account",
        accessKeyId: "Key_ID"
    },
    eventSource: "acm-pca.amazonaws.com",
    eventName: "CreateCertificateAuthority",
    awsRegion: "us-east-1",
    sourceIPAddress: "xx.xx.xx.xx",
    userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
    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:region:account:certificate-authority/ac507c2e-19c8-4258-b74e-351c2b791fe1"
    },
    requestID: "332b0c69-8779-4625-bdef-2e954a418265",
    eventID: "cd27c874-ae6e-4585-9c1b-5abf5537ec39"
}
```
감사 보고서 생성

다음 CloudTrail 예제에서는 CreateCertificateAuthorityAuditReport 작업을 호출한 결과를 보여 줍니다.

```json
{
    eventVersion: "1.05",
    userIdentity: {
        type: "IAMUser",
        principalId: "account",
        arn: "arn:aws:iam::account:account:user/name",
       accountId: "account",
        accessKeyId: "Key_ID"
    },
    eventTime: "2018-01-26T21:56:00Z",
    eventSource: "acm-pca.amazonaws.com",
    eventName: "CreateCertificateAuthorityAuditReport",
    awsRegion: "us-east-1",
    sourceIPAddress: "xx.xx.xx.xx",
    userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
    requestParameters: {
        certificateAuthorityArn: "arn:aws:acm-pca:region:account:certificate-authority/ac5a7c2e-19c8-4258-b74e-351c2b791fe1",
        s3BucketName: "your-bucket-name",
        auditReportResponseFormat: "JSON"
    },
    responseElements: {
        auditReportId: "2a7d28e7-a835-40a6-b19f-371186c62346",
        s3Key: "audit-report/ac5a7c2e-19c8-4258-b74e-351c2b791fe1/2a7d28e7-a835-40a6-b19f-371186c62346.json"
    },
    requestID: "3b56d220-1660-4941-8160-b54dcc70592d",
    eventID: "ea95f673-e7be-411d-bb54-calab884baaf",
    eventType: "AwsApiCall",
    recipientAccountid: "account"
}
```

CA 삭제

다음 CloudTrail 예제에서는 DeleteCertificateAuthority 작업에 대한 호출의 결과를 보여 줍니다. 이 예제에서 인증 기관이 ACTIVE 상태에 있지 않았기 때문에 삭제할 수 없습니다.

```json
{
    eventVersion: "1.05",
    userIdentity: {
        type: "IAMUser",
        principalId: "account",
        arn: "arn:aws:iam::account:account:user/name",
        accountId: "account",
        accessKeyId: "Key_ID"
    },
    eventTime: "2018-01-26T22:01:11Z",
    eventSource: "acm-pca.amazonaws.com",
    eventName: "DeleteCertificateAuthority",
    awsRegion: "us-east-1",
    sourceIPAddress: "xx.xx.xx.xx",
    requestParameters: {
        certificateAuthorityArn: "arn:aws:acm-pca:region:account:certificate-authority/ac5a7c2e-19c8-4258-b74e-351c2b791fe1",
        s3BucketName: "your-bucket-name",
        auditReportResponseFormat: "JSON"
    },
    responseElements: {
        auditReportId: "2a7d28e7-a835-40a6-b19f-371186c62346",
        s3Key: "audit-report/ac5a7c2e-19c8-4258-b74e-351c2b791fe1/2a7d28e7-a835-40a6-b19f-371186c62346.json"
    },
    requestID: "3b56d220-1660-4941-8160-b54dcc70592d",
    eventID: "ea95f673-e7be-411d-bb54-calab884baaf",
    eventType: "AwsApiCall",
    recipientAccountid: "account"
}
```
인증 기관 복원

다음 CloudTrail 예제에서는 RestoreCertificateAuthority 작업에 대한 호출의 결과를 보여 줍니다. 이 예제에서 인증 기관이 DELETED 상태에 있으므로 복원할 수 없습니다.

```json
{
    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: "us-east-1",
    sourceIPAddress: "xx.xx.xx.xx",
    userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
    errorCode: "InvalidStateException",
    errorMessage: "The certificate authority is not in a valid state for restoration.",
    requestParameters: {
        certificateAuthorityArn: "arn:aws:acm-pca:region:account:certificate-authority/09517d62-4f11-4bf8-a2c9-9e863792b675"
    },
    responseElements: null,
    requestID: "dae3e14f-62f6-42f3-acf4-630c47a09ee4",
    eventID: "c40abfac-53f7-420a-9b55-c3f2f2139de8",
    eventType: "AwsApiCall",
    recipientAccountId: "account"
}
```

CA 설명

다음 CloudTrail 예제에서는 DescribeCertificateAuthority 작업을 호출한 결과를 보여 줍니다.

```json
{
    eventVersion: "1.05",
    userIdentity: {
        type: "IAMUser",
        principalId: "account",
        arn: "arn:aws:iam::account:user/name",
        accountId: "account"
    },
    eventTime: "2018-01-26T22:01:11Z",
    eventSource: "acm-pca.amazonaws.com",
    eventName: "DescribeCertificateAuthority",
    awsRegion: "us-east-1",
    sourceIPAddress: "xx.xx.xx.xx",
    userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
    errorCode: "InvalidStateException",
    errorMessage: "The certificate authority is not in a valid state for restoration.",
    requestParameters: {
        certificateAuthorityArn: "arn:aws:acm-pca:region:account:certificate-authority/09517d62-4f11-4bf8-a2c9-9e863792b675"
    },
    responseElements: null,
    requestID: "dae3e14f-62f6-42f3-acf4-630c47a09ee4",
    eventID: "c40abfac-53f7-420a-9b55-c3f2f2139de8",
    eventType: "AwsApiCall",
    recipientAccountId: "account"
}
```
CA 인증서 검색
다음 CloudTrail 예제에서는 GetCertificateAuthorityCertificate 작업을 호출한 결과를 보여 줍니다.

```
{
  eventVersion: "1.05",
  userIdentity: {
    type: "IAMUser",
    principalId: "account",
    arn: "arn:aws:iam::account:account:user/name",
    accountId: "account",
    accessKeyId: "Key_ID"
  },
  eventTime: "2018-01-26T22:03:52Z",
  eventSource: "acm-pca.amazonaws.com",
  eventName: "GetCertificateAuthorityCertificate",
  awsRegion: "us-east-1",
  sourceIPAddress: "xx.xx.xx.xx",
  userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
  requestParameters: {
    certificateAuthorityArn: "arn:aws:acm-pca:region:account:certificate-authority/ac507c2e-19c8-4258-b74e-351c2b791fe1"
  },
  responseElements: null,
  requestID: "94cee046-bf52-4a69-b95c-eae662818410",
  eventID: "7dd83d274-8c5f-4b9a-b9b6-371b53771ce9",
  eventType: "AwsApiCall",
  recipientAccountId: "account"
}
```

인증 기관 서명 요청 검색
다음 CloudTrail 예제에서는 GetCertificateAuthorityCsr 작업을 호출한 결과를 보여 줍니다.

```
{
  eventVersion: "1.05",
  userIdentity: {
    type: "IAMUser",
    principalId: "account",
    arn: "arn:aws:iam::account:account:user/name",
    accountId: "account",
    accessKeyId: "Key_ID"
  },
  eventTime: "2018-01-26T22:03:52Z",
  eventSource: "acm-pca.amazonaws.com",
  eventName: "GetCertificateAuthorityCsr",
  awsRegion: "us-east-1",
  sourceIPAddress: "xx.xx.xx.xx",
  userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
  requestParameters: {
    certificateAuthorityArn: "arn:aws:acm-pca:region:account:certificate-authority/ac507c2e-19c8-4258-b74e-351c2b791fe1"
  },
  responseElements: null,
  requestID: "94cee046-bf52-4a69-b95c-eae662818410",
  eventID: "7dd83d274-8c5f-4b9a-b9b6-371b53771ce9",
  eventType: "AwsApiCall",
  recipientAccountId: "account"
}
```
인증서 검색
다음 CloudTrail 예제에서는 GetCertificate 작업을 호출한 결과를 보여 줍니다.

```
{
  eventVersion: "1.05",
  userIdentity: {
    type: "IAMUser",
    principalId: "account",
    arn: "arn:aws:iam::account:user/name",
    accountId: "account",
    accessKeyId: "Key_ID"
  },
  eventSource: "acm-pca.amazonaws.com",
  eventName: "GetCertificate",
  awsRegion: "us-east-1",
  sourceIPAddress: "xx.xx.xx.xx",
  userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
  requestParameters: {
    certificateArn: "arn:aws:acm-pca:region:account:certificate/ac507c2e-19c8-4258-b74e-351c2b791fe1/certificate/6707447683a9b7f4055627f7d55cebce"
  },
  responseElements: null,
  requestID: "ec0681a9-6202-496e-985a-adba939311e4",
  eventID: "89be1a2a-3340-4d74-a633-5582a2fba1d3",
  eventType: "AwsApiCall",
  recipientAccountId: "account"
}
```

CA 인증서 가져오기
다음 CloudTrail 예제에서는 ImportCertificateAuthorityCertificate 작업을 호출한 결과를 보여 줍니다.

```
{
  eventVersion: "1.05",
  userIdentity: {
    type: "IAMUser",
    principalId: "account",
    arn: "arn:aws:iam::account:user/name",
    accountId: "account",
    accessKeyId: "Key_ID"
  },
  eventTime: "2018-01-26T22:54Z",
  eventSource: "acm-pca.amazonaws.com",
  eventName: "GetCertificate",
  awsRegion: "us-east-1",
  sourceIPAddress: "xx.xx.xx.xx",
  userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
  requestParameters: {
    certificateArn: "arn:aws:acm-pca:region:account:certificate/ac507c2e-19c8-4258-b74e-351c2b791fe1/certificate/6707447683a9b7f4055627f7d55cebce"
  },
  responseElements: null,
  requestID: "ec0681a9-6202-496e-985a-adba939311e4",
  eventID: "89be1a2a-3340-4d74-a633-5582a2fba1d3",
  eventType: "AwsApiCall",
  recipientAccountId: "account"
}
```
인증서 발급
다음 CloudTrail 예제에서는 IssueCertificate 작업을 호출한 결과를 보여 줍니다.

```json
{
    eventVersion: "1.05",
    userIdentity: {
        type: "IAMUser",
        principalId: "account",
        arn: "arn:aws:iam::account:user/name",
        accountId: "account",
        accessKeyId: "Key_ID"
    },
    eventSource: "acm-pca.amazonaws.com",
    eventName: "ImportCertificateAuthorityCertificate",
    awsRegion: "us-east-1",
    sourceIPAddress: "xx.xx.xx.xx",
    userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
    requestParameters: {
        certificateAuthorityArn: "arn:aws:acm-pca:region:account:certificate-authority/ac5a7c2e-19c8-4258-b74e-351c2b791fe1",
        certificate: {
            hb: [45, 45, ...
            , offset: 0,
            isReadOnly: false,
            bigEndian: true,
            nativeByteOrder: false,
            mark: -1,
            position: 1257,
            limit: 1257,
            capacity: 1257,
            address: 0
        },
        certificateChain: {
            hb: [45, 45, ...
            , offset: 0,
            isReadOnly: false,
            bigEndian: true,
            nativeByteOrder: false,
            mark: -1,
            position: 1139,
            limit: 1139,
            capacity: 1139,
            address: 0
        }
    },
    responseElements: null,
    requestId: "36bbba0c-2d08-4995-99fc-964926103841",
    eventID: "17a38b26-49c1-41d2-8d15-f9362eeaaaa0",
    eventType: "AwsApiCall",
    recipientAccountId: "account"
}
```
CA 나열

다음 CloudTrail 예제에서는 ListCertificateAuthorities 작업을 호출한 결과를 보여 줍니다.

```json
{
  eventVersion: "1.05",
  userIdentity: {
    type: "IAMUser",
    principalId: "account",
    accountId: "account",
    accessKeyId: "Key_ID"
  },
  eventSource: "acm-pca.amazonaws.com",
  eventName: "ListCertificateAuthorities",
  awsRegion: "us-east-1",
  sourceIPAddress: "xx.xx.xx.xx",
  userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
  requestParameters: {
    certificateArn: "arn:aws:acm-pca:region:account:certificate-authority/ac5a7c2e-19c8-4258-b74e-351c2b791fe1/",
    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: "DAHIS"
  },
  idempotencyToken: "1234"
}
```
태그 나열

다음 CloudTrail 예제에서는 ListTags 작업을 호출한 결과를 보여 줍니다.

```json
{
  "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": "us-east-1",
  "sourceIPAddress": "xx.xx.xx.xx",
  "userAgent": "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
  "requestParameters": {
    "certificateAuthorityArn": "arn:aws:acm-pca:us-east-1:account:certificate-authority/ac507c2e-19c0-4258-b74e-351c2b791fe1"
  },
  "responseElements": {
    "tags": [
      {
        "key": "Admin",
        "value": "Alice"
      },
      {
        "key": "User",
        "value": "Bob"
      }
    ],
  "requestID": "72819d8d-c6bc-4921-a944-95bb99ed911",
  "eventID": "a349328f-e3e0-48ee-abc9-00526768080a",
  "eventType": "AwsApiCall",
  "recipientAccountId": "account"
}
```

인증서 취소

다음 CloudTrail 예제에서는 RevokeCertificate 작업을 호출한 결과를 보여 줍니다.

```json
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "account",
    "arn": "arn:aws:iam::account:user/name",
    "accountId": "account",
    "accessKeyId": "Key_ID"
  },
  "eventTime": "2018-01-26T22:35:03Z",
  "eventSource": "acm-pca.amazonaws.com",
  "eventSourceArn": "arn:aws:acm-pca:us-east-1:account:certificate-authority/ac507c2e-19c0-4258-b74e-351c2b791fe1"
}
```
사설 CA 태그 지정

다음 CloudTrail 예제에서는 TagCertificateAuthority 작업을 호출한 결과를 보여 줍니다.

```json
{
  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: "us-east-1",
  sourceIPAddress: "xx.xx.xx.xx",
  userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
  requestParameters: {
    certificateAuthorityArn: "arn:aws:acm-pca:us-east-1:account:certificate-authority/ac5a7c2e-19c8-4258-b74e-351c2b791fe1",
    revocationReason: "KEY_COMPROMISE"
  },
  responseElements: null,
  requestID: "816df59d-5022-47af-aa58-173e5c73c20a",
  eventID: "8c99648b-3d77-483f-b56b-5aaa85cb6cde",
  eventType: "AwsApiCall",
  recipientAccountId: "account"
}
```

사설 CA에서 태그 제거

다음 CloudTrail 예제에서는 UntagCertificateAuthority 작업을 호출한 결과를 보여 줍니다.

```json
{
  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: "us-east-1",
  sourceIPAddress: "xx.xx.xx.xx",
  userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 botocore/1.8.32",
  requestParameters: {
    certificateAuthorityArn: "arn:aws:acm-pca:us-east-1:account:certificate-authority/ac5a7c2e-19c8-4258-b74e-351c2b791fe1",
    revocationReason: "KEY_COMPROMISE"
  },
  responseElements: null,
  requestID: "816df59d-5022-47af-aa58-173e5c73c20a",
  eventID: "8c99648b-3d77-483f-b56b-5aaa85cb6cde",
  eventType: "AwsApiCall",
  recipientAccountId: "account"
}
```
AWS Certificate Manager Private
Certificate Authority 사용 설명서

사용 CloudTrail

CA 업데이트
다음 CloudTrail 예제에서는 UpdateCertificateAuthority 작업을 호출한 결과를 보여 줍니다.

```json
{
    eventVersion: "1.05",
    userIdentity: {
        type: "IAMUser",
        principalId: "account",
        arn: "arn:aws:iam::account:username",
        accountId: "account",
        accessKeyId: "Key_ID"
    },
    eventSource: "acm-pca.amazonaws.com",
    eventName: "UpdateCertificateAuthority",
    awsRegion: "us-east-1",
    sourceIPAddress: "xx.xx.xx.xx",
    userAgent: "aws-cli/1.14.28 Python/2.7.9 Windows/8 boto core/1.8.32",
    requestParameters: {
        revocationConfiguration: {
            crlConfiguration: {
                enabled: true,
                expirationInDays: 3650,
                customCname: "your-custom-name",
            s3BucketName: "your-bucket-name"
            }
        },
        status: "DISABLED"
    },
    responseElements: null,
    requestID: "24f849f9-9966-4f13-8ff6-3e2e84b327fc",
    eventID: "16c78ea0-e3d7-4817-9bb0-0b997789678f",
    eventType: "AwsApiCall",
    recipientAccountId: "account"
}
```
AWS Certificate Manager Private Certificate Authority의 규정 준수 확인


특정 규정 준수 프로그램의 범위 내에 있는 AWS 서비스 목록은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하십시오. 일반적인 내용은 AWS 규정 준수 프로그램을 참조하십시오.

AWS Artifact를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS 아티팩트에서 보고서 다운로드를 참조하십시오.

ACM Private CA 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS에서는 규정 준수에 도움이 되도록 다음과 같은 리소스를 제공합니다.

• Amazon S3 버킷을 암호화해야 하는 조직을 위해 다음 주제에서는 ACM Private CA 자산을 수용하도록 암호화를 구성하는 방법에 대해 설명합니다.
• 감사 보고서 암호화
• CRL 암호화
• 보안 및 규정 준수 빠른 시작 안내서 — 이 배포 가이드에서는 아키텍처 고려 사항에 대해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
• HIPAA 보안 및 규정 준수 기술 백서 설계 — 이 백서는 기업에서 AWS를 사용하여 HIPAA를 준수하는 애플리케이션을 생성하는 방법을 설명합니다.
• AWS 규정 준수 리소스 — 이 워크북 및 안내서 모음은 귀사가 속한 업계 및 국가에 적용될 수 있습니다.
• AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 — 이 AWS Config 서비스는 리소스 구성이 내부 사례, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
• AWS Security Hub — 이 AWS 서비스는 보안 업계 표준 및 모범 사례를 준수하는지 여부를 확인하는 데 도움이 되도록 AWS 내 보안 상태에 대한 포괄적인 보기를 제공합니다.

사설 CA에 대한 감사 보고서 생성

사설 CA가 발급 및 취소된 모든 인증서의 목록을 표시하는 감사 보고서를 생성할 수 있습니다. 이 보고서는 입력 시 지정한 신규 또는 기존 S3 버킷에 저장됩니다.

감사 보고서에 암호화 보호를 추가하는 방법에 대한 자세한 내용은 감사 보고서 암호화 (p. 34) 단원을 참조하십시오.

감사 보고서 파일의 경로와 파일 이름은 다음과 같습니다(CA_ID은 발급 CA의 고유 식별자이며UUID는 감사 보고서의 고유 식별자).

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

30분마다 새 보고서를 생성하고 버킷에서 이를 다운로드할 수 있습니다. 다음 예제에서는 CSV로 구분된 보고서를 보여 줍니다.

```
awsAccountId,requestedByServicePrincipal,certificateArn,serial,subject,notBefore,notAfter,issuedAt,revokedAt,revocationReason,templateArn
```

버전 latest
32
다음 예제에서는 JSON 형식의 보고서를 보여줍니다.

```json
[{
  "awsAccountId": "123456789012",
  "subject": "2.5.4.5=#012345678901,2.5.4.44=#0a1b3c4d,2.5.4.65=#0a1b3c4d5e6f,2.5.4.43=#0a1b3c4d5e,2.5.4.42=#0123456789abcdef0123456789abcdef0123,2.5.4.4=#0123456789abcdef01234567,2.5.4.12=#0a1b3c4d5e,2.5.4.46=#0123456789ab,CN=www.example1.com,OU=Sales,O=Example Company,L=Seattle,ST=Washington,C=US",
  "notBefore": "2020-02-26T18:39:57+0000",
  "notAfter": "2021-02-26T19:39:57+0000",
  "issuedAt": "2020-02-26T18:39:58+0000",
  "revokedAt": "2020-02-26T19:59:58+0000",
  "revocationReason": "UNSPECIFIED",
  "templateArn": "arn:aws:acm-pca:::template/EndEntityCertificate/V1"
},
{
  "awsAccountId": "123456789012",
  "requestedByServicePrincipal": "acm.amazonaws.com",
  "subject": "2.5.4.5=#012345678901,2.5.4.44=#0a1b3c4d,2.5.4.65=#0a1b3c4d5e6f,2.5.4.43=#0a1b3c4d5e,2.5.4.42=#0123456789abcdef0123456789abcdef0123,2.5.4.4=#0123456789abcdef01234567,2.5.4.12=#0a1b3c4d5e,2.5.4.46=#0123456789ab,CN=www.example1.com,OU=Sales,O=Example Company,L=Seattle,ST=Washington,C=US",
  "notBefore": "2020-01-22T20:10:49+0000",
  "notAfter": "2021-01-17T21:10:49+0000",
  "issuedAt": "2020-01-22T20:10:49+0000",
  "templateArn": "arn:aws:acm-pca:::template/EndEntityCertificate/V1"
}]
```

콘솔이나 AWS CLI에서 보고서를 생성할 수 있습니다.

감사 보고서 생성 방법(콘솔)

2. 사설 CA를 선택합니다.
3. 목록에서 사설 CA를 선택합니다.
4. 작업 메뉴에서 감사 보고서 생성을 선택합니다.
5. S3 버킷 새로 생성에서 예를 선택하고 고유한 버킷 이름을 입력하거나 아니요를 선택하고 목록에서 기존 버킷을 선택합니다.

예를 선택한 경우 ACM Private CA는 필요한 정책을 생성하여 버킷에 연결합니다. 아니요를 선택한 경우에 다음 정책을 버킷에 연결해야만 감사 보고서를 생성할 수 있습니다. 지침은 S3 버킷 정책을 추가해야 할까요?

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

```
감사 보고서 생성

감사 보고서 생성하는 방법(AWS CLI)

create-certificate-authority-audit-report 명령을 사용하여 감사 보고서를 생성합니다. 다음 정책을 버킷에 연결해야 합니다.

```
{
    "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:::bucket-name/*",
                "arn:aws:s3:::bucket-name"
            ]
        }
    ]
}
```

aws acm-pca create-certificate-authority-audit-report
   --certificate-authority-arn arn:aws:acm-pca:region:account:
   certificate-authority/12345678-1234-1234-1234-123456789012
   --s3-bucket-name >your-bucket-name
   --audit-report-response-format JSON

감사 보고서 암호화

선택에 따라 감사 보고서가 포함된 Amazon S3 버킷에 암호화를 구성할 수 있습니다. ACM Private CA는 S3의 자산에 대해 두 가지 암호화 모드를 지원합니다.

- Amazon S3 관리형 AES-256 키를 사용한 자동 서버 측 암호화.
- 사양에 맞게 구성된 CMK(고객 마스터 키)와 AWS Key Management Service를 사용한 고객 관리 암호화.
Note

ACM Private CA는 S3에 의해 자동으로 생성된 기본 CMK 사용을 지원하지 않습니다.

다음 절차에서는 각 암호화 옵션을 설정하는 방법에 대해 설명합니다.

자동 암호화를 구성하면

S3 서버 측 암호화를 활성화하려면 다음 단계를 수행하십시오.

2. 버킷 테이블에서 ACM Private CA 자산을 보존할 버킷을 선택합니다.
3. 버킷의 페이지에서 속성 탭을 선택합니다.
4. 기본 암호화 카드를 선택합니다.
5. AES-256을 선택합니다.
6. 필요에 따라 버킷 권한 정책을 확인한 다음 저장을 선택합니다.

사용자 지정 암호화를 구성하면

사용자 지정 CMK를 사용하여 암호화를 활성화하려면 다음 단계를 수행하십시오.

1. (선택 사항) AWS KMS CMK가 아직 없는 경우 다음 AWS CLI create-key 명령을 사용하여 생성합니다.

   ```bash
   aws kms create-key
   ```

   이 명령의 출력 화면에는 CMK의 키 ID와 Amazon 리소스 이름(ARN)이 포함됩니다. 다음은 예제 출력입니다.

   ```json
   {
     "KeyMetadata": {
       "KeyId": "6f815f63-e628-448c-8251-e40cb0d29f59",
       "Description": "",
       "Enabled": true,
       "KeyUsage": "ENCRYPT_DECRYPT",
       "KeyState": "Enabled",
       "CreationDate": 1478910250.94,
       "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-e40cb0d29f59",
       "AWSAccountId": "123456789012"
     }
   }
   ```

   2. 모든 AWS KMS CMK는 기본적으로 비공개입니다. 따라서 리소스 소유자가 이를 사용해서 데이터를 암호화 및 해독할 수 있습니다. 그러나 리소스 소유자가 원한다면 다른 사용자 및 리소스에게 CMK에 대한 액세스 권한을 부여할 수 있습니다. 다음 단계를 사용하여 ACM Private CA 서비스 보안 주체에 CMK를 사용할 수 있는 권한을 부여합니다. 서비스 보안 주체는 CMK가 저장된 위치와 동일한 리전에 있어야 합니다.

   a. 먼저, 아래 get-key-policy 명령을 사용하여 CMK에 대한 기본 정책을 policy.json로 저장합니다.

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

   b. 텍스트 편집기에서 policy.json 파일을 열고 다음 설명을 추가합니다.

   ```json
   {
     "Sid":"Allow ACM-PCA use of the key",
   }
   ```
AWS Certificate Manager Private Certificate Authority의 인프라 보안

관리형 서비스인 ACM Private CA는 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다.

AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 ACM Private CA에 액세스합니다. 클라이언트가 TLS(전송 계층 보안) 1.0 이상을 지원해야 합니다. TLS 1.2 이상을 권장합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 보안 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

주제
• ACM Private CA VPC 엔드포인트(AWS PrivateLink) (p. 36)

ACM Private CA VPC 엔드포인트(AWS PrivateLink)

인터페이스 VPC 엔드포인트를 구성하여 VPC와 ACM Private CA 간에 프라이빗 연결을 생성할 수 있습니다. 인터페이스 엔드포인트는 ACM Private CA API 작업에 대한 프라이빗 액세스를 위한 기술인 AWS PrivateLink를 통해 구동됩니다. AWS PrivateLink는 Amazon 네트워크를 통해 VPC와 ACM Private CA간에 모든 네트워크 트래픽을 라우팅하므로 개방형 인터넷에 트래픽이 노출되지 않습니다. 각 VPC 엔드포인트는 하나 이상의 탄력적 네트워크 인터페이스(ENI) 및 VPC 서브넷의 프라이빗 IP 주소로 표현됩니다.

인터페이스 VPC 엔드포인트는 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 VPC를 ACM Private CA에 직접 연결합니다. VPC에 있는 인스턴스는 퍼블릭 IP 주소가 있어도 ACM Private CA API에 통신할 수 있습니다.
VPC를 통해 ACM Private CA를 사용하려면 VPC 내부에 있는 인스턴스에서 연결해야 합니다. 아니라면 AWS Virtual Private Network(AWS VPN) 또는 AWS Direct Connect를 사용하여 프라이빗 네트워크에 VPC에 연결할 수 있습니다. AWS VPN에 대한 자세한 내용은 Amazon VPC 사용 설명서의 'VPN 연결'을 참조하십시오.

AWS Direct Connect에 대한 자세한 내용은 AWS Direct Connect 사용 설명서의 연결 생성을 참조하십시오.

ACM Private CA에서 반드시 AWS PrivateLink를 사용할 필요는 없지만 추가 보안 계층으로 사용하는 것이 좋습니다. AWS PrivateLink 및 VPC 엔드포인트에 대한 자세한 내용은 AWS PrivateLink를 통해 서비스 액세스를 참조하십시오.

ACM Private CA용 인터페이스 VPC 엔드포인트 고려 사항

ACM Private CA용 인터페이스 VPC 엔드포인트를 설정하기 전에 다음 사항을 고려하십시오.

- ACM Private CA는 us-east-1, us-west-2, ca-central-1 및 ap-northeast-2 리전의 일부 가용 영역에서 VPC 엔드포인트를 지원하지 않습니다. 관리 콘솔에서 VPC 엔드포인트를 생성할 때 지원되지 않는 가용 영역은 "이 가용 영역에서 지원되지 않는 서비스"로 표시가 됩니다.
- VPC 엔드포인트는 교차 리전 요청을 지원하지 않습니다. API 호출을 ACM Private CA로 발행할 계획인 동일 리전에서 엔드포인트를 생성해야 합니다.
- VPC 엔드포인트는 Amazon Route 53를 통해 Amazon 제공 DNS만 지원합니다. 자신의 DNS를 사용하는 경우에는 조건적인 DNS 전송을 사용할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 DHCP 옵션 세트 단원을 참조하십시오.
- VPC 엔드포인트에 연결된 보안 그룹은 VPC의 프라이빗 서브넷에서 443 포트로 들어오는 연결을 허용해야 합니다.
- AWS Certificate Manager는 VPC 엔드포인트를 지원하지 않습니다.
- FIPS 엔드포인트(및 해당 리전)는 VPC 엔드포인트를 지원하지 않습니다.

ACM Private CA용 VPC 엔드포인트 생성


엔드포인트에서 프라이빗 DNS 호스트 이름을 활성화하면 기본 ACM Private CA 엔드포인트가 VPC 엔드포인트로 확인됩니다. 기본 서비스 엔드포인트의 전체 목록은 서비스 엔드포인트 및 할당량을 참조하십시오.

프라이빗 DNS 호스트 이름을 활성화하지 않은 경우, Amazon VPC는 다음 형식으로 사용할 수 있는 DNS 엔드포인트를 제공합니다.

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

**Note**

region 값은 ACM Private CA에서 지원되는 AWS 리전의 리전 식별자를 나타냅니다(예: 미국 동부 (오하이오) 리전에 대한 us-east-2). ACM Private CA 목록은 AWS Certificate Manager 사용 인증 기관 엔드포인트 및 할당량을 참조하십시오.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하십시오.

ACM Private CA에 대한 VPC 엔드포인트 정책 생성

ACM Private CA에 대한 Amazon VPC 엔드포인트 정책을 생성하여 다음을 지정할 수 있습니다.
AWS Certificate Manager Private
Certificate Authority 사용 설명서
ACM Private CA 모범 사례

- 작업을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업.
- 작업을 수행할 있는 리소스.

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하시오.

예제 - ACM Private CA 작업에 대한 VPC 엔드포인트 정책


```json
{
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": ["acm-pca:IssueCertificate"],
      "Condition": {
        "StringEquals": {
          "acm-pca:TemplateArn": "arn:aws:acm-pca:::template/EndEntityCertificate/V1"
        }
      }
    },
    {
      "Principal": "*",
      "Effect": "Allow",
    }
  ]
}
```

ACM Private CA 모범 사례

모범 사례는 ACM Private CA를 효과적으로 사용하는 데 도움이 되는 권장 사항입니다. 다음 모범 사례는 현재 AWS Certificate Manager 및 ACM Private CA 고객의 실제 경험을 기반으로 합니다.
CA 구조 및 정책 문서화

AWS에서는 CA 운영에 대한 모든 정책과 관행을 문서화할 것을 권장합니다. 여기에는 다음이 포함될 수 있습니다.

- CA 구조에 대한 의사 결정을 위한 추론
- CA 및 CA 관계를 보여주는 다이어그램
- CA 유 효 기간에 대한 정책
- CA 승계 계획
- 경로 길이에 대한 정책
- 텔레그램 액세스
- 관리 제어 구조에 대한 설명
- 보안

이 정보는 CP(인증 정책)와 CPS(인증 사례 명세서)라는 두 문서에서 캡처할 수 있습니다. CA 작업에 대한 중요한 정보를 캡처하기 위한 프레임워크는 RFC 3647을 참조하십시오.

가능한 경우 루트 CA 사용 최소화

루트 CA는 일반적으로 총 CA에 대한 인증서를 발급하기 위한 용도로만 사용해야 합니다. 이렇게 하면 총 CA가 최종 엔터티 인증서를 발급하는 일상적인 작업을 수행하는 동안 루트 CA를 손상 없이 저장할 수 있습니다.

그러나 조직에서 루트 CA에서 직접 최종 엔터티 인증서를 발급하려는 경우, ACM Private CA에서는 보안 및 운영 제어를 개선하면서 이 워크플로를 지원할 수 있습니다. 이 시나리오에서 최종 엔터티 인증서를 발급하면 루트 CA가 최종 엔터티 인증서 템플릿을 사용하도록 허용하는 IAM 권한 정책이 필요합니다. IAM 정책에 대한 자세한 내용은 AWS Certificate Manager Private Certificate Authority의 Identity and Access Management (p. 7) 단원을 참조하십시오.

Note

이 구성의 경우, 제한 조건으로 인해 운영 문제가 야기될 수 있습니다. 예를 들어 루트 CA가 손상되거나 손실된 경우 새 루트 CA를 생성해서 환경의 모든 클라이언트에 배포해야 합니다. 이러한 문제를 해결하기 위해 보안 프로세스가 완료될 때까지 새 인증서를 발급할 수 있습니다. 루트 CA에서 직접 인증서를 발급한 액세스를 제한하고 루트에서 발급된 인증서 수를 한정하지 않게 되므로 둘 다 루트 CA 관리에 대한 모범 사례로 간주됩니다.

루트 CA에 자체 AWS 계정 제공

두 개의 서로 다른 AWS 계정에 루트 CA와 하위 CA를 생성하는 것이 가장 좋은 방법입니다. 이렇게 하면 루트 CA에 대해 추가 보호 및 액세스 제어를 제공할 수 있습니다. 이를 위해 루트의 하위 CA에서 CSR를 내보내고 다른 계정의 루트 CA로 CSR를 서명하면 됩니다. 이 접근 방식의 이점은 계정별로 CA에 대한 제어를 분리할 수 있다는 것입니다. 단점은 루트 CA에서 하위 CA의 CA 인증서에 서명하는 프로세스를 단순화하기 위해 AWS 관리 콘솔 액세스 제어를 사용할 수 없다는 것입니다.

AWS CloudTrail 케기

사설 CA를 생성하고 작동을 시작하기 앞서 CloudTrail 기록을 활성화합니다. CloudTrail로 계정에 대한 AWS API 호출 이력을 검색하여 AWS 배포를 모니터링할 수 있습니다. 이 내역에는 AWS 관리 콘솔, AWS SDK, AWS Command Line Interface, 상위 수준 AWS 서비스에서 시작된 API 호출이 포함됩니다. 또한 어떤 사용 자 및 계정이 PCA API 연산을 호출했는지, 어떤 소스 IP 주소에 호출이 되었는지, 언제 호출이 발생했는지 등 확인할 수 있습니다. API를 사용해 CloudTrail을 애플리케이션에 통합시킴으로써 조직에 대한 추적 생성을 자동화하고 추적 상태를 확인하며 관리자가 CloudTrail 로깅을 활성화하고 비활성화하는 방식을 제어할 수 있습니다.
CA 프라이빗 키 업데이트

사설 CA에 대한 프라이빗 키를 정기적으로 업데이트하는 것이 가장 좋은 방법입니다. 새 CA 인증서를 가져와서 키를 업데이트하거나 사설 CA를 새 CA로 바꿀 수 있습니다.

미사용 CA 삭제

사설 CA를 영구적으로 삭제할 수 있습니다. 더 이상 CA가 필요하지 않거나 보다 최신의 프라이빗 키를 가진 CA로 이를 대체하고 싶은 경우에 영구적 삭제를 원할 수 있습니다. CA를 안전하게 삭제하려면 사설 CA 삭제 (p. 64)에 설명된 프로세스를 따르는 것이 좋습니다.

Note

AWS는 삭제가 될 때까지 CA에 대해 비용을 청구합니다.
ACM Private CA 배포 계획

ACM Private CA는 조직의 프라이빗 퍼블릭 키 인프라(PKI)에 대해 클라우드 기반의 완벽한 제어를 제공하며, 루트 CA(인증 기관)에서 하위 CA를 통해 최종 엔터티 인증서까지 확장합니다. 안전하고 유지 관리 및 확장이 가능하며 조직의 요구 사항에 적합한 PKI를 위해서는 철저한 계획이 필수적입니다. 이 단원에서는 CA 계층 구조 설계, 사설 CA 및 사설 최종 엔터티 인증서 수명 주기의 관리, 보안을 위한 모범 사례 적용에 대한 지침을 제공합니다.

주제
• CA 계층 구조 설계 (p. 41)
• 사설 CA 수명 주기 관리 (p. 47)
• AWS Certificate Manager Private Certificate Authority의 복원성 (p. 49)

CA 계층 구조 설계

ACM Private CA를 사용하여 최대 5개의 레벨로 인증 기관의 계층 구조를 생성할 수 있습니다. 계층 트리 맨 위에 있는 루트 CA에는 원하는 수만큼 분기를 가질 수 있습니다. 루트 CA는 각 분기에 최대 4개의 하위 CA를 가질 수 있습니다. 또한 각각이 자체 루트를 가진 계층을 여러 개 만들 수 있습니다.

잘 설계된 CA 계층 구조는 다음과 같은 이점을 제공합니다.
• 각 CA에 적합한 세분화된 보안 제어
• 로드 밸런싱 및 보안 향상을 위한 관리 작업 분할
• 일관된 운영에 대한 신뢰가 제한적인 해지 가능한 CA 사용
• 유효 기간 및 인증서 경로 제한

다음 다이어그램에서는 간단한 3단계 CA 계층 구조를 보여줍니다.
트리의 각 CA에는 서명 권한이 있는 X.509 v3 인증서(펜 및 종이 아이콘이 표시)가 지원됩니다. 이는 곧 CA가 해당 CA에 종속된 다른 인증서에 서명할 수 있다는 의미입니다. CA가 하위 레벨의 CA 인증서에 서명하면 서명된 인증서에 제한적이고 해지 가능한 권한이 부여됩니다. 레벨 1의 루트 CA는 레벨 2의 상위 레벨 하위 CA 인증서에 서명합니다. 이러한 CA는 최종 엔터티 인증서를 관리하는 공공 키 인프라(PKI) 관리자가 사용하는 레벨 3의 CA에 대한 인증서에 서명합니다.

CA 계층 구조의 보안은 트리의 가장 강력하게 구성되어 있어야 합니다. 이러한 구조는 루트 CA 인증서와 하위 CA 인증서와 그 아래의 최종 엔터티 인증서에 대한 신뢰를 기반으로 합니다. 최종 엔터티 인증서의 손상으로 인해 지역화된 손상이 발생할 수 있지만, 루트 CA의 손상이 전체 PKI의 신뢰를 파괴합니다. 루트 및 상위 레벨의 하위 CA는 자주 사용되지 않습니다(일반적으로 다른 CA 인증서에 서명하는 데 사용). 따라서 손상 위험을 낮출 수 있도록 인증서에 대해 엄격한 제어 및 감사가 이루어집니다. 계층 구조의 하위 레벨에서는 보안이 덜 제한적입니다. 이 접근 방법을 사용하면 사용자, 컴퓨터 호스트 및 소프트웨어 서비스에 대한 최종 엔터티 인증서를 발급하고 해지하는 일상적인 관리 작업을 수행할 수 있습니다.

Note

루트 CA를 사용하여 하위 인증서에 서명하는 것은 몇 가지 상황에서만 발생하는 드문 경우입니다.
- PKI가 생성되는 경우
- 상위 레벨의 인증 기관을 교체해야 하는 경우
- 인증서 해지 목록(CRL) 또는 온라인 인증서 상태 프로토콜(OCSP) 응답자를 구성해야 하는 경우

루트 및 기타 상위 레벨 CA에는 매우 안전한 운영 프로세스와 액세스 제어 프로토콜이 필요합니다.

주의
- 최종 엔터티 인증서 유효성 검사 (p. 43)
- CA 계층 구조 계획 (p. 44)
최종 엔터티 인증서 유효성 검사

최종 엔터티 인증서는 하위 CA를 통해 루트 CA로 되돌아가는 인증 경로에서 신뢰를 파생시킵니다. 웹 브라우저 또는 다른 클라이언트는 최종 엔터티 인증서가 제공될 때 신뢰 체인을 구성하려고 시도합니다. 예를 들어, 인증서의 발급자 고유 이름 및 주체 고유 이름이 발급 CA 인증서의 해당 필드와 일치하는지 확인할 수 있습니다. 클라이언트가 신뢰 저장소에 포함된 신뢰할 수 있는 루트에 도달할 때까지 계층 구조의 각 연속 레벨에서 일치 작업이 계속됩니다.

신뢰 저장소는 브라우저 또는 운영 체제에 포함된 신뢰할 수 있는 CA의 라이브러리입니다. 프라이빗 PKI의 경우, 조직의 IT 팀은 각 브라우저 또는 시스템이 이전에 사설 루트 CA를 신뢰 저장소에 추가했는지 확인해야 합니다. 그렇지 않으면 인증 경로의 유효성을 검사할 수 없으므로 클라이언트 오류가 발생합니다.

다음 그림은 최종 엔터티 X.509 인증서와 함께 표시될 때 브라우저가 따르는 유효성 검사 경로를 보여줍니다. 최종 엔터티 인증서에는 서명 권한이 없으며, 이러한 권한을 소유한 엔터티를 인증하는 역할만 합니다.

![최종 엔터티 인증서 유효성 검사](image)

브라우저는 최종 엔터티 인증서를 검사합니다. 브라우저는 인증서가 하위 CA(레벨 3)의 서명을 신뢰 자격 증명으로 제공한다는 것을 확인합니다. 하위 CA의 인증서는 동일한 PEM 파일에 포함되어야 합니다. 또는 신뢰 체인을 구성하는 인증서가 포함된 별도의 파일에 있을 수도 있습니다. 이를 찾으면 브라우저는 하위 CA(레벨 3)의 인증서를 확인하고 하위 CA(레벨 2)의 서명을 제공하는지 확인합니다. 그러면 하위 CA(레벨 2)가 루트 CA(레벨 1)의 서명을 신뢰 자격 증명으로 제공합니다. 브라우저는 신뢰 저장소에 사전 설치된 사설 루트 CA 인증서의 복사본을 찾아 최종 엔터티 인증서가 신뢰할 수 있는지 유효성 검사를 합니다.
일반적으로 브라우저는 인증서 해지 목록(CRL)과 비교하여 각 인증서를 확인합니다. 만료, 해지 또는 잘못 구성된 인증서는 거부되고 유효성 검사가 실패합니다.

CA 계층 구조 계획

일반적으로 CA 계층 구조에는 조직의 구조가 반영되어야 합니다. 관리 및 보안 역할을 위임하는 데 필요한 레벨 이상의 깊이(즉, CA 레벨 수)를 목표로 합니다. 계층에 CA를 추가하면 인증 경로에서 인증서 수가 늘어나며 유효성 검사 시간이 길어집니다. 깊이를 최소한으로 유지하면 신뢰를 설정할 때 서버에서 클라이언트로 전송되는 인증서 수가 줄어듭니다. 깊이가 작을수록 클라이언트가 최종 엔터티 인증서의 유효성을 검사하기 위해 수행하는 작업의 양도 줄어듭니다.

이론적으로 path-length 파라미터(p. 45)가 없는 루트 CA는 하위 CA의 깊이를 인증할 수 있습니다. 하나의 하위 CA는 내부 구성에서 허용하는 수 만큼 하위 레벨에 하위 CA를 가질 수 있습니다. ACM Private CA에서는 관리되는 계층 구조는 최대 5개 레벨 깊이의 CA 인증 경로를 지원합니다.

잘 설계된 CA 구조에는 몇 가지 이점이 있습니다.
- 서로 다른 조직 단위에 대한 별도의 관리 제어
- 하위 CA에 대한 액세스 권한을 위임하는 기능
- 추가 보안 제어를 통해 상위 레벨의 CA를 보호하는 계층 구조

두 가지 공통 CA 구조가 이 모든 것을 수행합니다.
- 2개의 CA 레벨: 루트 CA 및 하위 CA
  이것은 루트 CA 및 하위 CA에 대해 별도의 관리, 제어 및 보안 정책을 허용하는 가장 간단한 CA 구조입니다. 하위 CA에 대해서는 보다 허용적으로 액세스를 허락하면서 루트 CA에 대해서는 제한적인 제어 및 정책을 유지할 수 있습니다. 후자는 최종 엔터티 인증서의 대량 발급에 사용됩니다.

- 3개의 CA 레벨: 루트 CA 및 두 계층의 하위 CA
  위와 유사하게 이러한 구조는 루트 CA를 하위 레벨의 CA 작업과 분리하기 위해 CA 계층을 추가합니다. 중간 CA 계층은 최종 엔터티 인증서의 발급을 수행하는 하위 CA에 서명하는 데만 사용됩니다.

 덜 일반적인 CA 구조는 다음과 같습니다.
- 4개 이상의 CA 레벨
  레벨이 3개인 계층 구조보다 덜 일반적이지만 레벨이 4개 이상인 CA 계층 구조가 가능하며, 이러한 구조는 관리 위임을 허용하는 데 필요할 수 있습니다.
- 1개의 CA 레벨: 루트 CA만
  이 구조는 보통 전체 신뢰 체인이 필요하지 않을 때 개발 및 테스트에 사용됩니다. 프로덕션 환경에서 사용되며 비정형적입니다. 또한 루트 CA와 최종 엔터티 인증서를 발급하는 CA에 대해 별도의 보안 정책을 유지하는 모범 사례를 위반합니다.

그러나 루트 CA에서 직접 인증서를 발급하는 경우에는 ACM Private CA로 마이그레이션할 수 있습니다. 이렇게 하면 OpenSSL 또는 기타 소프트웨어로 관리되는 루트 CA를 사용하는 것보다 보안 및 제어 측면에서 이점이 있습니다.

제조업체용 프라이빗 PKI의 예

이 예에서는 어떤 가상의 기술 업체가 스마트 전구와 스마트 토스터라는 2개의 사물 인터넷(IoT) 제품을 제조합니다. 생산 중에 각 디바이스에 최종 엔터티 인증서가 발급되므로 인터넷을 통해 제조업체와 안전하게 통
인증 경로에서 길이 제약 조건 설정

CA 계층의 구조는 각 인증서에 포함된 기본 제약 조건 확장에 의해 정의되고 적용됩니다. 확장은 두 가지 제약 조건을 정의합니다.

- **cA** – 인증서가 CA를 정의하는지 여부. 이 값이 false(기본값)이면 인증서는 최종 엔터티 인증서입니다.
AWS Certificate Manager Private
Certificate Authority 사용 설명서
인증 경로에서 길이 제약 조건 설정

- pathLenConstraint - 인증서를 포함하는 인증 경로의 최대 추가 길이.

루트 CA 인증서에는 최대한의 유연성이 요구되며, 경로 길이 제약 조건이 포함되지 않습니다. 이렇게 하면 루트가 모든 길이의 인증 경로를 정의할 수 있습니다.

Note

ACM Private CA은 인증 경로를 5개 레벨로 제한합니다.

하위 CA는 계층 구조 배치에서 위치와 원하는 기능에 따라 0 이상의 pathLenConstraint 값을 갖습니다. 예를 들어 CA가 3개인 계층 구조에서는 루트 CA에 대해 경로 제약 조건이 지정되지 않습니다. 첫 번째 하위 CA는 경로 길이가 1이므로 하위 CA에 서명할 수 있습니다. 이러한 각 하위 CA의 pathLenConstraint 값은 반드시 0이어야 합니다. 이는 최종 엔터티 인증서에 서명할 수 있지만 추가 CA 인증서를 발급할 수는 없다는 의미입니다. 새 CA를 생성할 수 있는 기능을 제한하는 것은 중요한 보안 제어책입니다.

다음 그림은 이렇게 제한된 권한이 계층 구조 아래로 전파되는 것을 보여줍니다.

Root CA
Path length: Unspecified

Subordinate CA
Path length: 2

Subordinate CA
Path length: 1

Subordinate CA
Path length: 0

Depth of child CAs allowed below each point

이렇듯 레벨이 4개인 계층 구조에서는 루트가 제약되지 않습니다(항상 그렇듯이). 그러나 첫 번째 하위 CA의 pathLenConstraint 값은 2입니다. 이 값은 하위 CA의 길이가 2개 레벨 이상이 되지 못하도록 제한합니다. 따라서 유효한 인증 경로의 경우, 제약 조건 값이 다음 두 레벨에서 0으로 감소해야 합니다. 웹 브라우저가 이하 두 경로 길이가 4보다 큰 분기에서 최종 엔터티 인증서를 발견하면 유효성 검사가 실패합니다. 실수로 생성된 CA, 잘못 구성된 CA 또는 무단 발급의 결과로 이러한 인증서가 생성될 수 있습니다.

템플릿을 사용하여 길이 관리

ACM Private CA에서는 루트, 하위 및 최종 엔터티 인증서를 발급하기 위한 템플릿을 제공합니다. 이러한 템플릿은 경로 길이를 포함하여 기본 제약 조건 값에 대한 모범 사례를 캡슐화합니다. 템플릿에는 다음과이 포함됩니다.
사용 설명서

사용 설명서

AWS Certificate Manager Private
Certificate Authority 사용 설명서
CA 수명 주기 관리

• RootCACertificate/V1
• SubordinateCACertificate_PathLen0/V1
• SubordinateCACertificate_PathLen1/V1
• SubordinateCACertificate_PathLen2/V1
• SubordinateCACertificate_PathLen3/V1
• EndEntityCertificate/V1

경로 길이를 발급 CA 인증서의 경로 길이보다 크거나 같게 해서 CA를 생성하려고 하면 IssueCertificate API가 오류를 반환합니다.

인증서 템플릿에 대한 자세한 내용은 인증서 템플릿 (p. 71) 단원을 참조하십시오.

AWS CloudFormation을 이용해 CA 계층 구조 설정 자동화

CA 계층 구조에 대한 설계를 결정했으면 이제는 AWS CloudFormation 템플릿을 사용해 이를 테스트하고 프로덕션 환경에 적용할 수 있습니다. 이러한 템플릿의 예는 AWS CloudFormation 사용 설명서의 사설 CA 계층 구조 선언을 참조하십시오.

사설 CA 수명 주기 관리

사설 CA는 수명(유효 기간)이 고정되어 있습니다. CA 인증서가 만료되면 CA 계층 구조에서 아래에 있는 하위 CA에서 직접 또는 간접적으로 발급한 모든 인증서가 유효하지 않게 됩니다. 미리 계획하면 CA 인증서 만료를 방지할 수 있습니다.

유효 기간 선택

X.509 인증서의 유효 기간은 필수적인 기본 인증서 필드입니다. 발급 CA가 인증서를 신뢰할 수 있음을 인증하는 기간 동안 해지를 금지하는 시간 범위를 결정합니다 (자체 서명된 루트 인증서는 자체 유효 기간을 인증).

ACM Private CA 및 AWS Certificate Manager을 사용하면 다음과 같은 제약 조건에 따라 인증서 유효 기간을 구성하는 데 도움이 될 수 있습니다.

• ACM Private CA에서 관리하는 인증서의 유효 기간은 인증서를 발급한 CA의 유효 기간보다 짧거나 같아야 합니다. 즉, 하위 CA 및 최종 엔터티 인증서의 유효 기간이 상위 인증서보다 길 수 없습니다. IssueCertificate API를 사용하여 유효 기간이 상위 CA보다 크거나 같은 CA 인증서를 발급하려고 하는 시도는 실패합니다.

• AWS Certificate Manager에서 발급하고 관리하는 인증서(ACM가 프라이빗 키를 생성하는 인증서)의 유효 기간은 13개월입니다. ACM은 이러한 인증서의 경신 프로세스를 관리합니다. ACM Private CA를 사용하여 인증서를 직접 발급하는 경우에는 유효 기간을 선택할 수 있습니다.

다음 다이어그램은 중첩된 유효 기간의 일반적인 구성을 보여 줍니다. 루트 인증서는 유효 기간이 가장 길고 최종 엔터티 인증서는 유효 기간이 비교적 짧으며, 하위 CA도 중간 정도입니다.
AWS Certificate Manager Private
Certificate Authority 사용 설명서
CA 승계 관리

CA 계층 구조를 계획할 때 CA 인증서의 최적 수명을 결정하십시오. 발급하려는 최종 엔터티 인증서의 원하는 수명에서 거슬러 내려가십시오.

최종 엔터티 인증서
최종 엔터티 인증서에는 사용 사례에 적합한 유효 기간이 있어야 합니다. 수명이 짧으면 포라이빗 키를 분실하거나 도난당할 경우 인증서 노출을 최소화할 수 있습니다. 그러나 수명이 짧으면 자주 경신을 해야 합니다. 만료되는 인증서를 경신하지 않으면 가동 중지가 발생할 수 있습니다.

최종 엔터티 인증서를 분산 사용하면 보안 위반이 발생할 경우 위치적으로 문제가 발생할 수 있습니다. 계획을 수립할 때는 경신 및 배포 인증서, 순상된 인증서의 해지, 인증서를 사용하는 클라이언트에 해지 내용이 전파되는 속도를 고려해야 합니다.

ACM을 통해 발급된 최종 엔터티 인증서의 기본 유효 기간은 13개월입니다. ACM Private CA에서는 발급 CA의 유효 기간보다 작기만 하더라도 IssueCertificate API를 사용하여 어떤 유효 기간에도 적용할 수 있습니다.

하위 CA 인증서
하위 CA 인증서는 발급되는 인증서보다 유효 기간이 훨씬 길어야 합니다. CA 인증서의 유효 기간은 발급되는 하위 CA 인증서 또는 최종 엔터티 인증서의 유효 기간보다 2~5배 간 것이 좋습니다. 예를 들어, 레벨이 2개인 CA 계층 구조(루트 CA 및 1개의 하위 CA)가 있다고 가정합니다. 유효 기간이 1년인 최종 엔터티 인증서를 발급하려는 경우, 하위 발급 CA의 유효 기간을 3년으로 구성할 수 있습니다. 이는 ACM Private CA에서 하위 CA 인증서의 기본 유효 기간입니다. 루트 CA 인증서를 교체하지 않고 하위 CA 인증서를 변경할 수 있습니다.

루트 인증서
루트 CA 인증서를 변경하면 전체 PKI에 영향을 미치며, 모든 종속 클라이언트 운영 체제 및 브라우저 신뢰 저장소를 업데이트해야 합니다. 운영에 미치는 영향을 최소화하려면 루트 인증서에 대해 긴 유효 기간을 선택해야 합니다. 루트 인증서의 ACM Private CA 기본값은 10년입니다.

CA 승계 관리
이전 CA를 교체하거나 새 유효 기간으로 CA를 다시 발급하는 등 두 가지 방법으로 CA 승계를 관리할 수 있습니다.

이전 CA 교체
이전 CA를 교체하려면 새 CA를 생성하고 이를 동일한 상위 CA에 연결합니다. 그런 다음 새 CA에서 인증서를 발급합니다.

새 CA에서 발급된 인증서에는 새 CA 체인이 있습니다. 새 CA가 설정되면 이전 CA를 비활성화하여 새 인증서를 발급하지 못하도록 할 수 있습니다. 비활성화가 된 이전 CA에서는 CA에서 발급된 이전 인증서를 해지
CA 해지

CA 인증서를 해지하면 CA에서 발급한 모든 인증서를 효과적으로 해지할 수 있습니다. 해지 정보는 CRL을 통해 클라이언트에게 배포됩니다. 클라이언트는 업데이트된 CRL을 받고 CA에서 발급한 인증서에 대한 신뢰를 중단합니다. 모든 최종 엔터티 및 CA 인증서를 효과적으로 해지하려는 경우에만 CA 인증서를 해지해야 합니다.

Note

CA를 교체한 경우에는 ACM을 통해 발급된 사설 인증서를 갱신할 수 없습니다. 발급 및 갱신에 ACM를 사용하는 경우 CA 인증서를 다시 발급하여 CA의 유효 기간을 연장해야 합니다.

AWS Certificate Manager Private Certificate Authority의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역으로 구축됩니다. AWS 리전에서는 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하여 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

이중화 및 재배 복구

CA 계층 구조를 계획할 때 이중화 및 재배 복구를 고려하십시오. ACM 사설 CA는 여러 리전에서 사용할 수 있으므로 여러 리전에서 이중 CA를 생성할 수 있습니다. 이러한 ACM Private CA 서비스는 가용성이 99.9%인 서비스 수준 계약(SLA)에 따라 작동합니다. 이중화 및 재배 복구를 위해 고려할 수 있는 접근 방법은 최소 두 가지입니다. 루트 CA 또는 가장 높은 레벨의 하위 CA에서 이중화 구성을 할 수 있습니다. 각 접근법에는 장단점이 있습니다.

1. 이중화 및 재배복구를 위해 서로 다른 2개의 AWS 리전에서 2개의 루트 CA를 생성할 수 있습니다. 이 구성을 사용하면 각 루트 CA가 각 AWS 리전에 독립적으로 작동하므로 단일 리진에서 재해가 발생할 때 시스템을 보호할 수 있습니다. 그러나 이중화된 루트 CA를 생성하면 운영 복잡성이 증가합니다. 두 루트 CA 인증서를 모두 사용자 환경의 브라우저 및 운영 체제의 신뢰 자료소에 배포해야 합니다.

2. 동일한 루트 CA에 연결되는 2개의 이중화된 하위 CA를 생성할 수도 있습니다. 이 접근 방법의 이점은 사용자 환경의 신뢰 자료소에 단일 루트 CA 인증서만 배포하면 된다는 것입니다. 재해 사항은 루트 CA가 존재하는 AWS 영향을 주는 재해가 발생할 경우 이중화된 루트 CA가 없다는 것입니다.

버전 latest
49
사설 CA 생성 및 관리

ACM Private CA를 사용하면 조직에서 내부적으로 사용할 수 있도록 전적으로 AWS에서 호스팅하는 사설 CA(인증 기관)를 생성할 수 있습니다. 루트 CA 인증서와 하위 CA 인증서는 모두 ACM Private CA에서 저장 및 관리합니다. 루트 인증 기관의 프라이빗 키는 AWS에 안전하게 저장됩니다.

AWS Management 콘솔, AWS CLI 및 ACM Private CA API를 사용하여 ACM Private CA에 액세스할 수 있습니다. 다음 주제는 콘솔 및 CLI를 사용하는 방법을 보여줍니다. API에 대한 자세한 내용은 ACM Private CA API 사용(Java 예제) (p. 80) 단원을 참조하십시오.

주제
• ACM Private CA 설정 (p. 50)
• 사설 CA 생성 (p. 51)
• 사설 CA에 대한 인증서 생성 및 설치 (p. 57)
• 사설 CA 업데이트 (p. 59)
• 사설 CA 삭제 (p. 64)
• 사설 CA 복원 (p. 65)
• ACM에 인증서 갱신 권한 할당 (p. 67)

ACM Private CA 설정

ACM Private CA를 사용하기 전에 몇 가지 작업을 완료해야 할 수 있습니다.

AWS에 가입

이미 Amazon Web Services(AWS) 고객이 아닌 경우, ACM Private CA를 사용하려면 가입을 해야 합니다. 계정은 사용 가능한 모든 서비스에 자동으로 액세스할 수 있지만, 사용하는 서비스에 대해서만 요금이 청구됩니다. 또한 AWS 신규 고객의 경우, 일부 서비스를 제한된 기간 동안 무료로 사용할 수 있습니다. 자세한 내용은 AWSAWS 프리 티어를 참조하십시오.

Note

ACM Private CA는 프리 티어에서 사용할 수 없습니다.
사설 CA 생성

이 단원에서는 ACM Private CA를 사용하여 사설 CA(인증 기관)를 생성하는 방법에 대해 설명합니다. 이러한 절차를 사용하여 루트 CA와 하위 CA 모두 생성할 수 있으므로 조직의 요구 사항과 일치하는 트러스트 관계에 대해 간단한 계층을 만들 수 있습니다.

CA를 사용하여 사용자, 디바이스 및 애플리케이션의 최종 엔터티 인증서에 서명하는 방법은 사설 최종 엔터티 인증서 발급 (p. 68) 단원을 참조하십시오.

Note

계정에는 사설 CA를 생성한 시점부터 각 사설 CA에 대한 월별 요금이 청구됩니다.

AWS Command Line Interface 설치(선택 사항)

AWS CLI를 설치하지 않았지만 사용하고 싶은 경우에는 AWS Command Line Interface의 지침을 따르십시오.

사설 CA 생성

이 단원에서는 ACM Private CA를 사용하여 사설 CA(인증 기관)를 생성하는 방법에 대해 설명합니다. 이러한 절차를 사용하여 루트 CA와 하위 CA 모두 생성할 수 있으므로 조직의 요구 사항과 일치하는 트러스트 관계에 대해 간단한 계층을 만들 수 있습니다.

CA를 사용하여 사용자, 디바이스 및 애플리케이션의 최종 엔터티 인증서에 서명하는 방법은 사설 최종 엔터티 인증서 발급 (p. 68) 단원을 참조하십시오.

Note

계정에는 사설 CA를 생성한 시점부터 각 사설 CA에 대한 월별 요금이 청구됩니다.

AWS 콘솔을 사용하여 프라이빗 API를 생성하는 방법

1. AWS 계정으로 로그인하여 https://console.aws.amazon.com/acm-pca/home에서 ACM Private CA 콘솔을 엽니다. CA가 없는 리전으로 콘솔이 열리면 소개 페이지가 나타납니다. 사설 인증 기관 아래에 있는 시작하기를 선택합니다. 시작하기를 다시 선택합니다. 이미 CA가 하나 이상 존재하는 리전으로 콘솔이 열리면 소개 페이지가 나타나지 않습니다. Private CAs(사설 CA)와 Create CA(CA 생성)를 차례로 선택합니다.

2. CA(인증 기관) 유형 선택 페이지에서 생성하려는 사설 CA의 유형을 선택합니다.
   - 루트 CA를 선택하면 새 CA 계층이 설정됩니다. 이 CA는 자체 서명된 인증서에 의해 지 novamente 있습니다. 이는 계층의 다른 CA 및 최종 엔터티 인증서에 대한 궁극적 서명 기관 역할을 합니다.
   - 하위 CA를 선택하면 계층에서 위에 있는 상위 CA가 서명해야 하는 CA가 생성됩니다. 하위 CA는 일반적으로 다른 하위 CA를 생성하거나 사용자, 컴퓨터 및 애플리케이션에 최종 엔터티 인증서를 발급하는 데 사용됩니다.

CA 선택을 완료한 후 다음을 선택합니다.

3. Configure the certificate authority (CA) subject name(CA(인증 기관) 주체 이름 구성)에서 사설 CA의 주체 이름을 구성합니다. 다음 값 중 하나 이상을 입력해야 합니다.
   - 조직(O)
   - 조직 단위(OU)
   - 국가 이름(C)
   - 주 또는 지방 이름
   - 시 이름
   - 일반 이름(CN)

백업 인증서는 자체 서명되므로 사설 CA에 제공하는 주체 정보는 공용 CA에 포함된 정보보다 더 적을 수 있습니다. 주체 고유 이름을 구성하는 각 값에 대한 자세한 내용은 X.500 고유 이름 (p. 134) 단원을 참조하십시오.

완료되면 다음을 선택합니다.

4. Configure the certificate authority (CA) key type(CA(인증 기관) 키 유형 구성)에서 사설 CA의 키 유형을 선택합니다.
   - 버전 latest
   - 51
사설 CA 생성

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

선택을 하고 Next(다음)을 선택합니다.

5. 인증서 해지 구성 페이지에는 ACM Private CA에서 관리하는 인증서 해지 목록(CRL)을 생성할 수 있는 옵션이 있습니다. 웹 브라우저 같은 클라이언트가 CRL을 취득하여 최종 엔터티 또는 하위 CA 인증서를 신뢰할 수 있는지 여부를 확인합니다. 자세한 내용은 사설 인증서 해지(p. 69) 단원을 참조하십시오.

ACM Private CA는 사용자가 지정한 Amazon S3 버킷에 CRL을 자동으로 배치하고 주기적으로 업데이트합니다.

Note

CRL은 해당 인증서를 참조하는 인증서가 발급된 경우에만 Amazon S3에 배치됩니다. 그 전에는 Amazon S3 버킷에 acm-pca-permission-test-key 파일만 표시됩니다.

CRL에 암호화 보호를 추가하는 방법에 대한 자세한 내용은 CRL 암호화 (p. 55) 단원을 참조하십시오.

각 CRL은 DER 인코딩 파일입니다. 파일을 다운로드하고 OpenSSL을 사용해 파일을 보려면 다음과 같은 명령을 사용하십시오.

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

CRL은 다음 형식을 취합니다.

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:
X509v3 CRL Number:
1519676905984
Revoked Certificates:
  Serial Number: E8CBD2B8DB122329F97706BCFEC990F8
  Revocation Date: Feb 26 20:00:36 2018 GMT
  CRL entry extensions:
    X509v3 CRL Reason Code:
      Key Compromise
  Serial Number: F7D7A3FD8882C6776483467BBF0B38C
  Revocation Date: Jan 30 21:21:31 2018 GMT
  CRL entry extensions:
    X509v3 CRL Reason Code:
      Key Compromise
Signature Algorithm: sha256WithRSAEncryption
fa:38:ff:ec:4c:0d:97:ac:02:51:26:7d:3e:fe:a6:
AWS Certificate Manager Private
Certificate Authority 사용 설명서
사설 CA 생성

CRL을 CA와 연결하면 ACM Private CA이 CA에서 발급된 인증서에 CRL 배포 지점 확장을 포함시킵니다. 이 확장은 클라이언트 디바이스가 인증서 해지 상태를 확인할 수 있도록 CRL에 대한 URL을 제공합니다.

CRL을 생성하려면 다음 단계를 완료하십시오.

1. CRL 배포 활성화를 선택합니다.
2. CRL 항목에 대한 Amazon S3 버킷을 새로 생성하려면 S3 버킷 새로 생성 옵션에서 예를 선택하고 고유한 버킷 이름을 입력합니다. (버킷에 대한 경로를 포함할 필요는 없습니다.) 그렇지 않으면 아니오를 선택하고 목록에서 기존 버킷을 선택합니다.
   예를 선택한 경우에는 ACM Private CA가 사용자에게 필요한 버킷 정책을 생성합니다. 아니오를 선택한 경우에는 다음 정책이 버킷에 연결되어 있는지 확인합니다. 자세한 내용은 버킷 정책 추가를 참조하십시오.

```json
{
  "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:::your-bucket-name/*",
        "arn:aws:s3:::your-bucket-name"
      ]
    }
  ]
}
```

Note
계정에 Amazon S3 퍼블릭 액세스 차단 설정이 적용되는 경우 사설 CA에서 CRL 버킷을 생성하지 못할 수 있습니다. 이 경우 Amazon S3 설정을 확인하십시오. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 단원을 참조하십시오.

3. 추가 구성 옵션을 보려면 고급을 확장합니다.
   • 사용자 지정 CRL 이름을 추가하여 Amazon S3 버킷에 대한 변경을 생성합니다. 이 이름은 RFC 5280에서 정의한 ‘CRL 배포 지점’ 확장에 따라 CA에서 발급한 인증서에 포함되어 있습니다.
   • CRL의 유효 일수를 입력합니다. 기본값은 7일입니다. 온라인 CRL의 경우, 유효 기간은 일반적으로 2~7일입니다. ACM Private CA에서는 지정된 기간의 1/2 시점에 CRL 재생성을 시도합니다.

[Next]를 선택합니다.
6. 선택에 따라 태그 추가 페이지에서 인증서에 태그를 지정할 수 있습니다. 태그는 AWS 리소스를 식별하고 구성하기 위한 메타데이터 역할을 하는 키/값 쌍입니다. ACM Private CA 태그 파라미터의 목록과 생성 후 CA에 태그를 추가하는 방법에 대한 자침은 사설 CA에 태그 추가 (p. 63) 단원을 참조하십시오. [Next]를 선택합니다.

7. CA 권한 구성

선택에 따라 자동 갱신 권한을 ACM 서비스 보안 주체에 위임합니다. ACM 는 이러한 권한이 부여된 경우에만 이 CA에서 생성된 사설 최종 엔터티 인증서를 자동으로 갱신할 수 있습니다. ACM Private CA API를 사용하여 언제든지 갱신 권한을 할당할 수 있습니다.

기본적으로 이러한 권한을 사용하도록 설정되어 있습니다. [Next]를 선택합니다.

8. 검토 및 생성 페이지에서 구성이 올바른지 확인하고 확인란을 선택하여 가격 정보를 확인한 다음, 확인 및 생성을 선택합니다.

CA 인증서를 계속해서 생성 및 설치하고 싶으면 성공 대화 상자에서 시작하기를 선택하고 사설 CA에 대한 인증서 생성 및 설치 (p. 57)의 지침을 따릅니다. 그렇지 않고 You can also finish this later(나중에 이 작업을 완료할 수도 있음)를 선택하면 사설 CA 목록으로 이동됩니다.

AWS CLI를 사용하여 사설 CA를 생성하는 방법

create-certificate-authority 명령을 사용하여 사설 CA를 생성합니다. CA 구성, 해지 구성 및 CA 유형을 지정해야 합니다. 선택에 따라 멱등성 토큰 또는 태그를 추가 할 수 있습니다. CA 구성에서 다음을 지정합니다.

• 알고리즘의 이름
• CA 프라이빗 키를 생성하는 데 사용되는 키 크기
• CA가 서명에 사용하는 서명 알고리즘 유형
• X.500 주체 정보

CRL 구성에서 다음을 지정합니다.

• CRL 만료 기간 일수(CRL의 유효 기간)
• CRL이 포함되는 Amazon S3 버킷
• CA가 발급하는 인증서에 포함된 S3 버킷의 CNAME 별칭

이 명령에서 사용하도록 아래 예제 파일을 수정할 수 있습니다.

C:\ca_config.txt

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

C:\revoke_config.txt
CRL 암호화

선택에 따라 CRL이 포함된 Amazon S3 버킷에서 암호화를 구성할 수 있습니다. ACM Private CA는 S3의 자산에 대해 두 가지 암호화 모드를 지원합니다.

• Amazon S3 관리형 AES-256 키를 사용한 자동 서버 측 암호화.
• 사양에 맞게 구성된 CMK(고객 마스터 키)와 AWS Key Management Service를 사용한 고객 관리 암호화.

Note

ACM Private CA는 S3에 의해 자동으로 생성된 기본 CMK 사용을 지원하지 않습니다.

다음 절차에서는 각 암호화 옵션을 설정하는 방법에 대해 설명합니다.

자동 암호화를 구성하려면

S3 서버 측 암호화를 활성화하려면 다음 단계를 수행하십시오.

2. 버킷 테이블에서 ACM Private CA 자산을 보존할 버킷을 선택합니다.
3. 버킷의 페이지에서 속성 탭을 선택합니다.
4. 기본 암호화 카드를 선택합니다.
5. AES-256을 선택합니다.
6. 필요에 따라 버킷 권한 정책을 확인한 다음 저장을 선택합니다.
사용자 지정 암호화를 구성하려면

사용자 지정 CMK를 사용하여 암호화를 활성화하려면 다음 단계를 수행하십시오.

1. (선택 사항) AWS KMS CMK가 아직 없는 경우 다음 AWS CLI `create-key` 명령을 사용하여 생성합니다.

   ```
   aws kms create-key
   ```

   이 명령의 출력 화면에는 CMK의 키 ID와 Amazon 리소스 이름(ARN)이 포함됩니다. 다음은 예제 출력입니다.

   ```
   {
   "KeyMetadata": {
   "KeyId": "6f815f63-e628-448c-8251-e40cb0d29f59",
   "Description": "",
   "Enabled": true,
   "KeyUsage": "ENCRYPT_DECRYPT",
   "KeyState": "Enabled",
   "CreationDate": 1478910250.94,
   "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-e40cb0d29f59",
   "AWSAccountId": "123456789012"
   }
   }
   ```

2. 모든 AWS KMS CMK는 기본적으로 비공개입니다. 따라서 리소스 소유자가 직접 사용해서 데이터를 암호화 및 해독할 수 있습니다. 그러나 리소스 소유자가 원한다면 다른 사용자 및 리소스에게 CMK에 대한 액세스 권한을 부여할 수 있습니다. 다음 단계를 사용하여 ACM Private CA 서비스 보안 주체에 CMK를 사용할 수 있는 권한을 부여합니다. 서비스 보안 주체는 CMK가 저장된 위치와 동일한 리전에 있어야 합니다.

   a. 먼저, 아래 `get-key-policy` 명령을 사용하여 CMK에 대한 기본 정책을 `policy.json`로 저장합니다.

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

   b. 텍스트 편집기에서 `policy.json` 파일을 열고 다음 설명을 추가합니다.

   ```
   {
   "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/*"
   ]
   }
   }
   }
   ```
사설 CA에 대한 인증서 생성 및 설치

사설 CA 인증서를 생성 및 설치하려면 다음 절차를 수행하십시오. 그러면 CA를 사용할 준비가 완료됩니다.

ACM Private CA는 CA 인증서 설치를 위한 세 가지 시나리오를 지원합니다.

• ACM Private CA에서 호스팅하는 루트 CA에 대한 인증서 설치
• 상위 기관이 ACM Private CA에 의해 호스팅되는 하위 CA 인증서 설치
• 상위 기관이 외부에서 호스팅되는 하위 CA 인증서 설치

다음 단원에서는 각 시나리오에 대한 절차에 대해 설명합니다. 콘솔 절차가 콘솔 페이지 사설 CA에서 시작됩니다.

루트 CA 인증서를 설치 중인 경우

사설 루트 CA에 대한 인증서를 생성 및 설치하는 방법

1. 이전에 루트 CA를 생성하고 성공 창에서 시작하기를 선택한 경우에는 루트 CA 인증서 설치 마법사로 직접 연결되었습니다. 인증서 설치의 연기하도록 선택한 경우 상태가 보류 중인 인증서 또는 활성인 루트 CA를 선택한 다음, 마지막으로 CA 인증서 설치를 선택하여 설치를 시작합니다. 이렇게 하면 루트 CA 인증서 설치 마법사가 열립니다.

2. 루트 CA 인증서 파라미터 지정, 섹션에서 다음과 같은 인증서 파라미터를 지정하십시오.
   • Validity(유효 기간) — 연, 월 또는 일 단위로 CA 인증서가 만료되는 시기를 결정합니다. 루트 CA 인증서의 ACM Private CA 기본 유효 기간은 10년입니다.
   • 서명 알고리즘 — 루트 CA가 새 인증서를 발급할 때 사용할 서명 알고리즘을 지정합니다.

3. 루트 CA 인증서 검토, 생성 및 설치 페이지에서 구성이 올바른지 확인하고 확인 및 설치를 선택합니다.

ACM Private CA에서 호스팅하는 하위 CA 인증서를 설치하는 경우

ACM Private CA에서 호스팅하는 하위 CA에 대한 인증서 생성 및 설치하는 방법

1. 이전에 루트 CA를 생성하고 성공 창에서 시작하기를 선택한 경우에는 하위 CA 인증서 설치 콘솔로 직접 연결되었습니다. 인증서 설치를 연기하도록 선택한 경우 https://console.aws.amazon.com/acm-pca/
외부 상위 CA에서 서명한 하위 CA 인증서를 설치하는 경우

외부 상위 CA에서 서명한 하위 CA 인증서를 생성 및 설치하는 방법

1. 이전에 투트 CA를 생성하고 성공 칭에서 시작하기 전에 하위 CA 인증서 설치 콘솔의 작업을 선택한 경우에 한해 하위 CA 인증서 설치 콘솔로 직접 연결되었습니다. 인증서 설치를 연기할 수 있도록 선택할 경우 https://console.aws.amazon.com/acm-pca/home에서 ACM Private CA 콘솔을 열고 선택한 상위 CA 인증서를 설치한 경우, 마지막으로 CA 인증서 설치를 연기하고 설치를 시작할 수 있습니다. 이렇게 하면 하위 CA 인증서 설치 콘솔이 열립니다.

2. 하위 CA 인증서 설치 콘솔에서 다음 옵션을 선택합니다.
   - [Next]를 선택합니다.

3. 인증서 서비스 요청(CSR) 내보내기 페이지에서 ACM Private CA 인증서 정보를 CSR를 생성하고 표시합니다. CSR를 파일로 내보내는 옵션을 선택할 수 있습니다.
CSR을 파일로 내보냈고 외부 상위 CA가 파일에 서명한 경우에는 다음을 선택합니다.

4. 서명된 인증 기관(CA) 인증서 가져오기 페이지에서 서명된 CA 인증서 및 인증서 체인(중간 인증서 포함)을 가져옵니다.

[Next]를 선택합니다.

5. 검토 및 설치 페이지에서 구성이 올바른지 확인하고 확인 및 설치를 선택합니다.

사설 CA 목록 페이지로 돌아가 상단에 설치 상태(성공 또는 실패)가 표시됩니다. 설치가 성공하면 새로 완료된 루트 CA의 상태가 목록에 활성으로 표시됩니다.

사설 CA 업데이트

사설 CA를 생성하고 나면 상태 또는 해지 구성물을 업데이트할 수 있습니다.

CA 상태 업데이트

사용자 작업 또는 경우에 따라 CA가 만료되는 경우와 같은 서비스 작업의 ACM Private CA 결과로 인해 관리되는 CA의 상태입니다. CA 관리자가 사용할 수 있는 상태 옵션은 CA의 현재 상태에 따라 다릅니다.

ACM Private CA는 다음과 같은 상태 값을 보고할 수 있습니다.

- **CREATING** – ACM Private CA가 사설 인증 기관을 생성 중입니다.
- **PENDING_CERTIFICATE** – CA가 생성되었으며 이제 작동 중인 인증서가 필요합니다. ACM Private CA에서 호스팅한 루트 또는 하위 CA나 온프레미스 루트 또는 하위 CA를 사용해 사설 CA CSR을 서명한 다음, 이를 ACM Private CA에 가져와야 합니다. 자세한 내용은 사설 CA에 대한 인증서 생성 및 설치 (p. 57) 단원을 참조하십시오.
- **ACTIVE** – 사설 CA는 활성 상태이고 인증서를 생성하고 발급하는 데 사용할 수 있습니다.
- **DISABLED** – 사설 CA가 비활성화 되었습니다.
- **EXPIRED** – 사설 CA의 CA 인증서의 기간이 만료되면 ACM Private CA가 상태를 EXPIRED로 설정합니다.

고려 사항:

- CA 인증서는 기본적으로 자동 갱신되지 않습니다. AWS Certificate Manager을 통한 갱신 자동화에 대한 자세한 내용은 ACM에 인증서 갱신 권한 할당 (p. 67) 단원을 참조하십시오.
- 만료된 CA는 더 이상 CRL을 생성하지 않습니다.
- 감사 보고서는 만료된 CA에서 계속 작동합니다.
- 계정에서 만료된 CA에 대한 요금이 계속 청구됩니다.
- 만료된 CA의 상태를 직접 변경할 수 없습니다. CA에 대한 새 인증서를 가져오면 해당 상태가 만료되기 전의 상태(활성 또는 비활성)로 돌아갑니다.
- 사설 CA에 대한 새로운 CA 인증서를 가져오는 경우, CA 인증서의 기간이 만료된 후에 DISABLED로 설정하지 않는 한 ACM Private CA가 ACTIVE로 상태를 재설정합니다.
- 만료된 CA로 새 인증서를 발급하려고 하면 IssueCertificate API가 InvalidStateException을 반환합니다. 만료된 루트 CA는 새 하위 인증서를 발급하기 앞서 새 루트 CA 인증서에 자체 서명해야 합니다.
- The ListCertificate Authorities 및 DescribeCertificateAuthority API는 CA 상태가 ACTIVE 또는 DISABLED로 설정되어 있는지 여부에 관계없이 CA 인증서가 만료된 경우에 EXPIRED 상태를 반환합니다. 그러나 만료된 CA가 DELETED로 설정된 경우 반환된 상태는 DELETED입니다.
- UpdateCertificateAuthority API는 만료된 CA의 상태를 업데이트할 수 없습니다.
- RevokeCertificate API를 사용하여 CA 인증서를 포함하여 만료된 모든 인증서를 해지할 수 있습니다.
• **FAILED** – CreateCertificateAuthority 작업이 실패했습니다. 이러한 실패는 네트워크 중단, 백엔드 AWS 실패 또는 기타 오류로 인해 발생할 수 있습니다.

고려 사항:
• 실패한 CA는 복구가 불가능합니다. CA를 삭제하고 새 CA를 생성합니다.
• **DELETED** – 사설 CA는 복원 기간 (7-30일) 내에 있습니다. 이 기간이 지나면 영구적으로 삭제됩니다.

고려 사항:
• 상태가 DELETED이고 인증서가 만료된 CA에서 RestoreCertificateAuthority API를 호출하면 CA가 EXPIRED로 설정됩니다.
• CA를 삭제하는 방법에 대한 자세한 내용은 사설 CA 삭제 (p. 64) 단원을 참조하십시오.

**Note**

삭제됨 및 실패를 제외한 모든 상태 값에 대해 CA에 대한 요금이 청구됩니다.
다음 다이어그램에서는 CA 상태와 관리 작업의 상호 작용으로 CA 수명 주기를 보여 줍니다.
AWS 콘솔을 사용해 CA 상태를 업데이트하는 방법

1. AWS 계정으로 로그인하여 **https://console.aws.amazon.com/acm-pca/home**에서 ACM Private CA 콘솔을 엽니다.
2. 사설 CA를 선택합니다.
3. 목록에서 사설 CA를 선택합니다.
4. 작업 메뉴에서 비활성화를 선택하여 현재 활성 상태인 사설 CA를 비활성화하거나 활성화를 선택하여 CA를 활성 상태로 설정합니다.

AWS CLI을 사용해 사설 CA 상태를 업데이트하는 방법

**update-certificate-authority** 명령을 사용합니다. 다음과 비슷한 파일을 사용하여 CRL 구성을 지정할 수 있습니다.

```json
{
  "CrlConfiguration":
  {
    "Enabled": true,
    "ExpirationInDays": 7,
    "CustomCname": "https://www.somename.crl",
    "S3BucketName": "your-crl-bucket-name"
  }
}
```

아래 명령은 이전 파일을 사용하여 취소를 구성하고 사설 CA의 상태를 **ACTIVE**로 설정합니다.

```
aws acm-pca update-certificate-authority \
  --certificate-authority-arn arn:aws:acm-pca:region:account:\
  certificate-authority/12345678-1234-1234-1234-1232456789012 \
  --revocation-configuration file://C:\revoke_config.txt \
  --status "ACTIVE"
```

**CRL 구성 업데이트**

사설 CA에 대한 인증서 해지 목록(CRL) 구성을 업데이트하여 다음 값을 변경할 수 있습니다.

- 사설 CA가 CRL을 생성하는지 여부
- CRL 만료까지 남은 일수입니다. 지정된 일수의 ½이 되면 ACM Private CA가 CRL 재생성을 시도하기 시작합니다.
- CRL이 저장되는 Amazon S3 버킷의 이름입니다.
- S3 버킷의 이름을 공개적으로 볼 수 없도록 숨기기 위한 별칭입니다.

**Important**

앞서 나온 파라미터를 변경하면 부정적인 결과가 발생할 수 있습니다. 예를 들어 CRL 생성을 비활성화하거나, 유효 기간을 변경하거나, 프로덕션 환경에서 사설 CA를 배치한 이후에 S3 버킷을 변경하면 CRL 및 현재 CRL 구성에 기반한 기존 인증서가 손상될 수 있습니다. 이전 별칭이 올바른 버킷에 계속 연결되어 있긴 하면 안전하게 별칭을 변경할 수 있습니다.
AWS 콘솔을 사용하여 CRL 구성 업데이트하는 방법

2. 사설 CA를 선택합니다.
3. 목록에서 사설 CA를 선택합니다.
4. 작업 메뉴에서 CA 해지 업데이트를 선택합니다.
5. CRL 배포 활성화를 선택하여 인증서 취소 목록(CRL)을 생성합니다.
6. S3 버킷 생성에서 예를 선택하고 고유한 버킷 이름을 입력하거나 아니오를 선택하고 목록에서 기존 버킷을 선택합니다.
7. 사용자 지정 CRL 이름에서 S3 버킷의 이름을 공개적으로 볼 수 없도록 숨기기 위해 별칭을 입력합니다.
8. 유호에 유효 기간 일수를 입력합니다.

사설 CA에 태그 추가

태그는 AWS 리소스를 식별 및 구성하기 위한 메타데이터로 작동하는 단어나 구문입니다. 각 태그는 키와 값으로 구성됩니다. ACM Private CA 콘솔, AWS Command Line Interface(AWS CLI) 또는 PCA API를 사용하여 사설 CA에 대한 태그를 추가, 확인 또는 제거할 수 있습니다.

요구 사항을 충족하는 사용자 지정 태그를 생성할 수 있습니다. 예를 들어 Environment=Prod 또는 Environment=Beta의 키값 쌍을 이용해 사설 CA에 태그를 지정하면 CA가 사용되는 환경을 식별할 수 있습니다. 생성시 그 이후에 언제든지 CA에 태그를 추가할 수 있습니다. 자세한 내용은 사설 CA 생성(p. 51)을 참조하십시오.

다른 AWS 리소스에서도 태그 지정을 지원합니다. 동일한 태그를 다른 리소스에 할당하여 해당 리소스의 관련 여부를 나타낼 수 있습니다. 예를 들어, Website=example.com과 같은 태그를 CA, Elastic Load Balancing 로드 밸런서 및 기타 관련 리소스에 할당할 수 있습니다. AWS 리소스 태그에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 리소스 태깅을 참조하십시오.

ACM Private CA 태그에 적용되는 기본 제한 사항은 다음과 같습니다.

• 사설 CA당 최대 태그 수는 50개입니다.
• 태그 키의 최대 길이는 128자입니다.
• 태그 값의 최대 길이는 256자입니다.
• 태그 키와 값에는 A-Z, a-z 및 :+=@_%-(하이픈)이 포함될 수 있습니다.
• 태그 키와 값은 대소문자를 구분합니다.
• 태그 스키마를 여러 서비스와 리소스에서 사용하려는 경우, 서비스마다 허용되는 문자에 대한 제한이 다를 수 있음을 유의하십시오. 해당 서비스에 대한 문서를 참조하십시오.
• ACM Private CA 태그는 AWS Management 콘솔의 Resource Groups and Tag Editor(리소스 그룹 및 태그 편집기)에서 사용할 수 없습니다.

ACM Private CA 콘솔, AWS Command Line Interface (AWS CLI) 또는 ACM Private CA API에서 사설 CA에 태그를 지정할 수 있습니다.

사설 CA에 태그를 지정하는 방법(콘솔)


버전 latest

63
사설 CA를 선택합니다.
3. 목록에서 사설 CA를 선택합니다.
5. [Edit]를 선택합니다.
6. 키-값 페어를 입력합니다.

사설 CA에 태그를 지정하는 방법(AWS CLI)
tag-certificate-authority 명령을 사용하여 사설 CA에 태그를 추가합니다.

```bash
aws acm-pca tag-certificate-authority \\
--certificate-authority-arn arn:aws:acm-pca:region:account: \\
certificate-authority/12345678-1234-1234-1234-123456789012 \\
--tags Key=Admin,Value=Alice
```

list-tags 명령을 사용하여 사설 CA에 대한 태그를 나열합니다.

```bash
aws acm-pca list-tags \\
--certificate-authority-arn arn:aws:acm-pca:region:account: \\
certificate-authority/123455678-1234-1234-1234-123456789012 \\
--max-results 10
```

untag-certificate-authority 명령을 사용하여 사설 CA에서 태그를 제거합니다.

```bash
aws acm-pca untag-certificate-authority \\
--certificate-authority-arn arn:aws:acm-pca:region:account: \\
certificate-authority/12345678-1234-1234-1234-123456789012 \\
--tags Key=Purpose,Value=Website
```

사설 CA 삭제
사설 CA를 영구적으로 삭제할 수 있습니다. 예를 들어 새로운 프라이빗 키를 가진 신규 CA로 이를 대체하기 위해 삭제를 원할 수 있습니다. CA를 안전하게 삭제하려면 다음과 같이 하십시오.

1. 대체 CA를 생성합니다.
2. 새로운 사설 CA가 생성 중인 경우에는 이전 CA를 비활성화하되, 즉시 이를 삭제해서는 안 됩니다.
3. 발급된 모든 인증서의 기간이 만료될 때까지 이전 CA를 비활성화 상태로 유지합니다.
4. 이전 CA를 삭제합니다.

ACM Private CA는 삭제 요청을 처리하기 전에는 발급된 모든 인증서의 기간이 만료되었는지 확인하지 않습니다. 어떤 인증서가 기간이 만료되었는지 확인할 수 있도록 감사 보고서 (p. 32)를 생성할 수 있습니다. CA가 비활성화된 상태에서는 인증서를 취소할 수 있지만, 새 인증서를 발급할 수 없습니다.

발급된 모든 인증서가 기간 만료되기 전에 사설 CA를 삭제해야 하는 경우에도 CA 인증서를 취소할 수 있습니다. CA 인증서는 상위 CA의 CRL에 나열되며, 사설 CA는 클라이언트가 신뢰하지 않습니다.

**Important**

PENDING_CERTIFICATE, CREATING, EXPIRED, DISABLED 또는 FAILED 상태인 경우 사설 CA를 삭제할 수 있습니다. ACTIVE 상태인 CA를 삭제하려면 먼저 비활성화를 해야 합니다. 그렇지 않으 exercises, which may include theoretical questions, practical exercises, or a combination of both. The exercises are designed to reinforce the concepts discussed in the text and to provide hands-on experience with AWS Certificate Manager Private Certificate Authority.
먼 삭제 요청으로 인해 에러가 발생합니다. PENDING_CERTIFICATE 또는 DISABLED 상태인 사설 CA를 삭제하려는 경우 복원 기간으로 7 ~ 30일을 설정할 수 있으며, 기본값은 30일입니다. 이 기간 동안 상태는 DELETED로 설정되고 CA를 복원할 수 있습니다. CREATING 또는 FAILED 상태에 있는 동안 삭제된 사설 CA에는 복원 기간이 할당되어 있지 않으므로 복원할 수 없습니다. 자세한 내용은 사설 CA 복원 (p. 65) 단원을 참조하십시오.

삭제된 사설 CA에 대해서는 요금이 부과되지 않습니다. 하지만 삭제한 CA를 복원하면 삭제 후 복원할 때까지의 기간에 대해 요금이 부과됩니다. 자세한 내용은 요금 (p. 5) 단원을 참조하십시오.

AWS Management 콘솔 또는 AWS CLI에서 사설 CA를 삭제할 수 있습니다.

사설 CA를 삭제하는 방법(콘솔)

2. 사설 CA를 선택합니다.
3. 목록에서 사설 CA를 선택합니다.
4. CA가 ACTIVE 상태인 경우 CA를 비활성화해야 합니다. 작업 메뉴에서 삭제를 선택합니다.
6. CA가 PENDING_CERTIFICATE, EXPIRED 또는 DISABLED 상태인 경우 복원 기간을 7-30일로 지정합니다. 그런 다음 [Delete]를 선택합니다.

   Note

   사설 CA가 이러한 상태 중 하나에 해당하지 않으면 나중에 복원할 수 없습니다.
7. 사설 CA를 확인한 삭제하기로 한 경우 메시지가 표시되면 영구 삭제를 선택합니다. 사설 CA의 상태가 DELETED로 변경됩니다. 하지만 복원 기간이 끝나기 전에 사설 CA를 복원할 수 없습니다. DELETED 상태인 사설 CA의 복원 기간을 확인하려면 DescribeCertificateAuthority 또는 ListCertificateAuthorities 작업을 호출합니다.

사설 CA를 삭제하는 방법(AWS CLI)

delete-certificate-authority 명령을 사용하여 사설 CA를 삭제합니다.

```
aws acm-pca delete-certificate-authority \
--certificate-authority-arn arn:aws:acm-pca:region:account: \
certificate-authority/12345678-1234-1234-1234-123456789012 \
--permanent-deletion-time-in-days 16
```

사설 CA 복원

CA가 삭제 시 지정한 복원 기간 내에 드는 동안은 삭제된 사설 CA를 복원할 수 있습니다. 복원 기간은 7일 ~ 30일 범위 안에서 일수로 지정하며, 이 기간 동안 사설 CA가 복원 가능한 상태로 유지됩니다. 해당 기간이 끝나면 사설 CA가 영구적으로 삭제됩니다. 자세한 내용은 사설 CA 삭제 (p. 64) 단원을 참조하십시오. 영구적으로 삭제된 사설 CA를 복원할 수 없습니다.

   Note

   삭제된 사설 CA에 대해서는 요금이 부과되지 않습니다. 하지만 삭제한 CA를 복원하면 삭제 후 복원할 때까지의 기간에 대해 요금이 부과됩니다. 자세한 내용은 요금 (p. 5) 단원을 참조하십시오.

주제

• 사설 CA 복원(콘솔) (p. 66)
• 사설 CA 복원(AWS CLI) (p. 66)
사설 CA 복원(콘솔)

AWS Management 콘솔을 사용하여 사설 CA를 복원할 수 있습니다.

사설 CA를 복원하는 방법(콘솔)

2. 사설 CA를 선택합니다.
3. 목록에서 사설 CA를 선택합니다.
4. 현재 상태가 DELETED인 경우 사설 CA를 복원할 수 있습니다. 작업 메뉴에서 복원을 선택합니다.
5. 대화 상자가 나타나면 복원을 다시 선택합니다.
6. 성공적으로 복원되면 사설 CA의 상태가 삭제 전 상태로 설정됩니다. 상태를 ACTIVE로 변경하려면 작업 메뉴에서 활성화를 선택합니다. 사설 CA가 삭제 당시 PENDING_CERTIFICATE 상태였던 경우에는 활성화에 앞서 먼저 CA 인증서를 사설 CA로 가져와야 합니다.

사설 CA 복원(AWS CLI)

restore-certificate-authority 명령을 사용하여 DELETED 상태인 삭제된 사설 CA를 복원할 수 있습니다. 다음 단계에서는 사설 CA를 삭제, 복원 및 재활성화하는 데 필요한 전체 프로세스를 설명합니다.

사설 CA를 삭제, 복원 및 재활성화하는 방법(AWS CLI)

1. 사설 CA를 삭제합니다.
   
   delete-certificate-authority 명령을 실행하여 사설 CA를 삭제합니다. 사설 CA의 상태가 DISABLED 또는 PENDING_CERTIFICATE이면 --permanent-deletion-time-in-days 파라미터를 설정하여 사설 CA의 복원 기간을 7 ~ 30일로 지정할 수 있습니다. 복원 기간을 지정하지 않을 경우 기본값은 30일입니다. 성공할 경우 이 명령은 사설 CA의 상태를 DELETED로 설정합니다.
   
   Note

   복원이 가능하려면 삭제 당시 사설 CA의 상태가 DISABLED 또는 PENDING_CERTIFICATE여야 합니다.

   ```
   aws acm-pca delete-certificate-authority \
   --certificate-authority-arn arn:aws:acm-pca:region:account:\
   certificate-authority/12345678-1234-1234-1234-123456789012 \
   --permanent-deletion-time-in-days 16
   ```

2. 사설 CA를 복원합니다.

   restore-certificate-authority 명령을 실행하여 사설 CA를 복원합니다. delete-certificate-authority 명령을 사용하여 설정한 복원 기간이 만료되기 전에 이 명령을 실행해야 합니다. 성공할 경우 이 명령은 사설 CA의 상태를 삭제 전 상태로 설정합니다.

   ```
   aws acm-pca restore-certificate-authority \
   --certificate-authority-arn arn:aws:acm-pca:region:account:\
   certificate-authority/12345678-1234-1234-1234-123456789012
   ```

3. 사설 CA를 ACTIVE로 설정합니다.

   update-certificate-authority 명령을 실행하여 사설 CA의 상태를 ACTIVE로 변경합니다.

   ```
   aws acm-pca update-certificate-authority \
   --certificate-authority-arn arn:aws:acm-pca:region:account:\
   ```
ACM에 인증서 갱신 권한 할당

AWS Certificate Manager(ACM)의 관리형 갱신을 사용하면 공용 및 사설 CA 모두에서 인증서 갱신 프로세스를 자동화할 수 있습니다. ACM이 사설 CA에서 생성된 인증서를 자동으로 갱신할 수 있도록 ACM 서비스 보안 주체에게 CA 자체에서 가능한 모든 권한을 부여해야 합니다. CA의 소유자(또는 공인 대리인)가 각 사설 인증서를 수동으로 갱신해야 합니다.

갱신 권한은 사설 CA를 생성(p. 51)하는 동안 위임을 하거나, CA가 ACTIVE 상태에 있는 한 이후에 언제든지 변경할 수 있습니다.

ACM Private CA 콘솔, AWS Command Line Interface(AWS CLI) 또는 ACM Private CA API에서 사설 CA 권한을 관리할 수 있습니다.

사설 CA 권한을 할당하는 방법(콘솔)
2. 사설 CA를 선택합니다.
3. 목록에서 사설 CA를 선택합니다.
4. 권한 탭을 선택합니다.
5. ACM이 갱신을 위해 이 CA를 사용하도록 승인을 선택합니다.
6. 저장을 선택합니다.

ACM Private CA에서 권한을 관리하는 방법(AWS CLI)
create-permission 명령을 사용하여 ACM에 권한을 할당합니다. ACM에서 인증서를 자동으로 갱신하려면 가능한 모든 권한(IssueCertificate, GetCertificate 및 ListPermissions)을 할당해야 합니다.

```bash
aws acm-pca create-permission \
--certificate-authority-arn arn:aws:acm-pca:region:account:12345678-1234-1234-1234-123456789012 \
--actions IssueCertificate, GetCertificate, ListPermissions \
--principal acm.amazonaws.com
```

list-permissions 명령을 사용하여 CA에서 위임한 권한을 나열합니다.

```bash
aws acm-pca list-permissions \
```

delete-permission 명령을 사용하여 CA에서 AWS 서비스 보안 주체에 할당한 사용 권한을 해지합니다.

```bash
aws acm-pca delete-permission \
--certificate-authority-arn arn:aws:acm-pca:region:account:12345678-1234-1234-1234-123456789012 \
--principal acm.amazonaws.com
```
최종 엔터티 인증서 발급 및 해지

사설 CA를 생성한 후에는 이를 사용하여 다음 작업을 수행할 수 있습니다.

주제
• 사설 최종 엔터티 인증서 발급 (p. 68)
• 사설 인증서 해지 (p. 69)
• 인증서 템플릿 (p. 71)

사설 최종 엔터티 인증서 발급

AWS Certificate Manager(ACM) 또는 ACM Private CA 중 하나에서 사설 최종 엔터티 인증서를 생성할 수 있습니다. 두의 기능은 다음 표에서 비교가 되어 있습니다.

<table>
<thead>
<tr>
<th>기능</th>
<th>ACM</th>
<th>ACM Private CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>사설 CA 인증서 생성</td>
<td>지원되지 않음</td>
<td>✓ (IssueCertificate 사용)</td>
</tr>
<tr>
<td>인증서는 지원되는 인터넷 연결 AWS 서비스와 통합</td>
<td>✓</td>
<td>지원되지 않음</td>
</tr>
<tr>
<td>사설 CA 인증서 가져오기</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>최종 엔터티 인증서 생성</td>
<td>✓ (RequestCertificate와 GetCertificate를 함께 사용)</td>
<td>✓ (IssueCertificate 사용)</td>
</tr>
<tr>
<td>콘솔 지원</td>
<td>✓</td>
<td>✓ (CA 인증서만)</td>
</tr>
<tr>
<td>API 지원</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CLI 지원</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>사용자 구성 가능한 템플릿</td>
<td>지원되지 않음</td>
<td>✓</td>
</tr>
</tbody>
</table>

따라서 ACM Private CA API 및 AWS CLI는 모든 인증서 유형을 발급할 수 있습니다. ACM Private CA 콘솔에서 CA 인증서를 발급할 수 있습니다. 또한 ACM 서비스는 콘솔, API 또는 AWS CLI를 사용하여 최종 엔터티 인증서를 발급할 수 있습니다. ACM 사용에 대한 자세한 내용은 사설 인증서 요청을 참조하십시오.

ACM Private CA에서 인증서를 생성하는 경우, 인증서 유형과 경로 길이를 지정하는 사용자 정의 가능한 템플릿을 적용합니다. 인증서를 생성하는 API 또는 CLI 문에서 템플릿 ARN이 제공되지 않으면 최종 엔터티 템플릿이 기본적으로 적용됩니다. 사용 가능한 인증서 템플릿에 대한 자세한 내용은 인증서 템플릿 (p. 71) 단원을 참조하십시오.

ACM Private CA API 또는 AWS CLI에서 발급되는 사설 인증서는 ACM에서 발급되는 사설 인증서와 동일한 제한 사항이 적용되지 않습니다. ACM Private CA를 사용하여 다음을 수행할 수 있습니다.

• 어떤 주체 이름이든 인증서를 생성합니다.
• 지원되는 프라이빗 키 알고리즘과 키 길이는 무엇이든 사용할 수 있습니다.
• 현재 지원되는 서명 알고리즘은 무엇이든 사용할 수 있습니다.
• 사설 CA 및 사설 인증서에 대한 유효 기간을 설정합니다.
• ACM 및 IAM에 사설 인증서를 가져옵니다.
인증서 발급(AWS CLI)

`issue-certificate` 명령을 사용하여 사설 인증서를 요청할 수 있습니다. 이 명령은 인증서 발급에 사용하고 싶은 사설 CA의 Amazon 리소스 이름(ARN)을 요구합니다. 또한 발급하려는 인증서에 대한 인증서 서명 요청(CSR)이 필요합니다. 또한 IssueCertificate 작업을 사용할 수도 있습니다.

**Note**

ACM Private CA API 또는 AWS CLI를 사용해 사설 인증서를 발급합니다. 하지만 이럴 경우, ACM 콘솔, CLI 또는 API를 사용하여 인증서를 보거나 내보낼 수 없습니다. CSR을 생성하기 위해 반드시 필요하다는 점에서 프라이빗 키를 이미 가지고 있습니다(앞의 내용 참조). `get-certificate` 명령을 사용하여 인증서 세부 정보를 검색할 수 있습니다. 또한 감사 보고서(p. 32)를 생성하여 인증서가 발급되었는지 확인할 수 있습니다.

```bash
aws acm-pca issue-certificate \
--certificate-authority-arn arn:aws:acm-pca:region:account: \
 certificate-authority/12345678-1234-1234-1234-123456789012 \
 --csr file://C:\cert_1.csr \
 --signing-algorithm "SHA256WITHRSA" \
 --validity Value=365,Type="DAYS" \
 --idempotency-token 1234
```

이 명령은 발급된 인증서의 ARN을 출력합니다.

```json
{
 "CertificateArn": "arn:aws:acm-pca:region:account:
 certificate-authority/12345678-1234-1234-1234-123456789012/
 certificate/a2b95975ec6e1cd85d21c2104c5be129"
}
```

사설 인증서 해지

AWS CLI 또는 ACM Private CA API를 사용하여 발급된 인증서를 해지할 수 있습니다. 인증서는 사설 발급 CA의 인증서 취소 목록(CRL)에 포함이 됩니다. 취소된 인증서는 항상 감사 보고서에 기록됩니다.

주제

- CRL의 해지된 인증서 (p. 69)
- 감사 보고서에서 해지된 인증서 (p. 70)
- AWS CLI를 사용하여 인증서 해지 (p. 71)

CRL의 해지된 인증서

다음 예제에서는 인증서 취소 목록(CRL)에서 취소된 인증서를 보여줍니다. 보통은 인증서가 취소되고 약 30분 후에 CRL이 업데이트됩니다. 어떤 이유로든 CRL 업데이트가 실패하면 ACM PCA는 15분마다 업데이트를 추가로 시도합니다. Amazon CloudWatch를 사용하면 CRLGenerated 및 MisconfiguredCRLBucket 지표에 대한 경보를 생성할 수 있습니다. 자세한 내용은 지원되는 CloudWatch 지표를 참조하십시오. CRL 생성 및 구성에 대한 자세한 내용은 ??? (p. 52) 단원을 참조하십시오.
감사 보고서에서 해지된 인증서

취소된 인증서를 포함하여 모든 인증서는 사설 CA의 감사 보고서에 포함됩니다. 다음 예제에서는 하나의 인증서가 발급되고 하나의 인증서가 취소된 감사 보고서를 보여줍니다. 자세한 내용은 사설 CA에 대한 감사 보고서 생성 (p. 32) 단원을 참조하십시오.

```json
[{  
awsAccountId: "123456789012",  
subject: "1.2.840.113549.1.9.1=#161173616c6573406578616d706c652e636f6d,\nCN=www.example1.com,OU=Sales,O=Example Company,L=Seattle,ST=Washington,C=US",  
notBefore: "2018-02-26T18:39:57+0000",  
notAfter: "2019-02-26T19:39:57+0000",  
issuedAt: "2018-02-26T19:39:57+0000",  
revokedAt: "2018-02-26T19:39:57+0000",  
revocationReason: "KEY_COMPROMISE"
},
{  
awsAccountId: "123456789012",  
certificateArn: "arn:aws:acm-pca:region:account:certificate-authority/CA_ID/  certificate/2bae9a75d71b42bed4e1e36f8b4b488fc",  
subject: "1.2.840.113549.1.9.1=#161173616c6573406578616d706c652e636f6d,\nCN=www.example1.com,OU=Sales,O=Example Company,L=Seattle,ST=Washington,C=US",  
notBefore: "2018-02-26T18:39:57+0000",  
notAfter: "2019-02-26T19:39:57+0000",  
issuedAt: "2018-02-26T19:39:57+0000",  
revokedAt: "2018-02-26T19:39:57+0000",  
revocationReason: "KEY_COMPROMISE"
}]
```
**AWS CLI를 사용하여 인증서 해지**

`revoke-certificate` 명령을 사용하여 사설 인증서를 해지합니다. 또한 `RevokeCertificate` 함수도 사용할 수 있습니다. 일련 번호는 16진수 형식이어야 합니다. `get-certificate` 명령을 호출하여 이 일련 번호를 검색할 수 있습니다. `revoke-certificate` 명령은 응답을 반환하지 않습니다.

```bash
aws acm-pca revoke-certificate \
  --certificate-authority-arn arn:aws:acm-pca:region:account: \
  certificate-authority/12345678-1234-1234-1234-123456789012 \
  --revoication-reason "KEY_COMPROMISE"
```

### 인증서 템플릿

ACM Private CA는 템플릿을 사용하여 CA 인증서와 사용자, 호스트, 리소스 및 디바이스를 식별하는 최종 엔터티 인증서를 모두 생성합니다. 콘솔에서 인증서를 생성하면 템플릿이 자동으로 적용됩니다. 적용되는 템플릿은 선택한 인증서 유형과 지정한 경로 길이에 기반으로 합니다. CLI 또는 API를 사용하여 인증서를 생성하는 경우 적용할 템플릿의 ARN을 직접 입력합니다 (ARN을 입력하지 않으면 `EndEntityCertificate` 템플릿이 적용).

다음 링크는 ACM Private CA를 사용하여 인증서를 생성하는 방법에 대한 정보를 제공합니다.

- 콘솔 사용: 사설 인증서 발급.
- 명령줄 사용: `issue-certificate`.
- API 사용: `IssueCertificate`.

아래 표에는 ACM Private CA에서 지원하는 템플릿 유형과 해당 정의에 대한 링크가 나와 있습니다.

<table>
<thead>
<tr>
<th>템플릿 이름</th>
<th>템플릿 ARN</th>
<th>인증서 유형</th>
<th>경로 길이</th>
</tr>
</thead>
<tbody>
<tr>
<td>EndEntityCertificate/ V1 (p. 73)</td>
<td>arn:aws:acm-pca:::template/EndEntityCertificate/ V1</td>
<td>최종 엔터티</td>
<td>해당 사항 없음</td>
</tr>
<tr>
<td>EndEntityCertificate_CSRPassthrough/ V1 (p. 74)</td>
<td>arn:aws:acm-pca:::template/EndEntityCertificate_CSRPassthrough/ V1</td>
<td>End-entity/ passthrough</td>
<td>해당 사항 없음</td>
</tr>
<tr>
<td>EndEntityClientAuthCertificate/ V1 (p. 74)</td>
<td>arn:aws:acm-pca:::template/EndEntityClientAuthCertificate/ V1</td>
<td>최종 엔터티</td>
<td>해당 사항 없음</td>
</tr>
<tr>
<td>템플릿 이름</td>
<td>템플릿 ARN</td>
<td>인증서 유형</td>
<td>경로 길이</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>EndEntityClientAuthCertificate_CSRPassthrough/ V1 (p. 75)</td>
<td>arn:aws:acm-pca:::template/EndEntityClientAuthCertificate_CSRPassthrough/ V1</td>
<td>End-entity/ passthrough</td>
<td>해당 사항 없음</td>
</tr>
<tr>
<td>EndEntityServerAuthCertificate/ V1 (p. 75)</td>
<td>arn:aws:acm-pca:::template/EndEntityServerAuthCertificate/ V1</td>
<td>최종 엔터티</td>
<td>해당 사항 없음</td>
</tr>
<tr>
<td>OCSPSigningCertificate/ V1 (p. 76)</td>
<td>arn:aws:acm-pca:::template/OCSPSigningCertificate/ V1</td>
<td>OCSP 서명</td>
<td>해당 사항 없음</td>
</tr>
<tr>
<td>OCSPSigningCertificate_CSRPassthrough/ V1 (p. 76)</td>
<td>arn:aws:acm-pca:::template/OCSPSigningCertificate_CSRPassthrough/ V1</td>
<td>OCSP 서명</td>
<td>해당 사항 없음</td>
</tr>
<tr>
<td>RootCACertificate/ V1 (p. 77)</td>
<td>arn:aws:acm-pca:::template/RootCACertificate/ V1</td>
<td>CA</td>
<td>무제약</td>
</tr>
<tr>
<td>SubordinateCACertificate_PathLen0/ V1 (p. 77)</td>
<td>arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/ V1</td>
<td>CA</td>
<td>0</td>
</tr>
<tr>
<td>SubordinateCACertificate_PathLen1/ V1 (p. 78)</td>
<td>arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen1/ V1</td>
<td>CA</td>
<td>1</td>
</tr>
<tr>
<td>SubordinateCACertificate_PathLen2/ V1 (p. 78)</td>
<td>arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen2/ V1</td>
<td>CA</td>
<td>2</td>
</tr>
</tbody>
</table>

AWS Certificate Manager Private
Certificate Authority 사용 설명서
CodeSigningCertificate/V1 정의

**CodeSigningCertificate/V1 정의**

이 템플릿은 코드 서명에 대한 인증서를 생성하는 데 사용됩니다. 사설 CA 인프라를 기반으로 하는 모든
코드 서명 솔루션에서는 ACM Private CA에서 코드 서명 인증서를 사용할 수 있습니다. 예를 들어 AWS
IoT에 대한 코드 서명을 사용하는 고객은 ACM Private CA를 사용하여 코드 서명 인증서를 생성하고 AWS
Certificate Manager으로 가져올 수 있습니다. 자세한 내용은 AWS IoT용 코드 서명이란 무엇입니까? 및 코드
서명 인증서 획득 및 가져오기를 참조하십시오.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>CA:FALSE</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>심각, 디지털 서명</td>
</tr>
<tr>
<td>X509v3 확장 키 사용</td>
<td>심각, 코드 서명</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

*CRL 배포 지점은 CA가 CRL 생성을 사용하도록 구성된 경우에만 템플릿에 포함됩니다.

**CodeSigningCertificate_CSRPassthrough/V1 정의**

이 템플릿은 한 가지 차이점을 제외하고 CodeSigningCertificate 템플릿과 동일합니다. 이 템플릿에서
는 확장이 템플릿에 지정되지 않은 경우에 ACM Private CA가 인증서 서명 요청(CSR)에서 인증서로 추가 확
장을 전달합니다. 템플릿에 지정된 확장은 항상 CSR의 확장보다 우선합니다.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>CA:FALSE</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>심각, 디지털 서명</td>
</tr>
<tr>
<td>X509v3 확장 키 사용</td>
<td>심각, 코드 서명</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

*CRL 배포 지점은 CA가 CRL 생성을 사용하도록 구성된 경우에만 템플릿에 포함됩니다.

**EndEntityCertificate/V1 정의**

이 템플릿은 운영 체제 또는 웹 서버와 같은 최종 엔터티에 대한 인증서를 생성하는 데 사용됩니다.
## EndEntityCertificate_CSRPassthrough/V1 정의

이 템플릿은 한 가지 차이점을 제외하고 EndEntityCertificate 템플릿과 동일합니다. 이 템플릿에서는 확장이 템플릿에 지정되지 않은 경우에 ACM Private CA가 인증서 서명 요청(CSR)에서 인증서로 추가 확장을 전달합니다. 템플릿에 지정된 확장은 항상 CSR의 확장보다 우선합니다.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>CA:FALSE</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>심각, 디지털 서명, 키 암호화</td>
</tr>
<tr>
<td>X509v3 확장 키 사용</td>
<td>TLS 웹 서버 인증, TLS 웹 클라이언트 인증</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

*CRL 배포 지점은 CA가 CRL 생성을 사용하도록 구성된 경우에만 템플릿에 포함됩니다.

## EndEntityClientAuthCertificate/V1 정의

이 템플릿은 TLS 웹 클라이언트 인증으로 제한하는 확장 키 사용 값에서만 EndEntityCertificate와 다릅니다.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>CA:FALSE</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
</tbody>
</table>

*CRL 배포 지점은 CA가 CRL 생성을 사용하도록 구성된 경우에만 템플릿에 포함됩니다.
AWS Certificate Manager Private Certificate Authority 사용 설명서
EndEntityClientAuthCertificate_CSRPassthrough/V1 정의

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 키 사용</td>
<td>심각, 디지털 서명, 키 암호화</td>
</tr>
<tr>
<td>X509v3 확장 키 사용</td>
<td>TLS 웹 클라이언트 인증</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

* CRL 배포 지점은 CA가 CRL 생성을 사용하도록 구성된 경우에만 템플릿에 포함됩니다.

**EndEntityClientAuthCertificate_CSRPassthrough/V1 정의**

이 템플릿은 한 가지 차이점을 제외하고 EndEntityClientAuthCertificate 템플릿과 동일합니다. 이 템플릿에서는 확장이 템플릿에 지정되지 않은 경우에 ACM Private CA가 인증서 서명 요청(CSR)에서 인증 서로 추가 확장을 전달합니다. 템플릿에 지정된 확장은 항상 CSR의 확장보다 우선합니다.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>CA:FALSE</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>심각, 디지털 서명, 키 암호화</td>
</tr>
<tr>
<td>X509v3 확장 키 사용</td>
<td>TLS 웹 클라이언트 인증</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

* CRL 배포 지점은 CA가 CRL 생성을 사용하도록 구성된 경우에만 템플릿에 포함됩니다.

**EndEntityServerAuthCertificate/V1 정의**

이 템플릿은 TLS 웹 서버 인증으로 제한하는 확장 키 사용 값에서만 EndEntityCertificate과 다릅니다.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>CA:FALSE</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>심각, 디지털 서명, 키 암호화</td>
</tr>
<tr>
<td>X509v3 확장 키 사용</td>
<td>TLS 웹 서버 인증</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>
EndEntityServerAuthCertificate_CSRPassthrough/V1 정의

이 템플릿은 한 가지 차이점만을 제외하고 EndEntityServerAuthCertificate 템플릿과 동일합니다. 이 템플릿에서는 확장이 템플릿에 지정되지 않은 경우에 ACM Private CA가 인증서 서명 요청(CSR)에서 인증 서로 추가 확장을 전달합니다. 템플릿에 지정된 확장은 항상 CSR의 확장보다 우선합니다.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>CA:FALSE</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>심각, 디지털 서명, 키 암호화</td>
</tr>
<tr>
<td>X509v3 확장 키 사용</td>
<td>TLS 웹 서버 인증</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

OCSPSigningCertificate/V1 정의

이 템플릿은 OCSP 응답에 서명하기 위한 인증서를 생성하는 데 사용됩니다. 확장 키 사용 값이 코드 서명 대신 OCSP 서명을 지정한다는 점을 제외하면 CodeSigningCertificate 템플릿과 동일합니다.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>CA:FALSE</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>심각, 디지털 서명</td>
</tr>
<tr>
<td>X509v3 확장 키 사용</td>
<td>심각, OCSP 서명</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

OCSPSigningCertificate_CSRPassthrough/V1 정의

이 템플릿은 한 가지 차이점만을 제외하고 OCSPSigningCertificate 템플릿과 동일합니다. 이 템플릿에서 는 확장이 템플릿에 지정되지 않은 경우에 ACM Private CA가 인증서 서명 요청(CSR)에서 인증서로 추가 확장을 전달합니다. 템플릿에 지정된 확장은 항상 CSR의 확장보다 우선합니다.
AWS Certificate Manager Private
Certificate Authority 사용 설명서
RootCACertificate/V1 정의

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>CA:FALSE</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>심각, 디지털 서명</td>
</tr>
<tr>
<td>X509v3 확장 키 사용</td>
<td>심각, OCSP 서명</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

*CRL 배포 지점은 CA가 CRL 생성을 사용하도록 구성된 경우에만 템플릿에 포함됩니다.

RootCACertificate/V1 정의

이 템플릿은 자체 서명된 루트 CA 인증서를 발급하는 데 사용됩니다. CA 인증서에는 CA 인증서를 발급하는 데 사용할 수 있음을 지정하도록 CA 필드가 TRUE로 설정된 중요한 기본 제약 조건 확장이 포함됩니다. 경로 길이가 CA 체인의 최대 길이(CA 인증 길이)를 제한하기 때문에 이 템플릿은 경로 길이를 지정하지 않습니 다. 체인 길이를 제한하면 계층의 항후 확장을 막을 수 있습니다. CA 인증서가 TLS 클라이언트 또는 서버 인증 서로 사용되는 막기 위해 확장 키 사용은 제외됩니다. 자체 서명된 인증서를 해지할 수 없으므로 CRL 정보가 지정되지 않았습니다.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>심각, CA:TRUE</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>심각, 디지털 서명, keyCertSign, CRL 서명</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점</td>
<td>해당 사항 없음</td>
</tr>
</tbody>
</table>

SubordinateCACertificate_PathLen0/V1 정의

이 템플릿은 경로 길이가 0인 하위 CA 인증서를 발급하는 데 사용됩니다. CA 인증서에는 CA 인증서를 발급하는 데 사용할 수 있음을 지정하도록 CA 필드가 TRUE로 설정된 중요한 기본 제약 조건 확장이 포함됩니다. 확장 키 사용은 포함되지 않으므로 CA 인증서가 TLS 클라이언트 또는 서버 인증서로 사용되지 않습니다. 인증 경로에 대한 자세한 내용은 인증 경로에서 길이 제약 조건 설정을 참조하십시오.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>심각, CA:TRUE,pathlen= 0</td>
</tr>
</tbody>
</table>
### SubordinateCACertificate_PathLen1/V1 정의

이 템플릿은 경로 길이가 1인 하위 CA 인증서를 발급하는 데 사용됩니다. CA 인증서에는 CA 인증서를 발급하는 데 사용할 수 있음을 지정하도록 CA 필드가 TRUE로 설정된 중요한 기본 제약 조건 확장이 포함됩니다. 확장 키 사용은 포함되지 않으므로 CA 인증서가 TLS 클라이언트 또는 서버 인증서로 사용되지 않습니다.

인증 경로에 대한 자세한 내용은 인증 경로에서 길이 제약 조건 설정을 참조하십시오.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>중요, 디지털 서명, keyCertSign, CRL 서명</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

*CRL 배포 지점은 CA가 CRL 생성을 사용하도록 구성된 경우에만 이 템플릿으로 발급된 인증서에 포함됩니다.

### SubordinateCACertificate_PathLen2/V1 정의

이 템플릿은 경로 길이가 2인 하위 CA 인증서를 발급하는 데 사용됩니다. CA 인증서에는 CA 인증서를 발급하는 데 사용할 수 있음을 지정하도록 CA 필드가 TRUE로 설정된 중요한 기본 제약 조건 확장이 포함됩니다. 확장 키 사용은 포함되지 않으므로 CA 인증서가 TLS 클라이언트 또는 서버 인증서로 사용되지 않습니다.

인증 경로에 대한 자세한 내용은 인증 경로에서 길이 제약 조건 설정을 참조하십시오.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>심각, CA:TRUE, pathlen: 2</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>중요, 디지털 서명, keyCertSign, CRL 서명</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

*CRL 배포 지점은 CA가 CRL 생성을 사용하도록 구성된 경우에만 이 템플릿으로 발급된 인증서에 포함됩니다.
SubordinateCACertificate_PathLen3/V1 정의

이 템플릿은 경로 길이가 3인 하위 CA 인증서를 발급하는 데 사용됩니다. CA 인증서에는 CA 인증서를 발급하는 데 사용할 수 있음을 지정하도록 CA 필드가 TRUE로 설정된 중요한 기본 제약 조건 확장이 포함됩니다. 확장 키 사용은 포함되지 않으므로 CA 인증서가 TLS 클라이언트 또는 서버 인증서로 사용되지 않습니다.

인증 경로에 대한 자세한 내용은 인증 경로에서 길이 제약 조건 설정을 참조하십시오.

<table>
<thead>
<tr>
<th>파라미터</th>
<th>값</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509v3 제목 대체 이름</td>
<td>[CSR에서 복사]</td>
</tr>
<tr>
<td>X509v3 기본 제약 조건</td>
<td>심각, CA:TRUE, pathlen: 3</td>
</tr>
<tr>
<td>X509v3 인증 기관 키 식별자</td>
<td>[AKI]</td>
</tr>
<tr>
<td>X509v3 제목 키 식별자</td>
<td>[SKI]</td>
</tr>
<tr>
<td>X509v3 키 사용</td>
<td>중요, 디지털 서명, keyCertSign, CRL 서명</td>
</tr>
<tr>
<td>X509v3 CRL 배포 지점*</td>
<td>[CRL 배포 지점]</td>
</tr>
</tbody>
</table>

*CRL 배포 지점은 CA가 CRL 생성을 사용하도록 구성된 경우에만 이 템플릿으로 발급된 인증서에 포함됩니다.*
ACM Private CA API 사용 (Java 예제)

AWS Certificate Manager Private Certificate Authority API를 사용하면 HTTP 요청을 전송하여 서비스와 프로그래밍 방식으로 상호 작용할 수 있습니다. 서비스는 HTTP 응답을 반환합니다. 자세한 정보는 AWS Certificate Manager Private Certificate Authority API Reference 단원을 참조하십시오.

HTTP API 외에도 AWS SDK 및 명령줄 도구를 사용하여 ACM Private CA와 상호 작용할 수 있습니다. HTTP API보다는 이 방법을 사용할 것이 권장하고 있습니다. 자세한 내용은 Amazon Web Services용 도구 단원을 참조하십시오. 다음 주제에서는 AWS SDK for Java를 사용하여 ACM PCA API를 프로그래밍하는 방법을 보여 줍니다.

주제
• CreateCertificateAuthority (p. 80)
• CreateCertificateAuthorityAuditReport (p. 83)
• CreatePermission (p. 85)
• DeleteCertificateAuthority (p. 87)
• DeletePermission (p. 88)
• DescribeCertificateAuthority (p. 90)
• DescribeCertificateAuthorityAuditReport (p. 91)
• GetCertificate (p. 93)
• GetCertificateAuthorityCertificate (p. 95)
• GetCertificateAuthorityCsr (p. 96)
• ImportCertificateAuthorityCertificate (p. 98)
• IssueCertificate (p. 100)
• ListCertificateAuthorities (p. 102)
• ListPermissions (p. 105)
• ListTags (p. 106)
• RestoreCertificateAuthority (p. 108)
• RevokeCertificate (p. 109)
• TagCertificateAuthorities (p. 111)
• UntagCertificateAuthority (p. 112)
• UpdateCertificateAuthority (p. 114)

CreateCertificateAuthority

다음 Java 샘플은 CreateCertificateAuthority 작업을 사용하는 방법을 보여줍니다.
이 작업은 사설 하위 인증 기관(CA)을 생성합니다. CA 구성, 취소 구성, CA 유형 및 역동성 토큰 옵션을 지정해야 합니다.

CA 구성에서 다음을 지정합니다.

- CA 프라이빗 키를 생성하는 데 사용할 알고리즘 및 키 크기의 이름
- CA가 서명에 사용하는 서명 알고리즘의 유형
- X.500 주체 정보

CRL 구성에서 다음을 지정합니다.

- CRL 만료 기간 일수(CRL의 유효 기간)
- CRL이 포함되는 Amazon S3 버킷
- CA가 발급하는 인증서에 포함된 S3 버킷의 CNAME 별칭

이 함수가 성공적으로 수행되면 CA의 Amazon 리소스 이름(ARN)이 반환됩니다.

```java
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.AWSACMPCAClientBuilder;
import com.amazonaws.services.acmpca.ASN1Subject;
import com.amazonaws.services.acmpca.CertificateAuthorityConfiguration;
import com.amazonaws.services.acmpca.CertificateAuthorityType;
import com.amazonaws.services.acmpca.CreateCertificateAuthorityResult;
import com.amazonaws.services.acmpca.CreateCertificateAuthorityRequest;
import com.amazonaws.services.acmpca.CrlConfiguration;
import com.amazonaws.services.acmpca.KeyAlgorithm;
import com.amazonaws.services.acmpca.SigningAlgorithm;
import com.amazonaws.services.acmpca.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.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. " +
```

버전 latest
81
"Please make sure that your credentials file is at the correct " + "location (C:\\Users\\joneps\\.aws\\.credentials), and is in valid
format.",
);
}

// Define the endpoint for your sample.
String endpointProtocol = "acm-pca.us-west-2.amazonaws.com";
String endpointRegion = "us-west-2";
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("Amazon");
subject.setOrganizationalUnit("AWS");
subject.setCountry("US");
subject.setState("Washington");
subject.setLocality("Seattle");
subject.setCommonName("www.amazon.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
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");
CreateCertificateAuthorityAuditReport
다음과 유사하게 출력되어야 합니다:

```java
```

CreateCertificateAuthorityAuditReport
다음 Java 샘플은 CreateCertificateAuthorityAuditReport 작업을 사용하는 방법을 보여줍니다.
이 작업은 인증서가 발급 또는 해지될 때마다 나열되는 감사 보고서를 생성합니다. 이 보고서는 입력 시 지정한 Amazon S3 버킷에 저장됩니다. 30분에 한 번씩 새 보고서를 생성할 수 있습니다.

```java
package com.amazonaws.samples;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
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";
        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();

        req.setCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
                "certificate-authority/12345678-1234-1234-1234-123456789012");

        // 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;
        }
    }
}
다음과 유사하게 출력되어야 합니다:

58904752-7de3-4bdf-ba89-6953e48c3cc7
audit-report/16075838-061c-4f7a-b54b-49bb11b1c1f/58904752-7de3-4bdf-ba89-6953e48c3cc7.json

CreatePermission

다음 Java 샘플에서는 CreatePermission 작업을 사용하는 방법을 보여 줍니다.


권한이 생성되면 ListPermissions 함수를 사용하여 해당 권한을 검사하거나 DeletePermissions 함수를 사용하여 권한을 삭제할 수 있습니다.

```java
public static void main(String[] args) throws Exception {
    catch(InvalidStateException ex) {
        throw ex;
    }
    String ID = result.getAuditReportId();
    String S3Key = result.getS3Key();
    System.out.println(ID);
    System.out.println(S3Key);
}
```

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;
public class CreatePermission {
    public static void main(String[] args) throws Exception {
        버전 latest
        85
// 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";
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:region:account:" +
    "certificate-authority/12345678-1234-1234-1234-123456789012");

// Set the permissions to give the user.
req.setActions(["IssueCertificate", "GetCertificate", "ListPermissions"]);

// Set the AWS principal.
req.setPrincipal("acm.amazonaws.com");

// Create a result object.
CreatePermissionsResult 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

다음 Java 샘플은 DeleteCertificateAuthority 작업을 사용하는 방법을 보여줍니다.

이 작업은 CreateCertificateAuthority 작업을 사용하여 생성한 사설 CA(인증 기관)를 삭제합니다. DeleteCertificateAuthority 작업을 수행하려면 삭제할 CA의 ARN을 제공해야 합니다. ListCertificateAuthorities 작업을 호출하여 ARN을 검색할 수 있습니다. 상태가 CREATING 또는 PENDING_CERTIFICATE인 사설 CA는 즉시 삭제할 수 있습니다. 그러나 이미 인증서를 가져왔으면 즉시 삭제할 수 없습니다. 먼저 UpdateCertificateAuthority 작업을 호출하고 Status 파라미터를 DISABLED로 설정하여 CA를 비활성화해야 합니다. 그런 다음 다음 DeleteCertificateAuthority 작업에서 PermanentDeletionTimeInDays 파라미터를 사용하여 7 ~ 30까지 일 수를 지정할 수 있습니다. 이 기간 동안 사설 CA를 disabled 상태로 복원할 수 있습니다. PermanentDeletionTimeInDays 파라미터를 설정하지 않을 경우 기본 복원 기간은 30일입니다. 이 기간이 지나면 사설 CA는 영구적으로 삭제되고 복원할 수 없습니다. 자세한 내용은 CA 복원 (p. 65) 단원을 참조하십시오.

RestoreCertificateAuthority 작업의 사용 방법을 보여주는 Java 샘플은 RestoreCertificateAuthority (p. 108)을 참조하십시오.

```java
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";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);
```
// Create a client that you can use to make requests.
AWSACMPCA client = AWSACMPCAClientBuilder.standard()
    .withEndpointConfiguration(endpoint)
    .withCredentials(new AWSStaticCredentialsProvider(credentials))
    .build();

// Create a request object and set the ARN of the private CA to delete.
DeleteCertificateAuthorityRequest req = new DeleteCertificateAuthorityRequest();

// Set the certificate ARN.
req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
    "certificate-authority/1111111-2222-3333-4444-555555555555");

// 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;
}
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";
        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:region:account:" +
            "certificate-authority/12345678-1234-1234-1234-123456789012");

        // Set the AWS service principal.
        req.setPrincipal("acm.amazonaws.com");

        // Create a result object.
        DeletePermissionsResult result = null;
        try {
            result = client.createPermission(req);
        }
        catch(InvalidArnException ex) {
            throw ex;
        }
        catch(InvalidStateException ex) {
            throw ex;
        }
        catch(RequestFailedException ex) {
            throw ex;
        }
        catch(ResourceNotFoundException ex) {
            throw ex;
        }
    }
}
DescribeCertificateAuthority

다음 Java 샘플은 DescribeCertificateAuthority 작업을 사용하는 방법을 보여줍니다.

이 작업은 사설 인증 기관(CA)에 대한 정보를 나열합니다. 사설 CA의 Amazon 리소스 이름(ARN)을 지정해야 합니다. 출력에는 CA의 상태가 포함되어 있습니다. 이것은 다음 중 하나가 될 수 있습니다.

- CREATING – ACM Private CA가 사설 인증 기관을 생성 중입니다.
- PENDING_CERTIFICATE – 인증서가 잘못되었습니다. 온프레미스 루트 또는 하위 CA를 사용해 사설 CA CSR을 서명한 다음, 이를 PCA에 가져와야 합니다.
- ACTIVE – 사설 CA가 활성 상태입니다.
- DISABLED – 사설 CA가 비활성화되었습니다.
- EXPIRED – 사설 CA 인증서가 기간이 만료되었습니다.
- FAILED – 사설 CA를 생성할 수 없습니다.
- DELETED – 사설 CA가 복원 기간 이내이며, 이 기간이 경과되면 영구 삭제됩니다.

```
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 endpointProtocol = "https://acm-pca.region.amazonaws.com/");
        String endpointRegion = "region");
        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.
DescribeCertificateAuthorityAuditReport req = new DescribeCertificateAuthorityAuditReport();
req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account:"+
  "certificate-authority/12345678-1234-1234-1234-123456789012");

// Create a result object.
DescribeCertificateAuthorityAuditReportResult result = null;
try {
    result = client.describeCertificateAuthorityAuditReport(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);

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;
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";
        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:region:account:" +
            "certificate-authority/12345678-1234-1234-1234-123456789012");

        // 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);
    }
}
다음과 유사하게 출력되어야 합니다:

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

다음 Java 샘플은 GetCertificate 작업을 사용하는 방법을 보여줍니다.


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;
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";
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/12345678-1234-1234-1234-123456789012" +
    "/certificate/793f0d5b6a04125e2c9cfb52373598fe");

// Set the certificate authority ARN.
req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
    "certificate-authority/12345678-1234-1234-1234-123456789012");

// 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);
}

지정한 CA 및 인증서에 대해 다음과 비슷한 인증서 체인이 출력됩니다.

-----BEGIN CERTIFICATE----- base64-encoded certificate -----END CERTIFICATE-----

-----BEGIN CERTIFICATE----- base64-encoded certificate -----END CERTIFICATE-----

-----BEGIN CERTIFICATE----- base64-encoded certificate -----END CERTIFICATE-----
다음 Java 샘플은 GetCertificateAuthorityCertificate 작업을 사용하는 방법을 보여줍니다.

이 작업은 사설 인증 기관(CA)을 위한 인증서와 인증서 체인을 검색합니다. 인증서와 인증서 체인 모두 base64 PEM 인코딩됩니다. 인증서 체인에는 CA 인증서가 포함되지 않습니다. 체인의 각 인증서는 앞의 인증서를 서명합니다.

```java
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";
        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,
        GetCertificateAuthorityCertificateRequest req =
            new GetCertificateAuthorityCertificateRequest()
                .withCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
                             "certificate-authority/12345678-1234-1234-1234-123456789012");

        // Create a result object.
        GetCertificateAuthorityCertificateResult result = null;
        try{
            result = client.getCertificateAuthorityCertificate(req);
        }
```
지정한 인증 기관(CA)에 대해 다음과 비슷한 인증서와 인증서 체인이 출력됩니다.

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

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

GetCertificateAuthorityCsr

다음 Java 샘플은 GetCertificateAuthorityCsr 작업을 사용하는 방법을 보여줍니다.

이 작업은 사설 인증 기관(CA)을 위한 인증서 서명 요청(CSR)을 검색합니다. CSR은 CreateCertificateAuthority 작업을 호출할 때 생성됩니다. CSR은 온프레미스 X.509 인프라로 가져와서 루트 또는 하위 CA를 사용해 서명합니다. 그런 다음 ImportCertificateAuthorityCertificate 작업을 호출하여 ACM PCA에 서명된 인증서를 다시 가져옵니다. CSR은 base64 PEM 인코딩 문자열로 반환됩니다.

```java
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;

public class GetCertificateAuthorityCsr {
    public static void main(String[] args) throws Exception {
      // Retrieve and display the certificate information.
      String strPcaCert = result.getCertificate();
      System.out.println(strPcaCert);
      String strPCACChain = result.getCertificateChain();
      System.out.println(strPCACChain);
    }
```
GetCertificateAuthorityCsr

// 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";
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 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(PrivateCAException e) {
            //Unexpected service exception.
        }
    }

// Create the request object and set the CA ARN.
GetCertificateAuthorityCsrRequest req = new GetCertificateAuthorityCsrRequest();
req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account: " +
    "certificate-authority/12345678-1234-1234-1234-123456789012");

// 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);
}

지정한 CA에 대해 다음과 비슷한 출력이 표시됩니다. 인증서 서명 요청(CSR)은 PEM 형식으로 base64 인코딩됩니다. CSR을 로컬 파일에 저장하고 온프레미스 X.509 인프라를 가져와서 루트 또는 하위 CA를 사용해 서명합니다.
ImportCertificateAuthorityCertificate

다음 Java 샘플은 ImportCertificateAuthorityCertificate 작업을 사용하는 방법을 보여줍니다.

이 작업은 서명된 사설 CA 인증서를 ACM Private CA로 가져옵니다. 이 작업을 호출할 수 있으려면 먼저 CreateCertificateAuthority 작업을 호출하여 사설 CA를 생성해야 합니다. 그런 다음, GetCertificateAuthorityCsr 작업을 호출하여 인증서 서명 요청(CSR)을 생성해야 합니다. CSR를 온프레미스 CA로 가져와서 루트 인증서나 하위 인증서를 사용해 서명합니다. 인증서 체인을 생성하고 서명된 인증서와 인증서 체인을 작업 디렉터리에 복사합니다.

```java
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";
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-----" +
    "base64-encoded certificate" +
    "-----END CERTIFICATE-----";
ByteBuffer certByteBuffer = stringToByteBuffer(strCertificate);
req.setCertificate(certByteBuffer);

// Set the certificate chain.
String strCertificateChain =
    "-----BEGIN CERTIFICATE-----" +
    "base64-encoded certificate" +
    "-----END CERTIFICATE-----";
ByteBuffer chainByteBuffer = stringToByteBuffer(strCertificateChain);
req.setCertificateChain(chainByteBuffer);

// Set the certificate ARN.
req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account: " +
    "certificate-authority/12345678-1234-1234-1234-123456789012");

// 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;
}
throw ex;
Java 샘플은 IssueCertificate 함수를 사용하는 방법을 보여줍니다. 이 함수는 사설 인증 기관(CA)을 사용하여 최종 엔티티 인증서를 발급합니다. 이 함수는 인증서의 Amazon 리소스 이름(ARN)을 반환합니다. GetCertificate를 호출하고 ARN을 지정하여 인증서를 검색할 수 있습니다.

```java
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";
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-----" +
  "    base64-encoded certificate     " +
  "-----END CERTIFICATE REQUEST-----";
ByteBuffer csrByteBuffer = stringToByteBuffer(strCSR);
req.setCsr(csrByteBuffer);

// 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; }
ListCertificateAuthorities

다음과 유사하게 출력되어야 합니다:

```java
arn:aws:acm-pca:region:account:certificate-authority/12345678-1234-1234-1234-123456789012/certificate/2669d5cacb539c0830998c3babababdc
```

다음 Java 샘플은 ListCertificateAuthorities 작업을 사용하는 방법을 보여줍니다.

이 작업은 CreateCertificateAuthority 작업을 사용하여 생성한 사설 CA를 나열합니다.

```java
package com.amazonaws.samples;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
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";
        EndpointConfiguration endpoint = new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);
        // Create a client that you can use to make requests.
        AWSACMPCA client = AWSACMPCAClientBuilder.standard().withClientConfiguration(endpointConfiguration).withCredentials(new AWSStaticCredentialsProvider(credentials)).build();
        ListCertificateAuthoritiesRequest request = new ListCertificateAuthoritiesRequest();
        ListCertificateAuthoritiesResult result = client.listCertificateAuthorities(request);
        // Retrieve and display the certificate ARN.
        String arn = result.getCertificateArn();
        System.out.println(arn);
    }
}
```
AWS Certificate Manager Private
Certificate Authority 사용 설명서

ListCertificateAuthorities

```
.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());
```

나열할 CA가 있는 경우에는 다음과 비슷하게 출력이 됩니다.

```
[{  
    Arn: arn: aws: acm-pca: region: account: certificate-authority/12345678-1234-1234-1234-123456789012,  
    CreatedAt: TueNov07 12:05:39PST2017,  
    LastStateChangeAt: WedJan10 12:35:39PST2018,  
    Type: SUBORDINATE,  
    Serial: 4109,  
    Status: DISABLED,  
    NotBefore: TueNov07 12:19:15PST2017,  
    NotAfter: FriNov05 13:19:15PDT2027,  
    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: WedSep13 12:54:52PDT2017,  
    LastStateChangeAt: WedSep13 12:54:52PDT2017,  
    Type: SUBORDINATE,  
    Serial: 4100,  
    Status: ACTIVE,  
    NotBefore: WedSep13 11:19PDT2017,  
    NotAfter: SatSep11 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
    }
}
ListPermissions
다음 Java 샘플에서는 ListPermissions 함수를 사용하는 방법을 보여 줍니다.
이 함수는 사설 CA가 할당한 권한(있는 경우)을 나열합니다. IssueCertificate, GetCertificate 및 ListPermissions를 포함한 권한은 CreatePermission 작업을 사용하여 AWS 서비스 보안 주체에 할당하고 DeletePermissions 작업을 사용해 해지할 수 있습니다.

```java
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);
        }
```

버전 latest 105
// Define the endpoint for your sample.
String endpointRegion = "region";
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.
ListPermissionsRequest req = new ListPermissionsRequest();
req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
    "certificate-authority/12345678-1234-1234-1234-123456789012");

// 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);

// ListTags

지정된 사설 CA가 서비스 보안 주체에 권한을 할당한 경우 출력은 다음과 유사해야 합니다.

`[[
    "Arn": arn:aws:acm-pca:region:account:permission/12345678-1234-1234-1234-123456789012,
    "CreatedAt": WedFeb0317: 05: 39PST2019,
    "Principal": acm.amazonaws.com,
    "Permissions": {
        "ISSUE_CERTIFICATE",
        "GET_CERTIFICATE",
        "DELETE_CERTIFICATE"
    },
    "SourceAccount": 012345678901
]]`

ListTags

다음 Java 샘플은 ListTags 작업을 사용하는 방법을 보여줍니다.
이 작업은 사설 CA에 연결되는 태그를 나열합니다. 태그는 CA를 식별 및 구성하는 데 사용할 수 있는 레이블입니다. 각 태그는 키와 값(선택 사항)으로 구성됩니다. TagCertificateAuthority 작업을 호출하여 CA에 태그 를 하나 이상 추가합니다. UntagCertificateAuthority 작업을 호출하여 태그를 제거합니다.

```java
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";
        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.
        ListTagsRequest req = new ListTagsRequest();
        req.withCertificateAuthorityArn("arn:aws:acmpca:region:account:" +
            "certificate-authority/12345678-1234-1234-1234-123456789012");

        // List the tags
        ListTagsResult result = null;
        try {
            result = client.listTags(req);
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        }
    }
}
```
이 사용 설명서에 제시된 RestoreCertificateAuthority 작업을 사용하는 방법을 보여줍니다. 사설 CA는 복원 기간 중 언제든지 복원할 수 있습니다. 현재 이 기간은 삭제일로부터 7일 ~ 30일로 설정할 수 있으며 CA를 삭제할 때 정의할 수 있습니다. 자세한 내용은 CA 복원 (p. 65) 단원을 참조하십시오. DeleteCertificateAuthority (p. 87) Java 예제도 참조하십시오.

```java
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.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 {
        try {
            AWSCredentials credentials = new ProfileCredentialsProvider("default").getCredentials();
            EndpointConfiguration endpoint = new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);
            AWSPrivateCA client = new AWSPrivateCAClientBuilder().withEndpointConfiguration(endpoint).withCredentials(new AWSStaticCredentialsProvider(credentials)).build();
            AmazonClientException e = null;
            try {
                String endpointRegion = "region";
                RestoreCertificateAuthorityRequest request = new RestoreCertificateAuthorityRequest();
                client.restoreCertificateAuthority(request);
            }
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load your credentials from file.", e);
        }
    }

    // Define the endpoint for your sample.
    String endpointRegion = "region";
    EndpointConfiguration endpoint = new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

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

다음 Java 샘플은 RevokeCertificate 작업을 사용하는 방법을 보여줍니다.

이 작업은 IssueCertificate 작업을 호출하여 발급된 인증서를 해지합니다. 사설 CA를 생성 또는 업데이트할 때 인증서 해지 목록(CRL)을 사용하도록 설정한 경우에는 해지된 인증서에 대한 정보가 CRL에 포함됩니다. ACM Private CA는 지정한 Amazon S3 버킷에 CRL을 기록합니다. 자세한 내용은 CrlConfiguration 구조를 참조하십시오.

```java
package com.amazonaws.samples;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
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 {
    // 버전 latest
    109
```
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";
    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:region:account:" +
        "certificate-authority/12345678-1234-1234-1234-123456789012");

    // Set the certificate serial number.

    // Set the RevocationReason.
    req.withRevocationReason(RevocationReason.KEY_COMPROMISE);

    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;
    }
}
다음 Java 샘플은 TagCertificateAuthority 작업을 사용하는 방법을 보여줍니다. 이 작업은 사설 CA에 하나 이상의 태그를 추가합니다. 태그는 AWS 리소스를 식별 및 구성하는 데 사용할 수 있는 레이블입니다. 이 작업을 호출할 때 사설 CA의 Amazon 리소스 이름(ARN)을 지정합니다. 키-값 페어를 사용하여 태그를 지정합니다. 해당 CA의 구체적인 특성을 식별하기 위해 하나의 사설 CA에만 태그를 적용할 수 있습니다. 또는 이러한 CA 간의 공통 관계를 필터링하기 위해 여러 사설 CA에 동일한 태그를 적용할 수 있습니다. 하나 이상의 태그를 제거하려면 UntagCertificateAuthority 작업을 사용합니다. ListTags 작업을 호출하여 CA에 어떤 태그가 연결되는지 확인합니다.

```java
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
        // 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";
        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 ARN of the certificate.
TagCertificateAuthorityRequest req = new TagCertificateAuthorityRequest();
req.setCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
  "certificate-authority/12345678-1234-1234-1234-123456789012");
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;
}
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";
        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);
        tags.add(tag);

        // Create a request object and specify the ARN of the certificate.
        UntagCertificateAuthorityRequest req = new UntagCertificateAuthorityRequest();
        req.withCertificateAuthorityArn("arn:aws:acm-pca:region:account:" +
            "certificate-authority/12345678-1234-1234-1234-123456789012");
        req.withTags(tags);

        // Delete the tag
        try{
            client.untagCertificateAuthority(req);
        } catch (InvalidArnException ex) {
            throw ex;
        } catch (ResourceNotFoundException ex) {
            throw ex;
        }
    }
}
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";
        EndpointConfiguration endpoint =
            new AwsClientBuilder.EndpointConfiguration(endpointProtocol, endpointRegion);

        // Create a new ACM PCA client using your credentials, endpoint configuration, and AWS region.
        AWSACMPCA client = new AWSACMPCAClientBuilder() // Please refer to API documentation.
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .withEndpointConfiguration(endpoint)
            .build();

        // Use the client to update the certificate authority.
        UpdateCertificateAuthorityRequest request = new UpdateCertificateAuthorityRequest(
            "my-certificate-authority-arn", // Replace with your ARN.
            // Set the new status to either ACTIVE or DISABLED.
            CertificateAuthorityStatus.DISABLED // Replace with the desired status.
        );
        CertificateAuthorityStatus newStatus = client.updateCertificateAuthority(request);
    }
}
// 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:region:account:" +
    "certificate-authority/12345678-1234-1234-1234-123456789012");

// 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;
}
외부 CA를 사용하여 사설 CA 인증서 서명

사설 CA 계층 구조의 신뢰 루트가 ACM Private CA 외부의 CA여야 하는 경우에는 자체 루트 CA를 생성하고 자체 서명할 수 있습니다. 또는 조직에서 운영하는 외부 사설 CA에서 서명한 사설 CA 인증서를 얻을 수 있습니다. 외부에서 얻은 이러한 CA를 사용하여 ACM Private CA가 관리하는 사설 하위 CA 인증서에 서명합니다.

**Note**

외부 CA를 생성하거나 얻는 절차는 이 설명서에서는 다루지 않습니다.

ACM Private CA와 함께 외부 상위 CA를 사용하면 CA 이름 제약 조건을 적용할 수도 있습니다. 이름 제약 조건은 인터넷 퍼블릭 키 인프라(PKI) 표준 RFC 5280에 정의되어 있습니다. 이 제약 조건을 사용하면 CA 관리자가 인증서에 주제 이름을 제한할 수 있습니다. 자세한 내용은 RFC 5280의 이름 제약 조건 섹션을 참조하십시오.

ACM Private CA는 이름 제약 조건을 사용하여 CA 인증서를 직접 발급할 수 없습니다. 그러나 외부 상위 CA의 이름 제약 조건을 포함하는 CA 인증서를 발급할 수 있습니다. ACM Private CA는 하위 CA 및 최종 엔터티 인증서를 발급할 때 이러한 제약 조건을 적용합니다. 자세한 내용은 사설 CA에서 이름 제약 조건 적용 (p. 122) 단원을 참조하십시오.

주제
- 인증서 서명 요청(CSR) 가져오기 (p. 116)
- 사설 CA 인증서 서명 (p. 118)
- 사설 CA 인증서를 ACM Private CA로 가져오기 (p. 119)
- ACM Private CA에서 생성된 사설 인증서 검색 (p. 121)
- CA 인증서 검색 (p. 121)
- 사설 CA에서 이름 제약 조건 적용 (p. 122)

인증서 서명 요청(CSR) 가져오기

외부 CA로 서명할 사설 하위 CA를 생성한 경우에는 인증서 서명(CSR)을 검색해야 합니다. 그런 다음 파일에 저장합니다. 다음 절차에 설명되어 있듯이 AWS Management 콘솔 또는 AWS CLI를 사용해 검색할 수 있습니다. CSR을 검사하고 싶다면 다음 OpenSSL 명령을 사용합니다.

```bash
openssl req -in path_to_CSR_file -text -noout
```

이 명령은 다음과 비슷한 출력을 생성합니다. CA 확장은 TRUE이고, 이는 CSR이 CA 인증서에 대한 것임을 나타냅니다.

```
Certificate Request:
  Data:
    Version: 0 (0x0)
    Subject: CN=Example CA 1
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
```

버전 latest
116
CSR 검색(콘솔): 방법 1

단계에 따라 ACM Private CA에서 사설 CA를 생성 (p. 51)하고 성공 대화 상자를 그대로 열어둔 경우에는 이 절차를 사용합니다.

1. ACM Private CA에서 사설 CA가 성공적으로 생성된 직후에 시작하기를 선택합니다. ACM Private CA 콘솔의 CSR를 반환합니다. 나중에 이 단계로 돌아갈 수 있습니다.
2. 파일로 CSR 내보내기를 선택하고 이를 로컬에 저장합니다.
4. 사설 CA 인증서 서명 (p. 118)의 지침을 따르십시오.

CSR 검색(콘솔): 방법 2

단계에 따라 ACM Private CA에서 사설 CA를 생성 (p. 51)하고 성공 대화 상자를 닫은 경우에는 이 절차를 사용합니다.

1. 계속 진행할 준비가 되면 AWS Certificate Manager 콘솔을 열고 왼쪽 탐색 창에서 사설 CA를 선택합니다.
2. 목록에서 사설 CA를 선택합니다.
3. 작업 메뉴에서 CA 인증서 가져오기를 선택합니다. ACM Private CA 콘솔이 CSR를 반환합니다.
4. 파일로 CSR 내보내기를 선택하고 이를 로컬에 저장합니다.
5. [Next]를 선택합니다.
6. 사설 CA 인증서 서명 (p. 118)의 지침을 따르십시오.

**CSR 검색(AWS CLI)**

이 절차에 따라 AWS Command Line Interface를 사용하여 CSR을 검색합니다.

**CSR 검색 방법(AWS CLI)**

1. `get-certificate-authority-csr` 명령을 사용하여 사설 CA에 대한 인증서 요청(CSR)을 검색합니다. 디스플레이에 CSR을 전송하고 싶다면 --output text 옵션을 사용하여 각 줄의 끝에서 CR/LF 문자를 없입니다. CSR을 파일로 전송하려면 파일 이름 다음에 나오는 리디렉션 옵션(>)을 사용합니다.

   ```
   aws acm-pca get-certificate-authority-csr 
   --certificate-authority-arn arn:aws:acm-pca:region:account:
   certificate-authority/12345678-1234-1234-1234-123456789012 
   --output text
   ```

2. 사설 CA 인증서 서명 (p. 118)의 지침을 따르십시오.

**사설 CA 인증서 서명**

ACM Private CA를 사용해 사설 CA를 생성 (p. 51)하고 인증서 서명 요청(CSR)을 검색 (p. 116)하고 난 후에는 CSR을 외부 X.509 인프라에 가져와야 합니다. 중간 또는 루트 CA를 사용하여 사설 CA 인증서를 생성하고 서명합니다. 서명은 조직 내 사설 CA의 자격 증명을 확인합니다. 이러한 프로세스가 완료되면 사설 CA 인증서를 ACM Private CA로 가져오기 (p. 119)의 지침을 따릅니다.

**Important**

- X.509 인프라와 인프라 내 CA 계층 구조에 대한 자세한 내용은 본 문서에서는 다루지 않습니다. 자세한 내용은 사설 CA 인증서의 생성 및 서명 (p. 125) 단원을 참조하십시오.
- 사설 CA의 유효 기간은 사설 CA 인증서를 생성할 때 지정한 유효 기간에 의해 결정됩니다. 하한 및 상한의 필드를 설정할 수 있습니다. 정의된 기간을 적용하는 것 외에도 ACM Private CA는 CA의 수명을 제한하지 않습니다.
- 효과적으로 기간이 만료되지 않는 CA 인증서를 생성해야 하는 경우에는 상한 필드에 특수 값 `99991231235959Z`를 설정합니다. 그러나 이것은 권장되는 모범 사례는 아닙니다.

서명된 인증서는 보통 base64 인코딩 PEM 파일 또는 문자열 형태로 반환됩니다. 방법은 다음 예제와 같습니다. 인증서가 다른 형식으로 인코딩된 경우에는 이를 PEM으로 변환해야 합니다. 형식 변환을 위해 다양한 OpenSSL 명령어를 사용할 수 있습니다.

```
-----BEGIN CERTIFICATE-----
MIIDRzCCA..................9YFbLXtPgZooy2IgZ
------END CERTIFICATE------
```

OpenSSL `x509` 명령을 사용하여 서명된 PEM 형식 인증서의 내용을 볼 수 있습니다.

```
openssl x509 -in path_to_certificate_file -text -noout
```
이 명령은 다음 예제와 비슷한 인증서를 출력합니다.

Certificate:
Data:
  Version: 3 (0x2)
  Serial Number: 4122 (0x101a)
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C=US, ST=Washington, L=Seattle, O=Example Company, OU=Corp,
          CN=www.example.com/emailAddress=corp@www.example.com
  Validity
    Not After: Mar 26 19:28:43 2028 GMT
  Subject: O=Example Company, OU=Corporate Office, CN=Example Company CA 1
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (2048 bit)
      Modulus:
        38:08:95:50:4c:be:7d:bf:df:ad:01:29:37:
        14:4d:1f:fc:95:2b:51:fc:de:8f:ee:02:80:0c:6b:
        9a:17:46:
      Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Subject Key Identifier:
  X509v3 Authority Key Identifier:
X509v3 Basic Constraints: critical
  CA:TRUE
X509v3 Key Usage: critical
  Digital Signature, Certificate Sign, CRL Sign
Signature Algorithm: sha256WithRSAEncryption
  88:19

사설 CA 인증서를 ACM Private CA로 가져오기

사설 CA 생성 (p. 51) 하고, 인증서 서명 요청(CSR)을 검색 (p. 116) 하고, CA 인증서를 서명 (p. 118) 하고 난 후에는 인증서를 ACM Private CA로 가져와야 합니다. 인증서를 서명하고 가져온 후에는 사설 CA를 사용하여 신뢰할 수 있는 사설 SSL/TLS 인증서 발급 및 취소할 수 있습니다. 이러한 인증서 덕분에 조직 내 버전 latest
119
의 사용자, 애플리케이션, 컴퓨터 및 기타 디바이스 간에 신뢰할 수 있는 통신이 가능합니다. 사설 인증서는 공개적으로 신용할 수 없습니다.

또한 사설 CA 인증서와 이전 인증서를 서명하는 데 사용되는 중간 또는 루트 CA의 인증서를 포함하고 있는 인증서 체인을 검색해야 합니다. 인증서 체인을 생성하기 위해 루트 인증서(사용 가능한 경우)와 단일 파일에 포함시킬 수 있는 모든 하위 인증서를 연결합니다. 이를 위해 cat 명령(Windows)을 사용할 수 있습니다. 각 인증서는 앞에 지정된 인증서를 직접 인증해야 합니다. 다음 예제의 경우에는 세 개의 인증서가 포함되어 있지만, PKI 인프라의 경우 그보다 많거나 적을 수 있습니다.

Note

체인은 PEM 형식이어야 합니다.

```
-----BEGIN CERTIFICATE-----
Base64-encoded intermediate CA certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Base64-encoded intermediate CA certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Base64-encoded root or intermediate CA certificate
-----END CERTIFICATE-----
```

사설 CA 인증서 가져오기(콘솔)

AWS Management 콘솔을 사용하여 사설 CA 인증서를 가져올 수 있습니다.

CA 인증서를 가져오는 방법(콘솔)

1. 콘솔이 여전히 서명된 CA 인증서 가져오기 페이지로 열리는 경우에는 7단계로 건너뛰십시오. 그렇지 않은 경우 계속 진행합니다.
3. 사설 CA를 선택합니다.
4. 목록에서 사설 CA를 선택합니다.
5. 작업 메뉴에서 CA 인증서 가져오기를 선택합니다.
7. 인증서 본문에서 서명된 사설 CA 인증서를 텍스트 상자에 복사하거나 파일에서 가져옵니다.
8. 인증서 체인에서 인증서 체인을 텍스트 상자로 복사하거나 파일에서 가져옵니다.
10. 확인 및 가져오기를 선택하여 사설 CA 인증서를 가져옵니다.

사설 CA 인증서 가져오기(AWS CLI)

시작에 앞서 서명된 CA 인증서와 인증서 체인이 PEM 형식 파일에 있는지 확인합니다.

CA 인증서를 가져오는 방법(AWS CLI)

import-certificate-authority-certificate 명령을 사용하여 ACM Private CA에 사설 CA 인증서를 가져옵니다.

```
aws acm-pca import-certificate-authority-certificate
```
ACM Private CA에서 생성된 사설 인증서 검색

ACM Private CA API 및 AWS CLI를 사용하여 사설 인증서를 발급할 수 있습니다. 이 경우 AWS CLI 또는 ACM Private CA API를 사용하여 해당 인증서를 검색할 수 있습니다. ACM을 사용해 사설 CA를 생성하고 인증서를 요청한 경우에는 ACM을 사용하여 해당 인증서와 암호화된 프라이빗 키를 내보내야 합니다. 자세한 내용은 사설 인증서 내보내기를 참조하십시오.

get-certificate 명령을 사용하여 사설 인증서를 검색합니다. 또한 GetCertificate 작업을 사용할 수도 있습니다. <CR><LF> 페어 없이 인증서를 출력하려면 --output text 옵션을 사용하십시오.

Note

인증서를 취소하고 싶은 경우에는 get-certificate 명령을 사용하여 16진수 형식의 일련 번호를 검색할 수 있습니다. 또한 감사 보고서를 생성하여 16진수 일련 번호를 검색할 수 있습니다. 자세한 내용은 사설 CA에 대한 감사 보고서 생성 (p. 32) 단원을 참조하십시오.

CA 인증서 검색

ACM Private CA API 및 AWS CLI를 사용하여 사설 CA에 대한 CA 인증서를 검색할 수 있습니다. get-certificate-authority-certificate 명령을 실행할 수 있습니다. 또한 GetCertificateAuthorityCertificate 작업을 호출할 수도 있습니다. <CR><LF> 페어 없이 인증서를 출력하려면 --output text 옵션을 사용하십시오.
이명령은 base64로 인코딩된 PEM 형식의 인증서와 인증서 체인을 출력합니다.

```
-----BEGIN CERTIFICATE-----
...Base64-encoded certificate...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
...Base64-encoded certificate...
-----END CERTIFICATE-----
```

사설 CA에서 이름 제약 조건 적용

이름 제약 조건은 인터넷 퍼블릭 키 인프라(PKI) 표준 RFC 5280에 정의되어 있습니다. 이러한 제약 조건을 사용하면 CA 관리자가 인증서의 주제 이름을 제한할 수 있습니다. 관리자는 현재 주제 CA에서 발급한 인증서에서 허용 또는 금지되는 이름을 제어할 수 있습니다. 예를 들어 “project1.corp” 같은 프라이빗 도메인 이름을 허용할 수 있습니다. 그러나 “project2.corp” 같은 다른 프라이빗 도메인 이름은 다른 내부 프로젝트에 대해 예약이 될 수 있습니다. 마찬가지로 “example.com” 또는 “*.com” 같은 공용 DNS 이름은 거부될 수 있습니다. 자세한 내용은 RFC 5280의 이름 제약 조건 섹션을 참조하십시오.

ACM Private CA는 CA 인증서에 포함된 이름 제약 조건을 적용합니다. 그러나 ACM Private CA에 의해 발급된 CA 인증서는 이름 제약 조건이 포함될 수 없습니다. 사설 CA 계층 구조에서 이름 제약 조건을 적용하면 CA 관리자가 CA 인증서 발급을 지원하는 외부 CA를 사용하여 서비치 CA 인증서에 제약 조건을 설정할 수 있습니다. 그러한 혼합 인증서는 ACM Private CA로 가져옵니다. 이때 사설 CA에서 발급한 인증서가 이름 제약 조건을 상속합니다.

**Note**

외부 CA를 생성하거나 얻는 절차는 이 설명서에서는 다루지 않습니다.

이름 제약 조건은 다운스트림 인증서 하위 트리에서 허용 또는 제외될 수 있는 이름을 정의합니다. 이름 제약 조건을 적용하면 하위 CA 관리자가 인증서의 범위를 벗어난 도메인이나 호스트에 대한 인증서를 발급할 수 없습니다.

ACM Private CA는 다음과 같은 이름 양식에 대해 이름 제약 조건을 지정합니다.

- IPAddress
- DirectoryName
- UniformResourceIdentifier
- EmailAddress
- DNSName

ACM Private CA는 다음과 같이 지원되지 않는 이름 양식에 대해 이름 제약 조건을 지원합니다.

- OtherName
- X400Address
- RegisteredID
- EDIPartyName

이름 제약 정책의 예

아래의 각 예에는 이름 제약 조건의 집합을 정의하는 스타트 인증서 정책이 포함되어 있습니다. 이 예에는 제약 조건으로 인한 인증서 발급 제한에 대한 설명도 포함되어 있습니다. 다음과 같은 일반 규칙이 적용됩니다.
• 이름 제약 조건은 허용 하위 트리, 제외 하위 트리, 또는 두 다로 표시됩니다.
• 허용 및 제외 하위 트리에는 일치하는 패턴이 포함되어 있으며, 비어 있을 수 있습니다. permitted 하위 트리가 비어 있으면 해당 양식의 모든 이름이 거부됩니다. 마찬가지로 excluded 하위 트리가 비어 있으면 해당 양식의 모든 이름이 허용됩니다.
• 이름 양식이 나열되지 않으면 해당 양식의 모든 이름이 허용됩니다.
• 한 이름 양식에 제한 조건을 적용해도 다른 이름 양식에는 영향을 주지 않습니다.

허용 하위 트리만 해당

nameConstraints=critical,permitted;
DNSName:.private
DNSName:.local
DNSName:.corp
UniformResourceIdentifier:

결과: "private", "local" 또는 "corp"와 일치하는 경우에만 DNS 이름이 허용됩니다. 일치하는 패턴이 없기 때문에 UniformResourceIdentifier 양식의 이름이 거부됩니다. 지원되는 미등록 양식에 있는 모든 이름이 허용됩니다.

제외 하위 트리만 해당

nameConstraints=critical,excluded;
DNSName:.example.com
DNSName:.local

결과: "example.com" 또는 "local"와 일치하는 경우에는 DNS 이름이 거부됩니다. 지원되는 미등록 양식에 있는 모든 이름이 허용됩니다.

허용 및 제외 하위 트리

nameConstraints=critical,permitted;
DNSName:.private
DNSName:.local
DNSName:.corp
nameConstraints=critical,excluded;
DNSName:.secret.corp
DNSName:.example.com

결과: "private" 또는 "local"과 일치하는 DNS 이름이 허용됩니다. "corp"와 일치하는 DNS 이름은 "secret.corp"과 일치하지 않는 경우에만 허용됩니다. "example.com"과 일치하는 DNS 이름은 거부됩니다. 지원되는 미등록 양식에 있는 모든 이름이 허용됩니다.

허용 하위 트리만 해당(체인의 두 CA 인증서에 지정)

Root CA:
nameConstraints=critical,permitted;
DNSName:.private
DNSName:.local
DNSName:.corp

Subordinate CA:
nameConstraints=critical,permitted;
DNSName:sub.private
DNSName:a.b.local
결과: "sub.private" 또는 "a.b.local"과 일치하는 경우에 DNS 이름이 허용됩니다. "corp"와 일치하는 DNS 이름이 거부됩니다. 지원되는 미등록 양식에 있는 모든 이름이 허용됩니다.

제외된 하위 트리만 해당(체인의 두 CA 인증서에 지정)

```
Root CA:
nameConstraints=critical,excluded;
DNSName:foo.private
DNSName:foo.example
Subordinate CA:
nameConstraints=critical,excluded;
DNSName:bar.private
DNSName:.example
IPAddress:0.0.0.0/0
```

결과: "foo.private", "bar.private" 또는 ".example"와 일치하지 않으면 DNS 이름이 허용됩니다. 일치하는 패턴이 모든 IP 주소와 일치하므로 IPAddress 양식의 이름이 거부됩니다. 지원되는 미등록 양식에 있는 모든 이름이 허용됩니다.
문제 해결

AWS Certificate Manager Private Certificate Authority 사용 설명서 사용 설명서를 사용하는 동안 문제가 발생할 경우 다음 주제를 참조하십시오.

주제
• 사설 CA 인증서의 생성 및 서명 (p. 125)
• CRL 버킷 생성을 허용하도록 Amazon S3 구성 (p. 125)
• 자체 서명된 CA 인증서 삭제 (p. 125)
• Amazon S3 권한 및 CRL 버킷 (p. 125)
• 예외 처리 (p. 126)

사설 CA 인증서의 생성 및 서명

사설 CA를 생성하고 나면 CSR을 검색하여 X.509 인프라의 중간 또는 루트 CA에 제출해야 합니다. CA는 CSR을 사용하여 사설 CA 인증서를 생성한 다음, 사용자에게 반환하기 앞서 인증서를 서명합니다.

안타깝게도 사설 CA 인증서의 생성 및 서명과 관련된 문제에 대해 구체적인 조언을 제공할 수 없습니다. X.509 인프라와 인프라 내 CA 계층에 대한 자세한 내용은 본 문서에서는 다루지 않습니다.

CRL 버킷 생성을 허용하도록 Amazon S3 구성

계정에 Amazon S3 퍼블릭 액세스 차단(버킷 설정)이 적용되는 경우 사설 CA가 CRL 버킷을 생성하지 못할 수 있습니다. 이 경우 Amazon S3 설정을 확인하십시오. 자세한 내용은 Amazon S3 퍼블릭 액세스 차단 사용 단원을 참조하십시오.

자체 서명된 CA 인증서 삭제

자체 서명된 CA 인증서는 해지할 수 없습니다. 대신 CA를 삭제해야 합니다.

Amazon S3 권한 및 CRL 버킷

ACM Private CA 및 IAM 보안 주제 모두 CRL에 대해 지정한 Amazon S3 버킷에 쓰기를 수행할 수 있는 권한이 있어야 합니다. 호출을 수행하는 IAM 보안 주체가 버킷에 쓰기를 수행할 수 있는 권한을 가지고 있지 않으면 에러가 발생합니다. 이에 대한 알림을 제공하려면 Amazon CloudWatch의 MisconfiguredCRLBucket 지표에 대한 경보를 구성합니다. 자세한 내용은 ACM 사설 CA에 대한 액세스 구성 및 지원되는 CloudWatch 지표를 참조하십시오.

이 예외가 발생하면 Amazon S3 계정에서 새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단 구성 옵션을 선택하지 않았는지 확인합니다. 자세한 내용은 S3 버킷에 대한 퍼블릭 액세스
예외 처리

여러 가지 이유로 ACM Private CA 명령이 실패할 수 있습니다. 각 예외와 예외를 해결하기 위한 권장 사항은 아래 표를 참조하십시오.

<table>
<thead>
<tr>
<th>ACM Private CA 예외</th>
<th>설명</th>
<th>권장 사항</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessDeniedException</td>
<td>해당 명령을 사용하는 데 필요한 권한을 사설 CA가 호출 계정에 위임되지 않았습니다.</td>
<td>ACM Private CA의 권한 위임에 대한 자세한 내용은 ACM에서 생성한 인증서 생성 권한 할당 (p. 67) 단원을 참조하십시오.</td>
</tr>
<tr>
<td>InvalidArgsException</td>
<td>잘못된 파라미터를 사용하여 인증서 생성 또는 갱신 요청이 수행되었습니다.</td>
<td>명령의 개별 설명서에서 입력 매개변수의 유용성 확인합니다. 새 인증서를 생성하는 경우, 요청된 서명 알고리즘은 CA의 키 유형과 함께 사용할 수 있는지 확인합니다.</td>
</tr>
<tr>
<td>InvalidStateException</td>
<td>연결된 사설 CA가 ACTIVE 상태가 아니므로 인증서를 갱신할 수 없습니다.</td>
<td>사설 CA 복원 (p. 65)을 시도합니다. 사설 CA가 복원 기간을 벗어나면 CA를 복원할 수 없으며 인증서를 갱신할 수 없습니다.</td>
</tr>
<tr>
<td>LimitExceededException</td>
<td>각 CA(인증기관)에는 발급이 가능한 인증서 할당량이 있습니다. 지정된 인증서와 연결된 사설 CA가 할당량에 도달했습니다. 자세한 내용은 할당량 (p. 2)을 참조하십시오.</td>
<td>할당량 증가를 요청하려면 AWS 지원 센터에 문의하십시오.</td>
</tr>
<tr>
<td>MalformedCSRException</td>
<td>ACM Private CA에 제출된 인증서 서명 요청(CSR)을 확인하거나 유효성 검사를 할 수 없습니다.</td>
<td>CSR가 올바르게 생성되고 구성되는지 확인합니다.</td>
</tr>
<tr>
<td>OtherException</td>
<td>내부 오류로 인해 요청이 실패했습니다.</td>
<td>명령을 다시 실행해 보십시오. 문제가 지속되면 AWS 지원 센터에 문의하십시오.</td>
</tr>
<tr>
<td>RequestFailedException</td>
<td>사용자 AWS 환경의 네트워킹 문제로 인해 요청이 실패했습니다.</td>
<td>요청을 다시 시도하십시오. 오류가 지속되면 Amazon VPC(VPC) 구성을 확인합니다.</td>
</tr>
<tr>
<td>ResourceNotFoundException</td>
<td>인증서를 발급한 사설 CA가 삭제되어 이상 존재하지 않습니다.</td>
<td>다른 활성 CA에서 새 인증서를 요청합니다.</td>
</tr>
<tr>
<td>ThrottlingException</td>
<td>요청된 API 작업이 할당량을 초과하여 실패했습니다.</td>
<td>ACM Private CA 서비스 할당량 (p. 2)에서 허용하는 것보다 많은 호출을 발생하고 있지 않은지 확인합니다. 할당량 초과가 아니라 일시적인 조건으로 인해 ThrottlingException 오류가 발생하였습니다.</td>
</tr>
<tr>
<td>ACM Private CA에 의해 반환된 예외</td>
<td>설명</td>
<td>권장 사항</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>ValidationException</td>
<td>요청의 입력 파라미터가 형식이 잘못되었거나, 루트 인증서의 유효 기간이 요청된 인증서의 유효 기간 이전에 끝났습니다.</td>
<td>발생할 수도 있습니다. 할당량을 초과하여 호출을 하지 않았는데도 오류가 발생한 경우에는 요청을 다시 시도하십시오. 할당량에 따라 실행 중인 경우에는 할당량 증가를 요청할 수 있습니다. 자세한 내용은 API 요청에 대한 할당량 (p. 3) 단원을 참조하십시오. 명령 입력 파라미터의 구문 요구 사항과 CA 루트 인증서의 유효 기간을 확인합니다. 유효 기간 변경에 대한 자세한 내용은 사설 CA 업데이트 (p. 59) 단원을 참조하십시오.</td>
</tr>
</tbody>
</table>
용어 및 개념

다음 용어 및 개념은 AWS Certificate Manager Private Certificate Authority(ACM Private CA) 작업 시 도움이 될 수 있습니다.

주제
- 비대칭 키 암호화 기법 (p. 128)
- 기본 제약 조건 경로 길이 (p. 128)
- 인증 기관 (p. 129)
- 인증 기관 인증서 (p. 129)
- CA 깊이 (p. 130)
- CA 인증서 (p. 130)
- 인증서 서명 (p. 130)
- 인증서 경로 (p. 130)
- Domain Name System(DNS) (p. 130)
- 도메인 이름 (p. 130)
- 최종 엔터티 인증서 (p. 131)
- 사설 CA (p. 131)
- 사설 인증서 (p. 132)
- 퍼블릭 키 인프라(PKI) (p. 132)
- 루트 CA (p. 133)
- 루트 인증서 (p. 133)
- Secure Sockets Layer(SSL) (p. 133)
- 보안 HTTPS (p. 133)
- 자체 서명된 인증서 (p. 134)
- SSL 서버 인증서 (p. 134)
- TLS(전송 계층 보안) (p. 134)
- 신뢰 (p. 134)
- X.500 고유 이름 (p. 134)

비대칭 키 암호화 기법

비대칭 키 암호화 기법에서는 다른지만 수학적으로 관련된 키를 사용하여 내용을 암호화 및 해독합니다. 키 중 하나는 퍼블릭 키이며 X.509 v3 인증서에서 제공됩니다. 다른 키는 프라이빗 키이며 보안 방식으로 저장됩니다. X.509 인증서는 사용자, 컴퓨터 또는 다른 리소스(인증서 주체)의 자격 증명을 퍼블릭 키로 바인딩합니다.

사설 CA 인증서와 사설 CA에서 발급된 인증서는 X.509 SSL/TLS 인증서입니다. 이들은 사용자, 서비스, 애플리케이션, 컴퓨터 또는 기타 디바이스의 자격 증명을 인증서에 포함된 퍼블릭 키에 바인딩합니다. 연결된 프라이빗 키는 AWS에 안전하게 저장됩니다.

기본 제약 조건 경로 길이

기본 제약 조건 경로 길이는 CA 아래에 있는 CA 계층 구조의 최대값 CA 길이 (p. 130)을 정의하는 CA 인증 서의 인증서 확장입니다. 예를 들어 경로 길이 제한 조건이 0인 CA에는 하위 CA가 있을 수 없습니다. 경로 길이
인증 기관

인증 기관(CA)은 디지털 인증서를 발급하고 필요할 경우 취소합니다. 가장 일반적인 유형의 인증서는 ISO X.509 표준을 기반으로 합니다. X.509 인증서는 인증서 주체의 자격 증명을 확인하고 이를 퍼블릭 키로 바인딩합니다. 여기서 주체는 사용자, 애플리케이션, 컴퓨터 또는 기타 디바이스일 수 있습니다. CA는 인증서 내용을 해시한 다음, 인증서의 퍼블릭 키와 관련된 프라이빗 키로 해당 해시를 암호화하여 인증서를 서명합니다. 웹 브라우저가 이 인증서를 인증하기 위해서는, 해당 인증서가 인증된 인증서의 하위 인증서로 인증되어야 합니다. 키 쌍에 대한 자세한 내용은 비대칭 키 암호화 기법 (p. 128) 단원을 참조하십시오. 자세한 내용은 인증 기관 인증서 (p. 129) 단원을 참조하십시오.

ACM Private CA를 사용하여 사설 CA를 생성하고 이 사설 CA를 사용하여 인증서를 발급할 수 있습니다. 사설 CA는 조직 내에서 사용할 사설 SSL/TLS 인증서만 발급할 수 있습니다. 자세한 내용은 사설 인증서 (p. 132) 단원을 참조하십시오. 자세한 내용은 인증 기관 인증서 (p. 129) 단원을 참조하십시오.

인증 기관 인증서

인증 기관(CA) 인증서는 CA의 자격 증명을 확인하고 인증서에 포함된 퍼블릭 키에 이를 바인딩합니다.

ACM Private CA를 사용하여 사설 루트 CA 또는 사설 하위 CA를 생성할 수 있으며 각 CA에서 CA 인증서가 지원됩니다. 하위 CA 인증서는 신뢰 체인에서 보다 상위로 있는 또 다른 CA 인증서에 의해 서명됩니다. 외부 루트 기관에 온프레미스에서 호스팅됨을 설정할 수도 있습니다. 그런 다음 루트 CA의 경우 인증서는 자체 서명됩니다. 다양한 루트 기관이 온프레미스에서 호스팅될 경우 통합된 루트 CA 인증서는 지원할 수 있습니다.

다음 예제에서는 ACM Private CA X.509 CA 인증서에 포함된 일반 필드를 보여 줍니다. CA 인증서의 경우 CA: 필드의 Basic Constraints 값이 TRUE로 설정됩니다.

<table>
<thead>
<tr>
<th>Certificate:</th>
<th>Data:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version: 3 (0x2)</td>
<td>Serial Number: 4121 (0x1019)</td>
</tr>
<tr>
<td>Signature Algorithm: sha256WithRSAEncryption</td>
<td>Issuer: C=US, ST=Washington, L=Seattle, O=Example Company Root CA, OU=Corp, CN=www.example.com/emailAddress=<a href="mailto:corp@www.example.com">corp@www.example.com</a></td>
</tr>
<tr>
<td>Validity</td>
<td>Not Before: Feb 26 20:27:56 2018 GMT</td>
</tr>
<tr>
<td>Subject</td>
<td>C=US, ST=WA, L=Seattle, O=Examples Company Subordinate CA, OU=Corporate Office, CN=www.example.com</td>
</tr>
</tbody>
</table>

Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:c0: ... a3:4a:51
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Basic Constraints: critical CA:TRUE
CA 깊이
CA 계층의 인증서 수는 루트 CA에서 시작하여 하위 CA로 이동합니다. 루트는 하나로 계산되고, 추가 인증서는 각 수준에서 상위 CA에서 하위 CA로의 각 단계마다 계산됩니다. 예를 들어 루트 CA와 루트에서 직접 발급된 최종 엔터티 인증서만 있는 신뢰 체인은 깊이(경로 길이)가 2입니다. 신뢰 체인의 각 하위 CA는 깊이를 하나씩 증가시킵니다. 하나의 하위 CA와 최종 엔터티 인증서가 있는 루트 CA의 깊이는 3입니다.

CA 인증서
인증 기관을 식별합니다. CA 인증서에는 기본 제약 조건 플래그라는 특수 확장이 포함되어야 합니다. 인증서를 발급하는 데 이러한 인증서를 사용할 수 있도록 하려면 이 플래그를 true로 설정해야 합니다.

인증서 서명
디지털 서명은 인증서에 대해 암호화된 해시입니다. 서명은 인증서 데이터의 무결성을 확인하는 데 사용됩니다. 서명을 생성하기 위해서는 인증서 데이터의 암호화 해시를 사용해야 합니다. 이 해시 함수는 효과적으로 위조 불가한 고정 크기 데이터 문자열을 생성합니다. 이 문자열은 해시라고 합니다. CA는 프라이빗 키로 해시 값을 암호화하고 암호화된 해시를 인증서에 연결합니다.

서명된 인증서를 확인하기 위해 클라이언트 애플리케이션이 CA 퍼블릭 키를 사용해 서명을 해독합니다. 그 런 다음, CA가 인증서의 나머지 부분에 대한 해시를 계산하는 데 사용한 것과 동일한 서명 알고리즘을 사용합니다. CA에서 사용하는 서명 알고리즘은 인증서에 나열됩니다. 계산된 해시 값이 해독된 해시 값과 같으면 인증서가 변조되지 않은 것입니다.

인증서 경로
인증서를 사용하는 클라이언트는 최종 엔터티 인증서에서 신뢰할 수 있는 루트까지의 경로가 있는지 확인합니다. 클라이언트는 경로로 하는 각 인증서가 유효한지(해지되지 않았는지) 확인합니다. 또한 인증서가 만료되지 않았는지, 무결성을 갖추고 있는지(변조 또는 수정되지 않았는지), 인증서의 제약 조건이 적용되는지 확인합니다.

Domain Name System(DNS)
Domain Name System(DNS)은 인터넷이나 프라이빗 네트워크에 연결된 컴퓨터 및 기타 리소스에 대한 계층적 분산 명명 시스템입니다. DNS는 aws.amazon.com과 같은 텍스트 도메인 이름을 192.0.2.0 형태의 숫자 IP(인터넷 프로토콜) 주소로 변환하는 데 주로 사용됩니다.

도메인 이름
도메인 이름은 Domain Name System(DNS)에 의해 IP 주소로 변환될 수 있는 www.example.com 같은 텍스트 문자열입니다. 인터넷을 포함하는 컴퓨터 네트워크에는 텍스트 이름보다는 IP 주소가 사용됩니다. 도메인 이름은 마침표로 구분되는 개별 레이블로 구성됩니다.
TLD
오른쪽 끝에 있는 레이블을 TLD(최상위 도메인)라고 합니다. 일반적인 예로는 .com, .net 및 .edu가 있습니다. 또한 일부 국가에 등록된 TLD는 국가 이름의 약어로 국가 코드라고도 합니다. 예를 들면 영국의 경우 .uk, 러시아의 경우 .ru, 그리고 프랑스의 경우에는 .fr입니다. 국가 코드를 사용하는 경우 흔히 등록된 개체의 유형을 식별하기 위한 TLD에 대한 2차 수준 계층 구조가 추가됩니다. 예를 들면 .co.uk TLD는 영국의 상용 기업을 나타냅니다.

Apex 도메인
apex 도메인 이름은 최상위 도메인을 포함하고 확장합니다. 국가 코드를 포함하는 도메인 이름의 경우 apex 도메인은 있는 경우, 등록된 개체의 유형을 식별하는 코드와 레이블을 포함합니다. apex 도메인은 하위 도메인을 포함하지 않습니다(다음 단락 참조). www.example.com에서 apex 도메인의 이름은 example.com입니다. www.example.co.uk에서 apex 도메인의 이름은 example.co.uk입니다. apex 대신에 자주 사용되는 그 밖의 이름으로는 base, root, root apex 또는 zone apex가 있습니다.

하위 도메인
하위 도메인 이름은 apex 도메인 이름 앞에 붙으며 마침표로 서로 간에 구분됩니다. 가장 일반적으로 사용되는 하위 도메인 이름은 www지만, 아무 이름이나 사용할 수 있습니다. 또한 하위 도메인 이름에는 여러 수준을 지정할 수 있습니다. 예를 들면, jake.dog.animals.example.com에서 하위 도메인은 jake, dog 및 animals 순서대로 지정되어 있습니다.

FQDN
FQDN(정규화된 도메인 이름)은 네트워크 또는 인터넷에 연결된 컴퓨터, 웹 사이트 또는 기타 리소스에 대한 완전한 DNS 이름입니다. 예를 들면 aws.amazon.com은 Amazon Web Services의 FQDN입니다. FQDN에는 최상위 도메인까지의 모든 도메인이 포함됩니다. 예를 들면 [subdomain₁].[subdomain₂]...[subdomainₙ].[apex domain].[top-level domain]은 FQDN의 일반 형식을 나타냅니다.

PQDN
정규화되지 않은 도메인 이름은 PQDN(부분 정규화된 도메인 이름)이라고 하며 오클하합니다. [subdomain₁.subdomain₂] 같은 이름은 투트 도메인을 확인할 수 없으므로 PQDN입니다.

등록
도메인 이름을 사용할 수 있는 권한은 도메인 이름 등록자로부터 위임을 받습니다. 등록자는 일반적으로 국제인터넷주소관리기구(ICANN)에서 인증합니다. 또한, 등록 기관이라고 하는 그 밖의 조직도 TLD 데이터베이스를 관리합니다. 도메인 이름을 요청할 경우 등록자는 해당 사용자의 정보를 해당 TLD 등록 기관으로 전송합니다. 등록 기관은 도메인 이름을 할당하고 TLD 데이터베이스를 업데이트하며 정보를 WHOIS에 게시합니다. 일반적으로 도메인 이름은 구매해야 합니다.

최종 엔터티 인증서
최종 엔터티 인증서는 서버, 인스턴스, 컨테이너 또는 디바이스 같은 리소스를 식별합니다. CA 인증서와 달리 최종 엔터티 인증서를 사용해 인증서를 발급할 수 없습니다. 최종 엔터티 인증서의 다른 일반적인 용어는 “클라이언트” 또는 “가입자” 인증서입니다. ACM Private CA 설명서에서 최종 엔터티 인증서라는 용어를 사용합니다.

사설 CA
프라이빗 네트워크나 조직 내에서(퍼블릭 인터넷이 아닌) 사용할 인증서를 생성 및 발급합니다.
사설 인증서

ACM Private CA 인증서는 조직 내에서 사용할 수 있는 사설 SSL/TLS 인증서입니다. 사설 인증서는 클라이언트, 서버, 애플리케이션, 서비스, 디바이스 및 사용자 같은 리소스를 식별하는 데 사용됩니다. 암호화된 보안 통신 채널을 구성할 때 각 리소스가 다음과 같은 인증서를 비롯해 암호화 기법을 사용하여 다른 리소스에 자격 증명을 제공합니다. 내부 API 엔드포인트, 웹 서버, VPN 사용자, IoT 디바이스 및 다른 많은 애플리케이션에서 사설 인증서를 사용하여 보안 작업에 필요한 암호화된 통신 채널을 구성합니다. 기본적으로 사설 인증서는 공개적으로 신뢰할 수 없습니다. 내부 관리자는 사설 인증서를 신뢰하고 인증서를 배포하기 위한 애플리케이션을 명시적으로 구성해야 합니다.

Certificate:

Data:
Version: 3 (0x2)
Serial Number:
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:
X509v3 Subject Key Identifier:
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

퍼블릭 키 인프라(PKI)

퍼블릭 키 인프라(PKI)는 디지털 인증서의 생성, 발급, 관리, 배포, 사용, 저장 및 취소를 지원하는 포괄적인 시스템입니다. PKI는 사람, 하드웨어, 소프트웨어, 정책, 문서 및 절차로 구성됩니다. ACM Private CA에서 PKI의 용도는 사설 인증서를 관리하는 것입니다. ACM Private CA는 다음을 구현합니다.

퍼블릭 키 인증서

사실 CA는 조직 내의 사용자, 서버, 애플리케이션, 컴퓨터 및 기타 디바이스의 자격 증명을 확인하기 위해 인증서를 발급합니다. 자세한 내용은 사설 인증서 (p. 132) 단원을 참조하십시오.
인증서 리포지토리

사설 CA는 발급된 모든 인증서를 저장합니다. 물론 ACM Private CA 리포지토리에서 인증서를 삭제할 수 있습니다.

인증서 취소

ACM Private CA는 인증서 해지를 지원합니다. 더 이상 신뢰할 수 없는 인증서는 인증 기관이 취소해야 합니다. 어떤 이유로든 유효 기간이 끝나기 전에 인증서를 취소할 수 있습니다. 일반적인 이유로는 키 위반, 연산 종료, 소속 변경, 권한 철회 등이 있습니다. 사설 CA를 생성할 때 CA가 인증서를 지원하기를 원하는지 여부를 지정합니다. CA가 취소를 지원하는 경우에는 취소한 인증서를 검토하기 위해 감사 보고서를 요청할 수 있습니다.

키 스토리지

CA 프라이빗 키는 Amazon이 소유 및 관리하는 하드웨어 보안 모듈(HSM)에 안전하게 잠겨 있습니다.

루트 CA

인증서 발급의 기반이 되는 암호화 구성 요소 및 신뢰 루트입니다. 루트 CA는 인증서 서명(발급)을 위한 프라이빗 키와 루트 CA를 식별하고 프라이빗 키를 CA 이름에 바인딩하는 루트 인증서로 구성됩니다. 루트 인증서는 환경 내에 있는 각 엔터티의 신뢰 저장소에 배포됩니다. 관리자는 신뢰하는 CA만 포함하도록 신뢰 저장소를 구성합니다. 관리자는 해당 환경에 있는 엔터티의 운영 체제, 인스턴스 및 호스트 컴퓨터 이미지로 신뢰 저장소를 업데이트하거나 빌드합니다. 라이선스는 생산 연결을 시도할 때 각 엔터티가 제시하는 인증서를 확인합니다. 클라이언트는 인증서의 유효 기간을 검사하고 인증서에서 신뢰 저장소에 설치된 루트 인증서까지의 체인이 존재하는지 여부를 확인합니다. 이러한 조건이 충족되면 라이선스 간에 "핸드셰이크"가 수행됩니다. 이러한 핸드셰이크는 각 엔터티의 신원을 암호로 상대에게 증명하고, 이들 간에 암호화된 통신 채널(TLS/SSL)을 생성합니다.

루트 인증서

인증 기관(CA)은 일반적으로 계층 구조 내에 존재하는데, 계층 구조에는 상위-하위 관계가 명확하게 정의된 기타 여러 CA가 포함되어 있습니다. 하위 또는 중속 CA는 상위 CA에서 인증되어 인증서 체인을 생성합니다. 계층 구조의 최상위에 있는 CA를 루트 CA라고 하며 해당 인증서를 루트 인증서라고 합니다. 이 인증서는 일반적으로 자체 서명됩니다.

Secure Sockets Layer(SSL)

Secure Sockets Layer(SSL)와 TLS(전송 계층 보안)는 컴퓨터 네트워크에서 통신 보안을 제공하는 암호화 프로토콜입니다. TLS는 SSL의 후속 프로토콜입니다. 두 프로토콜 모두 X.509 인증서를 사용하여 서버를 인증합니다. 또한 클라이언트와 서버 간에 이동하는 데이터를 암호화하기 위해 대칭 키를 협상합니다.

보안 HTTPS

HTTPS는 모든 주요 브라우저와 서버에서 지원되는 보안 형식의 HTTP인 HTTP over SSL/TLS를 나타냅니다. HTTP는 HTTP 요청과 응답은 네트워크를 통해 전송되기 전에 암호화됩니다. HTTPS는 HTTP 프로토콜을 대칭, 비대칭 및 X.509 인증서 기반 암호화 기술을 중합합니다. HTTPS는 개방형 시스템 간 상호 연결(OSI) 모델에서 HTTP 애플리케이션 계층 아래 및 TCP 전송 계층 위에 암호화 보안 계층을 삽입하여 작동합니다. 보안 계층은 Secure Sockets Layer(SSL) 프로토콜 또는 TLS(전송 계층 보안) 프로토콜을 사용합니다.
자체 서명된 인증서

상위 CA 대신 발급자가 서명한 인증서입니다. CA에서 유지 관리하는 안전한 루트에서 발급된 인증서와 달리, 자체 서명된 인증서는 자체 루트의 역할을 하므로 유지 관리자에게 인증서를 요청할 때 사용할 수 있지만, ID를 확인하는 데는 사용할 수 없기 때문에 해지를 불가능하다는 심각한 한계가 있습니다. 보안 관점에서는 응답하지 않으나, 그러한 조직은 쉽게 생성할 수 있고, 전문 지식이나 인프라가 필요하지 않으므로, 많은 애플리케이션에서 사용할 수 있습니다.

자체 서명된 인증서는 자체 루트의 역할을 하므로, 유선 암호화를 제공하는 데는 사용할 수 있지만, ID를 확인하는 데는 사용할 수 없기 때문에 해지를 불가능하다는 심각한 한계가 있습니다. 보안 관점에서는 응답하지 않으나, 그러한 조직은 쉽게 생성할 수 있고, 전문 지식이나 인프라가 필요하지 않으므로, 많은 애플리케이션에서 사용할 수 있습니다.

SSL 서버 인증서

HTTPS 트랜잭션에서는 서버를 인증하기 위해 서버 인증서가 필요합니다. 서버 인증서는 인증서의 퍼블릭 키를 인증서의 주체와 바인딩하는 X.509 v3 데이터 구조입니다. SSL/TLS 인증서는 CA(인증 기관)가 서명을 합니다. 여기에는 서버 이름, 유효 기간, 퍼블릭 키, 서명 알고리즘 등이 포함됩니다.

X.500 고유 이름

X.500 고유 이름(DN)은 ACM Private CA가 생성한 X.509 퍼블릭 키 인증서에서 사용자, 컴퓨터, 애플리케이션, 서비스, 서버 및 기타 디바이스를 식별하는 데 사용됩니다. 여기에는 사용 인증서와 사용 CA 인증서가 포함됩니다. 일반 루트에서는 다음과 포함되어 있습니다.

- organizationName(O) – 인증서를 발급했거나 인증서의 주체인 조직의 이름
- organizationUnit(OU) – 조직 내의 부서 또는 사업부
- country(C) – 두 자로 된 국가 코드
- stateName(S) – 캔자스 같은 주 또는 지방의 이름
- localityName(L) – 시애틀 같은 시 이름
- commonName(CN) – 인증서 주체 또는 발행자의 일반 이름
문서 이력

다음 표에는 2018년 1월 이후 이 설명서의 중요한 변경 사항을 설명합니다. Amazon은 여기 나와 있는 주요 변경 사항 외에도 설명과 예제를 업데이트하고 고객이 제공한 피드백을 반영하도록 설명서를 자주 업데이트하고 있습니다. 중요한 변경 사항에 대해 알림을 받으려면 오른쪽 상단 모서리의 링크를 사용하여 RSS 피드를 구독합니다.

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>전용 보안 섹션 추가  (p. 135)</td>
<td>AWS에 대한 보안 설명서가 전용 보안 섹션으로 통합되었습니다. 보안에 대한 자세한 내용은 AWS Certificate Manager 사설 인증 기관의 보안을 참조하십시오.</td>
<td>March 26, 2020</td>
</tr>
<tr>
<td>VPC 엔드포인트(PrivateLink) 지원  (p. 135)</td>
<td>네트워크 보안을 강화하기 위해 VPC 엔드포인트(AWSPrivateLink) 사용에 대한 지원이 추가되었습니다. 자세한 내용은 ACM 사설 CA VPC 엔드포인트(AWS PrivateLink)를 참조하십시오.</td>
<td>March 26, 2020</td>
</tr>
<tr>
<td>템플릿 ARN이 감사 보고서에 추가되었습니다.  (p. 135)</td>
<td>자세한 내용은 사설 CA에 대한 감사 보고서 생성을 참조하십시오.</td>
<td>March 6, 2020</td>
</tr>
<tr>
<td>CloudFormation 지원  (p. 135)</td>
<td>AWS CloudFormation에 대한 지원이 추가되었습니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 ACMPCA 리소스 유형 참조를 참조하십시오.</td>
<td>January 22, 2020</td>
</tr>
<tr>
<td>CloudWatch 이벤트 통합  (p. 135)</td>
<td>CA 생성, 인증서 발급 및 CRL 생성을 포함한 비동기식 이벤트에 있어 CloudWatch 이벤트와 통합되었습니다. 자세한 내용은 CloudWatch 이벤트 사용을 참조하십시오.</td>
<td>December 23, 2019</td>
</tr>
<tr>
<td>FIPS 엔드포인트  (p. 135)</td>
<td>AWS GovCloud(미국 동부) 및 AWS GovCloud(미국 서부)에 대해 FIPS 엔드포인트가 추가되었습니다. ACM PCA 엔드포인트의 전체 목록은 AWS Certificate Manager 사설 인증 기관 엔드포인트 및 할당량을 참조하십시오.</td>
<td>December 13, 2019</td>
</tr>
<tr>
<td>태그 기반 권한  (p. 135)</td>
<td>새 API인 TagResource, UntagResource 및 ListTagsForResource를 사용한 태그 기반 권한이 지원됩니다. 태그 기반 제어에 대한 일반적인 내용은 IAM 리소스 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어를 참조하십시오.</td>
<td>November 5, 2019</td>
</tr>
<tr>
<td>질문</td>
<td>내용</td>
<td>일시</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>이름 제약 조건 적용 (p. 135)</td>
<td>가져온 CA 인증서에 보안 주체 이름 제약 조건을 적용하기 위한 지원이 추가되었습니다. 자세한 내용은 사설 CA에 이름 제약 조건 적용을 참조하십시오.</td>
<td>October 28, 2019</td>
</tr>
<tr>
<td>새 인증서 템플릿 (p. 135)</td>
<td>AWS Signer를 사용한 코드 서명용 템플릿을 포함하여 새 인증서 템플릿이 추가되었습니다. 자세한 내용은 템플릿 사용을 참조하십시오.</td>
<td>October 1, 2019</td>
</tr>
<tr>
<td>CA 계획 (p. 135)</td>
<td>ACM PCA를 사용한 PKI 계획에 대한 새로운 섹션이 추가되었습니다. 자세한 내용은 ACM 사설 CA 배포 계획을 참조하십시오.</td>
<td>September 30, 2019</td>
</tr>
<tr>
<td>추가된 리전 지원 (p. 135)</td>
<td>AWS 아시아 태평양(홍콩) 리전에 대한 지원이 추가되었습니다. 지원되는 리전의 전체 목록은 AWS Certificate Manager 사설 인증 기관 엔드포인트 및 할당량을 참조하십시오.</td>
<td>July 24, 2019</td>
</tr>
<tr>
<td>사설 CA 계층 구조에 대한 전체 지원 추가 (p. 135)</td>
<td>루트 CA 생성 및 호스팅이 지원되기 때문에 외부 상위 요소가 필요하지 않습니다.</td>
<td>June 20, 2019</td>
</tr>
<tr>
<td>추가된 리전 지원 (p. 135)</td>
<td>AWS GovCloud(미국 서부 및 미국 동부) 리전에 대한 지원이 추가되었습니다. 지원되는 리전의 전체 목록은 AWS Certificate Manager 사설 인증 기관 엔드포인트 및 할당량을 참조하십시오.</td>
<td>May 8, 2019</td>
</tr>
<tr>
<td>추가된 리전 지원 (p. 135)</td>
<td>AWS 아시아 태평양(몰바이 및 서울, 미국 서부(캘리포니아 북부) 및 EU(파리 및 스톡홀름) 리전에 대한 지원이 추가되었습니다. 지원되는 리전의 전체 목록은 AWS Certificate Manager 사설 인증 기관 엔드포인트 및 할당량을 참조하십시오.</td>
<td>April 4, 2019</td>
</tr>
<tr>
<td>인증서 생성 워크플로 테스트 (p. 135)</td>
<td>이제 고객은 ACM 관리형 생성 워크플로우의 구성요소를 수동으로 테스트할 수 있습니다. 자세한 내용은 ACM의 관리형 생성 구성 테스트를 참조하십시오.</td>
<td>March 14, 2019</td>
</tr>
<tr>
<td>추가된 리전 지원 (p. 135)</td>
<td>AWS EU(러던) 리전에 대한 지원이 추가되었습니다. 지원되는 리전의 전체 목록은 AWS Certificate Manager 사설 인증 기관 엔드포인트 및 할당량을 참조하십시오.</td>
<td>August 1, 2018</td>
</tr>
</tbody>
</table>
이전 업데이트


<table>
<thead>
<tr>
<th>변경 사항</th>
<th>설명</th>
<th>날짜</th>
</tr>
</thead>
<tbody>
<tr>
<td>새 가이드</td>
<td>이 릴리스는 AWS Certificate Manager Private Certificate Authority를 도입했습니다.</td>
<td>2018년 04월 4일</td>
</tr>
</tbody>
</table>