User Guide

AWS Certificate Manager



Version 1.0

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Certificate Manager: User Guide

Copyright © 2026 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

What is AWS Certificate Manager? 1
Supported Regions 1
Pricing 2
Concepts 2
Automated Certificate Management Environment (ACME) 3
ACM Certificate 3
ACM Root CAs 5
Apex Domain 6
Asymmetric Key Cryptography 6
Certificate Authority 6
Certificate Transparency Logging 7
Domain Name System 8
Domain Names 8
Encryption and Decryption 9
Fully Qualified Domain Name (FQDN) 9
Hypertext Transfer Protocol (HTTP) 9
Public Key Infrastructure (PKI) 10
Root Certificate 11
Secure Sockets Layer (SSL) 11
Secure HTTPS 11
SSL Server Certificates 11
Symmetric Key Cryptography 11
Transport Layer Security (TLS) 11
Trust 12
Wildcard Certificate 12
Choosing how to issue certificates with AWS 12
Getting started 15
Set up 16
Sign up for an AWS account 16
Register a domain name 16
(Optional) Configure a CAA record 16
Types of certificates 20
Choose a certificate type 20
Compare certificate types 22

How ACM identifies certificates ... 23
Public certificates ... 24
Characteristics and limitations ... 25
Request a public certificate ... 30
Exportable public certificates ... 35
Certificate validation ... 47
Private certificates ... 70
Conditions for use ... 71
Request a private certificate ... 72
Export certificate ... 75
Imported certificates ... 78
Prerequisites ... 79
Certificate format ... 81
Import certificate ... 83
Reimport certificate ... 84
Amazon N2 certificates ... 87
How it works ... 88
Roots and trust ... 89
Compare root hierarchies ... 90
When to use Amazon N2 certificates ... 91
Certificate characteristics ... 92
Request a certificate ... 94
Certificate lifecycle automation ... 97
Managed certificate renewal ... 98
Public certificates ... 99
Private certificates ... 103
Check renewal status ... 106
Managed automation with integrated services ... 109
ACME certificate automation ... 114
What is ACME? ... 115
How it works ... 115
Certificate characteristics ... 116
Working with certificates ... 117
When to use ACME ... 117
Persona model ... 118
Workflow overview ... 118

ACME endpoints ... 119
ACME domain validation ... 124
ACME external account bindings ... 129
Issuing certificates ... 134
Issuing ACME certificates over AWS PrivateLink ... 138
Workload Credentials Provider ... 141
How certificate automation works ... 142
Prerequisites ... 142
Install the provider ... 143
Configure the provider ... 146
Required permissions ... 149
Verify the installation ... 150
Configuration reference ... 151
Logging ... 152
Certificate management ... 154
Search certificates ... 154
View certificate details ... 159
Delete certificates ... 163
Tag resources ... 164
Tag restrictions ... 164
Managing tags ... 165
Managing tags (console) ... 165
Managing tags (CLI) ... 167
Manage tags ... 167
Security ... 169
Data protection ... 169
Security for certificate private keys ... 170
Identity and Access Management ... 171
Audience ... 172
Authenticating with identities ... 172
Managing access using policies ... 174
How AWS Certificate Manager works with IAM ... 175
Identity-based policy examples ... 180
ACM API permissions reference ... 186
AWS managed policies ... 188
Use condition keys ... 191

Condition keys for ACME ... 198
Use service-linked roles ... 203
IAM for ACME ... 206
Troubleshooting ... 209
Resilience ... 211
Infrastructure security ... 212
Granting programmative access to ACM ... 212
Monitor and log ... 215
Amazon EventBridge ... 215
Supported events ... 215
Example actions ... 221
CloudTrail ... 231
Supported API actions ... 232
API calls for integrated services ... 295
CloudWatch metrics ... 300
Use AWS Certificate Manager with the SDK for Java ... 302
AddTagsToCertificate ... 303
CreateAcmeDomainValidation ... 305
CreateAcmeEndpoint ... 306
CreateAcmeExternalAccountBinding ... 307
DeleteAcmeDomainValidation ... 309
DeleteAcmeEndpoint ... 309
DeleteAcmeExternalAccountBinding ... 310
DeleteCertificate ... 311
DescribeAcmeAccount ... 313
DescribeAcmeDomainValidation ... 313
DescribeAcmeEndpoint ... 314
DescribeAcmeExternalAccountBinding ... 315
DescribeCertificate ... 316
ExportCertificate ... 318
GetAcmeExternalAccountBindingCredentials ... 321
GetCertificate ... 322
ImportCertificate ... 325
ListAcmeAccounts ... 328
ListAcmeDomainValidations ... 329
ListAcmeEndpoints ... 330

ListAcmeExternalAccountBindings 330
ListCertificates 331
ListTagsForCertificate 334
ListTagsForResource 336
RemoveTagsFromCertificate 336
RenewCertificate 339
RequestCertificate 341
ResendValidationEmail 343
RevokeAcmeAccount 345
RevokeAcmeExternalAccountBinding 346
SearchCertificates 347
UpdateCertificateOptions 351
TagResource 353
ListCertificateDomainValidations 354
UntagResource 356
UpdateAcmeDomainValidation 357
UpdateAcmeEndpoint 358
Troubleshoot 359
Certificate requests 359
Request times out 359
Request fails 360
Certificate validation 361
DNS validation 362
Email validation 365
HTTP validation 367
Certificate renewal 368
Preparing for automatic domain validation 368
Handling failures in managed certificate renewal 369
Managed certificate renewal for email-validated certificates 369
Managed certificate renewal for DNS-validated certificates 369
Managed certificate renewal for HTTP-validated certificates 371
Understanding renewal timing 372
ACME automation 372
ACME failures not appearing in the console Monitoring tab 372
Domain validation does not become valid 373
Certificate issuance or revocation fails with access denied 373

Certificate request is rejected 374
Certificate issuance fails with a DNS CNAME error after the domain validation was previously valid 374
Account registration fails 375
ACME client times out waiting for certificate 376
Handling exceptions 376
Private certificate exception handling 376
Other problems 378
CAA records 379
Certificate import 379
Certificate pinning 380
API Gateway 381
Unexpected failure 381
Problems with the ACM service-linked role (SLR) 382
Quotas 383
General quotas 383
API rate quotas 387
ACME protocol request rate quotas 389
Best practices 391
Account-level separation 391
AWS CloudFormation 392
Custom Trust Stores 393
Certificate pinning 393
Domain validation 394
Adding or deleting domain names 394
Domain name privacy 395
Turn on AWS CloudTrail 395
Document history 396

What is AWS Certificate Manager?

AWS Certificate Manager (ACM) handles the complexity of creating, storing, and renewing public and private SSL/TLS X.509 certificates and keys that protect your AWS websites and applications. You can provide certificates for your integrated AWS services either by issuing them directly with ACM or by importing third-party certificates into the ACM management system. ACM certificates can secure singular domain names, multiple specific domain names, wildcard domains, or combinations of these. ACM wildcard certificates can protect an unlimited number of subdomains. You can also export ACM certificates signed by AWS Private CA for use anywhere in your internal PKI.

> **Note**
>
> You can use ACM certificates with stand-alone web servers and other customer-managed infrastructure, including Amazon EC2 instances. For publicly trusted web certificates on these servers, use the ACME protocol to automate issuance and renewal directly on your hosts. See ACME certificate automation. For private PKI scenarios, the following tutorial has instructions for setting up a secure server on an Amazon EC2 instance: Configure SSL/TLS on Amazon Linux 2023.

Topics

- Supported Regions
- Pricing for AWS Certificate Manager
- AWS Certificate Manager concepts
- Choosing how to issue certificates with AWS

Supported Regions

ACM supports IPv4 and IPv6 on public endpoints. Visit AWS Regions and Endpoints in the *AWS General Reference* or the AWS Region Table to see the regional availability for ACM.

Certificates in ACM are regional resources. To use a certificate with Elastic Load Balancing for the same fully qualified domain name (FQDN) or set of FQDNs in more than one AWS region, you must request or import a certificate for each region. For certificates provided by ACM, this means you

must revalidate each domain name in the certificate for each region. You cannot copy a certificate between regions.

To use an ACM certificate with Amazon CloudFront, you must request or import the certificate in the US East (N. Virginia) region. ACM certificates in this region that are associated with a CloudFront distribution are distributed to all the geographic locations configured for that distribution.

Pricing for AWS Certificate Manager

You are not subject to an additional charge for SSL/TLS certificates that you manage with AWS Certificate Manager. You pay only for the AWS resources that you create to run your website or application. For the latest ACM pricing information, see the AWS Certificate Manager Service Pricing page on the AWS website.

AWS Certificate Manager concepts

This section provides definitions of concepts used by AWS Certificate Manager.

Topics

- Automated Certificate Management Environment (ACME)
- ACM Certificate
- ACM Root CAs
- Apex Domain
- Asymmetric Key Cryptography
- Certificate Authority
- Certificate Transparency Logging
- Domain Name System
- Domain Names
- Encryption and Decryption
- Fully Qualified Domain Name (FQDN)
- Hypertext Transfer Protocol (HTTP)
- Public Key Infrastructure (PKI)
- Root Certificate
- Secure Sockets Layer (SSL)

- Secure HTTPS
- SSL Server Certificates
- Symmetric Key Cryptography
- Transport Layer Security (TLS)
- Trust
- Wildcard Certificate

Automated Certificate Management Environment (ACME)

The Automated Certificate Management Environment (ACME) is a protocol defined in RFC 8555 for automating certificate issuance and lifecycle management through machine-to-machine communication between certificate requesters (called ACME clients) and Certificate Authorities. ACME eliminates manual steps in certificate provisioning by enabling clients such as Certbot and cert-manager to request, validate, and install certificates without human intervention.

ACM provides a managed ACME server implementation for automating public certificate issuance on customer-managed infrastructure. For more information, see ACME certificate automation.

ACM Certificate

ACM generates X.509 version 3 certificates. Each public certificate is valid for 198 days and contains the following extensions. The validity of an Amazon N2 certificate depends on its root hierarchy. For more information, see Validity period.

- **Basic Constraints**- specifies whether the subject of the certificate is a certification authority (CA)
- **Authority Key Identifier**- enables identification of the public key corresponding to the private key used to sign the certificate.
- **Subject Key Identifier**- enables identification of certificates that contain a particular public key.
- **Key Usage**- defines the purpose of the public key embedded in the certificate.
- **Extended Key Usage**- specifies one or more purposes for which the public key may be used in addition to the purposes specified by the **Key Usage** extension.

> **Important**
>
> Effective June 11, 2025 AWS Certificate Manager no longer issues certificates with the "TLS Web Client Authentication" (clientAuth) extended key usage (EKU) to align with new

browser requirements for website certificates. This applies to ACM public certificates. To obtain a certificate that carries the clientAuth EKU for authentication between workloads, request an Amazon N2 certificate. For more information, see Amazon N2 certificates.

- **CRL Distribution Points**- specifies where CRL information can be obtained.

The plaintext of an ACM-issued certificate resembles the following example:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            f2:16:ad:85:d8:42:d1:8a:3f:33:fa:cc:c8:50:a8:9e
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: O=Example CA
        Validity
            Not Before: Jan 30 18:46:53 2018 GMT
            Not After : Jan 31 19:46:53 2018 GMT
        Subject: C=US, ST=VA, L=Herndon, O=Amazon, OU=AWS, CN=example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:ba:a6:8a:aa:91:0b:63:e8:08:de:ca:e7:59:a4:
                    69:4c:e9:ea:26:04:d5:31:54:f5:ec:cb:4e:af:27:
                    e3:94:0f:a6:85:41:6b:8e:a3:c1:c8:c0:3f:1c:ac:
                    a2:ca:0a:b2:dd:7f:c0:57:53:0b:9f:b4:70:78:d5:
                    43:20:ef:2c:07:5a:e4:1f:d1:25:24:4a:81:ab:d5:
                    08:26:73:f8:a6:d7:22:c2:4f:4f:86:72:0e:11:95:
                    03:96:6d:d5:3f:ff:18:a6:0b:36:c5:4f:78:bc:51:
                    b5:b6:36:86:7c:36:65:6f:2e:82:73:1f:c7:95:85:
                    a4:77:96:3f:c0:96:e2:02:94:64:f0:3a:df:e0:76:
                    05:c4:56:a2:44:72:6f:8a:8a:a1:f3:ee:34:47:14:
                    bc:32:f7:50:6a:e9:42:f5:f4:1c:9a:7a:74:1d:e5:
                    68:09:75:19:4b:ac:c6:33:90:97:8c:0d:d1:eb:8a:
                    02:f3:3e:01:83:8d:16:f6:40:39:21:be:1a:72:d8:
                    5a:15:68:75:42:3e:f0:0d:54:16:ed:9a:8f:94:ec:
                    59:25:e0:37:8e:af:6a:6d:99:0a:8d:7d:78:0f:ea:
                    40:6d:3a:55:36:8e:60:5b:d6:0d:b4:06:a3:ac:ab:
                    e2:bf:c9:b7:fe:22:9e:2a:f6:f3:42:bb:94:3e:b7:
                    08:73
```

```
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Authority Key Identifier:
                keyid:84:8C:AC:03:A2:38:D9:B6:81:7C:DF:F1:95:C3:28:31:D5:F7:88:42
            X509v3 Subject Key Identifier:
                97:06:15:F1:EA:EC:07:83:4C:19:A9:2F:AF:BA:BB:FC:B2:3B:55:D8
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:http://example.com/crl

    Signature Algorithm: sha256WithRSAEncryption
         69:03:15:0c:fb:a9:39:a3:30:63:b2:d4:fb:cc:8f:48:a3:46:
         69:60:a7:33:4a:f4:74:88:c6:b6:b6:b8:ab:32:c2:a0:98:c6:
         8d:f0:8f:b5:df:78:a1:5b:02:18:72:65:bb:53:af:2f:3a:43:
         76:3c:9d:d4:35:a2:e2:1f:29:11:67:80:29:b9:fe:c9:42:52:
         cb:6d:cd:d0:e2:2f:16:26:19:cd:f7:26:c5:dc:81:40:3b:e3:
         d1:b0:7e:ba:80:99:9a:5f:dd:92:b0:bb:0c:32:dd:68:69:08:
         e9:3c:41:2f:15:a7:53:78:4d:33:45:17:3e:f2:f1:45:6b:e7:
         17:d4:80:41:15:75:ed:c3:d4:b5:e3:48:8d:b5:0d:86:d4:7d:
         94:27:62:84:d8:98:6f:90:1e:9c:e0:0b:fa:94:cc:9c:ee:3a:
         8a:6e:6a:9d:ad:b8:76:7b:9a:5f:d1:a5:4f:d0:b7:07:f8:1c:
         03:e5:3a:90:8c:bc:76:c9:96:f0:4a:31:65:60:d8:10:fc:36:
         44:8a:c1:fb:9c:33:75:fe:a6:08:d3:89:81:b0:6f:c3:04:0b:
         a3:04:a1:d1:1c:46:57:41:08:40:b1:38:f9:57:62:97:10:42:
         8e:f3:a7:a8:77:26:71:74:c2:0a:5b:9e:cc:d5:2c:c5:27:c3:
         12:b9:35:d5
```

ACM Root CAs

The public end-entity certificates issued by ACM derive their trust from the following Amazon root CAs:

Distinguished name	Encryption algorithm
CN=Amazon Root CA 1,O=Amazon,C=US	2048-bit RSA (RSA_2048)
CN=Amazon Root CA 2,O=Amazon,C=US	4096-bit RSA (RSA_4096)
CN=Amazon Root CA 3,O=Amazon,C=US	Elliptic Prime Curve 256 bit (EC_prime256v1)
CN=Amazon Root CA 4,O=Amazon,C=US	Elliptic Prime Curve 384 bit (EC_secp384r1)

The default root of trust for ACM-issued certificates is CN=Amazon Root CA 1,O=Amazon,C=US, which offers 2048-bit RSA security. The other roots are reserved for future use. Amazon Root CA 1 to 4 are cross-signed by the Starfield Services Root Certificate Authority certificate. Amazon N2 certificates chain to separate roots. For more information, see Amazon N2 roots and trust.

For more information, see Amazon Trust Services.

Apex Domain

See Domain Names.

Asymmetric Key Cryptography

Unlike Symmetric Key Cryptography, asymmetric cryptography uses different but mathematically related keys to encrypt and decrypt content. One of the keys is public and is typically made available in an X.509 v3 certificate. The other key is private and is stored securely. The X.509 certificate binds the identity of a user, computer, or other resource (the certificate subject) to the public key.

ACM certificates are X.509 SSL/TLS certificates that bind the identity of your website and the details of your organization to the public key that is contained in the certificate. ACM uses your AWS KMS key to encrypt the private key. For more information, see Security for certificate private keys.

Certificate Authority

A certificate authority (CA) is an entity that issues digital certificates. Commercially, the most common type of digital certificate is based on the ISO X.509 standard. The CA issues signed digital

certificates that affirm the identity of the certificate subject and bind that identity to the public key contained in the certificate. A CA also typically manages certificate revocation.

Certificate Transparency Logging

To guard against SSL/TLS certificates that are issued by mistake or by a compromised CA, some browsers require that public certificates issued for your domain be recorded in a certificate transparency log. The domain name is recorded. The private key is not. Certificates that are not logged typically generate an error in the browser.

You can monitor the logs to make sure that only certificates you have authorized have been issued for your domain. You can use a service such as Certificate Search to check the logs.

Before the Amazon CA issues a publicly trusted SSL/TLS certificate for your domain, it submits the certificate to at least three certificate transparency log servers. These servers add the certificate to their public databases and return a signed certificate timestamp (SCT) to the Amazon CA. The CA then embeds the SCT in the certificate, signs the certificate, and issues it to you. The timestamps are included with other X.509 extensions.

```
X509v3 extensions:

  CT Precertificate SCTs:
    Signed Certificate Timestamp:
      Version   : v1(0)
        Log ID    : BB:D9:DF:...8E:1E:D1:85
        Timestamp : Apr 24 23:43:15.598 2018 GMT
        Extensions: none
        Signature : ecdsa-with-SHA256
                    30:45:02:...18:CB:79:2F
    Signed Certificate Timestamp:
      Version   : v1(0)
        Log ID    : 87:75:BF:...A0:83:0F
        Timestamp : Apr 24 23:43:15.565 2018 GMT
        Extensions: none
        Signature : ecdsa-with-SHA256
                    30:45:02:...29:8F:6C
```

All ACM public certificates are automatically recorded in certificate transparency logs. Per browser policy requirements, you cannot opt out of certificate transparency logging. ACM does not record

Amazon N2 certificates in certificate transparency logs. For more information, see Certificate transparency logging.

Domain Name System

The Domain Name System (DNS) is a hierarchical distributed naming system for computers and other resources connected to the internet or a private network. DNS is primarily used to translate textual domain names, such as `aws.amazon.com`, into numerical IP (Internet Protocol) addresses of the form `111.122.133.144`. The DNS database for your domain, however, contains a number of records that can be used for other purposes. For example, with ACM you can use a CNAME record to validate that you own or control a domain when you request a certificate. For more information, see AWS Certificate Manager DNS validation.

Domain Names

A domain name is a text string such as `www.example.com` that can be translated by the Domain Name System (DNS) into an IP address. Computer networks, including the internet, use IP addresses rather than text names. A domain name consists of distinct labels separated by periods:

TLD

The rightmost label is called the top-level domain (TLD). Common examples include `.com`, `.net`, and `.edu`. Also, the TLD for entities registered in some countries is an abbreviation of the country name and is called a country code. Examples include `.uk` for the United Kingdom, `.ru` for Russia, and `.fr` for France. When country codes are used, a second-level hierarchy for the TLD is often introduced to identify the type of the registered entity. For example, the `.co.uk` TLD identifies commercial enterprises in the United Kingdom.

Apex domain

The apex domain name includes and expands on the top-level domain. For domain names that include a country code, the apex domain includes the code and the labels, if any, that identify the type of the registered entity. The apex domain does not include subdomains (see the following paragraph). In `www.example.com`, the name of the apex domain is `example.com`. In `www.example.co.uk`, the name of the apex domain is `example.co.uk`. Other names that are often used instead of apex include base, bare, root, root apex, or zone apex.

Subdomain

Subdomain names precede the apex domain name and are separated from it and from each other by a period. The most common subdomain name is `www`, but any name is possible. Subdomain names can also have multiple levels. For example, in `jake.dog.animals.example.com`, the subdomains are `jake`, dog, and `animals` in that order.

Superdomain

The domain to which a subdomain belongs.

FQDN

A fully qualified domain name (FQDN) is the complete DNS name for a computer, website, or other resource connected to a network or to the internet. For example `aws.amazon.com` is the FQDN for Amazon Web Services. An FQDN includes all domains up to the top–level domain. For example, $[subdomain_1].[subdomain_2]...[subdomain_n].[apex\ domain].[top\text{–}level\ domain]$ represents the general format of an FQDN.

PQDN

A domain name that is not fully qualified is called a partially qualified domain name (PQDN) and is ambiguous. A name such as $[subdomain_1.subdomain_2.]$ is a PQDN because the root domain cannot be determined.

Encryption and Decryption

Encryption is the process of providing data confidentiality. Decryption reverses the process and recovers the original data. Unencrypted data is typically called plaintext whether it is text or not. Encrypted data is typically called ciphertext. HTTPS encryption of messages between clients and servers uses algorithms and keys. Algorithms define the step-by-step procedure by which plaintext data is converted into ciphertext (encryption) and ciphertext is converted back into the original plaintext (decryption). Keys are used by algorithms during the encryption or decryption process. Keys can be either private or public.

Fully Qualified Domain Name (FQDN)

See Domain Names.

Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) is the foundation of data communication on the World Wide Web. It's an application-layer protocol that enables the exchange of various content types.

HTTP operates on a client-server model, where web browsers typically act as clients requesting resources from web servers. As a stateless protocol, HTTP treats each request independently, without retaining information from previous requests.

In the context of ACM, HTTP can be used for domain validation when issuing SSL/TLS certificates. This process involves ACM sending specific HTTP requests to verify domain ownership. The server's ability to respond correctly to these requests demonstrates control over the domain.

Unlike DNS-validated certificates, you can't issue HTTP-validated certificates directly from ACM. Instead, these certificates are automatically issued and managed as part of the CloudFront provisioning process. Customers can use ACM to view, monitor, and manage these certificates, but the initial issuance is handled by the integration between ACM and CloudFront.

While HTTP is widely used, it's important to note that it transmits data in plain text. For secure communication, HTTPS (HTTP Secure) is used, which encrypts the data using SSL/TLS protocols. For more information on secure communications, see Secure HTTPS.

Public Key Infrastructure (PKI)

Public Key Infrastructure (PKI) is a system of processes, technologies, and policies that enables secure communication over public networks. In the context of ACM, PKI plays a crucial role in the issuance, management, and validation of digital certificates. PKI uses a pair of cryptographic keys: a public key that is freely distributed, and a private key that is kept secret by the owner. This system allows for secure data transmission, digital signatures, and authentication of digital entities.

ACM implements several key components of PKI. It acts as a Certificate Authority (CA), a trusted third party that issues digital certificates, binding public keys to entities such as domains or organizations. ACM issues X.509 certificates, which contain information about the entity, its public key, and the certificate's validity period. It also handles the complete lifecycle of certificates, including issuance, renewal, and revocation. To ensure the legitimacy of certificate requests, ACM supports various methods to validate domain ownership, such as DNS validation and HTTP validation.

By leveraging PKI, ACM enables secure HTTPS connections, digital signatures, and encrypted communication for AWS resources and applications. This infrastructure is essential for maintaining the confidentiality, integrity, and authenticity of data transmitted over the internet. For more information on how ACM implements PKI, see Getting started with AWS Certificate Manager certificates.

Root Certificate

A certificate authority (CA) typically exists within a hierarchical structure that contains multiple other CAs with clearly defined parent-child relationships between them. Child or subordinate CAs are certified by their parent CAs, creating a certificate chain. The CA at the top of the hierarchy is referred to as the root CA, and its certificate is called the root certificate. This certificate is typically self-signed.

Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are cryptographic protocols that provide communication security over a computer network. TLS is the successor of SSL. They both use X.509 certificates to authenticate the server. Both protocols negotiate a symmetric key between the client and the server that is used to encrypt data flowing between the two entities.

Secure HTTPS

HTTPS stands for HTTP over SSL/TLS, a secure form of HTTP that is supported by all major browsers and servers. All HTTP requests and responses are encrypted before being sent across a network. HTTPS combines the HTTP protocol with symmetric, asymmetric, and X.509 certificate-based cryptographic techniques. HTTPS works by inserting a cryptographic security layer below the HTTP application layer and above the TCP transport layer in the Open Systems Interconnection (OSI) model. The security layer uses the Secure Sockets Layer (SSL) protocol or the Transport Layer Security (TLS) protocol.

SSL Server Certificates

HTTPS transactions require server certificates to authenticate a server. A server certificate is an X.509 v3 data structure that binds the public key in the certificate to the subject of the certificate. An SSL/TLS certificate is signed by a certificate authority (CA) and contains the name of the server, the validity period, the public key, the signature algorithm, and more.

Symmetric Key Cryptography

Symmetric key cryptography uses the same key to both encrypt and decrypt digital data. See also Asymmetric Key Cryptography.

Transport Layer Security (TLS)

See Secure Sockets Layer (SSL).

Trust

In order for a web browser to trust the identity of a website, the browser must be able to verify the website's certificate. Browsers, however, trust only a small number of certificates known as CA root certificates. A trusted third party, known as a certificate authority (CA), validates the identity of the website and issues a signed digital certificate to the website's operator. The browser can then check the digital signature to validate the identity of the website. If validation is successful, the browser displays a lock icon in the address bar.

Wildcard Certificate

A wildcard certificate secures multiple subdomains at the same level by using an asterisk (*) in the leftmost position of the domain name. For example, a certificate for `*.example.com` protects `login.example.com` and `test.example.com`. A wildcard protects only one subdomain level: it does not protect the apex domain `example.com`, or multi-level subdomains such as `test.login.example.com`.

Choosing how to issue certificates with AWS

AWS offers several ways to issue and manage X.509 certificates. Choose the option that best fits your use case.

	ACM	ACM with ACME	AWS Private CA (direct issuance)
Best for	Public or private certificates for AWS integrated services (Elastic Load Balancing, CloudFront, API Gateway)	Public certificates for customer-managed infrastructure (on-premises, Kubernetes, hybrid)	Private certificates where you supply the CSR and manage the private key yourself
Certificate trust	Public (Amazon Trust Services) or private (via AWS Private CA integration)	Public (Amazon Trust Services)	Private (your CA hierarchy)

	ACM	ACM with ACME	AWS Private CA (direct issuance)
Private key management	AWS generates and manages the private key (you can export it for use outside AWS)	Your ACME client generates and holds the private key	You generate and hold the private key
Renewal	ACM managed renewal (automatic)	Client-driven (your ACME client renews before expiry)	Manual (you call `IssueCertificate` again)
Deployment	Bound to AWS integrated services, or exported for use anywhere	Installed on your systems by the ACME client	You install the certificate on your systems
Automation	AWS SDK and CLI	Industry-standard ACME clients (Certbot, cert-manager, acme.sh)	AWS SDK and CLI

To issue certificates for mutual TLS authentication between workloads, see Amazon N2 certificates.

AWS Certificate Manager (ACM)

Use ACM when you need certificates for AWS integrated services such as Elastic Load Balancing, Amazon CloudFront, and Amazon API Gateway, or when you want AWS to manage the certificate lifecycle including renewal. ACM generates and manages the private key and automates renewal. For workloads outside of AWS integrated services, ACM also supports exportable certificates that let you retrieve the private key and use the certificate on your own infrastructure. ACM can issue public certificates from Amazon Trust Services or private certificates when integrated with AWS Private CA.

ACM with ACME certificate automation

Use ACME when you need publicly trusted certificates for customer-managed infrastructure and want to automate the lifecycle using industry-standard ACME clients (Certbot, cert-manager, acme.sh) rather than AWS APIs. The private key is generated and held by your ACME client and never leaves your systems. Certificates issued through ACME appear in your ACM inventory for

central visibility but cannot be bound to AWS integrated services. For more information, see ACME certificate automation.

AWS Private CA (direct issuance)

Use AWS Private CA directly when you need private certificates and want full control over the private key. You create your own CA hierarchy, generate your own private key and CSR, and call `IssueCertificate`. Certificates issued by a private CA are not publicly trusted and cannot be used on the public internet. For more information, see the AWS Private CA User Guide.

Both ACM and ACME certificate automation are covered in this guide. You are in the right place.

Getting started with AWS Certificate Manager certificates

ACM manages public, private, and imported certificates. Certificates are used to establish secure communications across the internet or within an internal network. You can request a publicly trusted certificate directly from ACM (an "ACM certificate"), import a publicly trusted certificate issued by a third party. You can also request an Amazon N2 certificate for authentication between workloads. For more information, see Amazon N2 certificates. Self-signed certificates are also supported. To provision your organization's internal PKI, you can issue ACM certificates signed by a private certificate authority (CA) created and managed by AWS Private CA. The CA may either reside in your account or be shared with you by a different account.

You can also automate public certificate issuance using the Automated Certificate Management Environment (ACME) protocol for workloads running on customer-managed infrastructure. For more information, see ACME certificate automation.

> **Note**
>
> To automate public certificate issuance and renewal outside integrated services such as on Amazon EC2 instances, use the ACME protocol. For more information, see ACME certificate automation. Alternatively, if you cannot use ACME, you can automate exportable certificates issued from ACM through AWS Workload Credentials Provider.

> **Note**
>
> Because certificates signed by a private CA are not trusted by default, administrators must install them in client trust stores.

To begin issuing certificates, sign into the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/home. If the introductory page appears, choose **Get Started**. Otherwise, choose **Certificate Manager** or **Private CAs** in the left navigation pane.

Topics

- Set up to use AWS Certificate Manager

Set up to use AWS Certificate Manager

With AWS Certificate Manager (ACM) you can provision and manage SSL/TLS certificates for your AWS based websites and applications. You use ACM to create or import and then manage a certificate. You must use other AWS services to deploy the certificate to your website or application. For more information about the services integrated with ACM, see Managed automation with integrated services. The following sections discuss the steps you need to perform before using ACM.

Topics

- Sign up for an AWS account
- Register a domain name for ACM
- (Optional) Configure a CAA record

Sign up for an AWS account

To get started with AWS, you need an AWS account. There are two ways to sign up for AWS. If you want complete control over account configuration and where your content is stored, or if you have regulated workloads such as HIPAA or FedRAMP, use Sign up for AWS (advanced). If you are a new user and want to use a login you already own to sign in to AWS and create an AWS environment with preconfigured defaults in select AWS Regions, use Sign up for AWS.

Register a domain name for ACM

A fully qualified domain name (FQDN) is the unique name of an organization or individual on the Internet followed by a top-level domain extension such as `.com` or `.org`. If you do not already have a registered domain name, you can register one through Amazon Route 53 or dozens of other commercial registrars. Typically you go to the registrar's website and request a domain name. Domain name registration usually lasts for a set period of time such as one or two years before it must be renewed.

For more information about registering domain names with Amazon Route 53, see Registering Domain Names Using Amazon Route 53 in the *Amazon Route 53 Developer Guide*.

(Optional) Configure a CAA record

A CAA record specifies which certificate authorities (CAs) are allowed to issue certificates for a domain or subdomain. Creating a CAA record for use with ACM helps to prevent the wrong

CAs from issuing certificates for your domains. A CAA record isn't a substitute for the security requirements that are specified by your certificate authority, such as the requirement to validate that you're the owner of a domain.

After ACM validates your domain during the certificate request process, it checks for the presence of a CAA record to make sure it can issue a certificate for you. Configuring a CAA record is optional.

Use the following values when you configure your CAA record:

flags

Specifies whether the value of the **tag** field is supported by ACM. Set this value to **0**.

tag

The **tag** field can be one of the following values. Note that the **iodef** field is currently ignored.

issue

Indicates that the ACM CA that you specify in the **value** field is authorized to issue a certificate for your domain or subdomain.

issuewild

Indicates that the ACM CA that you specified in the **value** field is authorized to issue a wildcard certificate for your domain or subdomain. A wildcard certificate applies to the domain or subdomain and all of its subdomains. Note that if you plan to use HTTP validation, this setting won't apply because HTTP validation doesn't support wildcard certificates. Use DNS validation instead for wildcard certificates.

value

The value of this field depends on the value of the **tag** field. You must enclose this value in quotation marks ("").

When **tag** is **issue**

The **value** field contains the CA domain name. This field can contain the name of a CA other than an Amazon CA. However, if you do not have a CAA record that specifies one of the following four Amazon CAs, ACM cannot issue a certificate to your domain or subdomain:

- amazon.com
- amazontrust.com
- awstrust.com

- amazonaws.com

The **value** field can also contain a semicolon (;) to indicate that no CA should be permitted to issue a certificate for your domain or subdomain. Use this field if you decide at some point that you no longer want a certificate issued for a particular domain.

When **tag** is **issuewild**

The **value** field is the same as that for when **tag** is **issue** except that the value applies to wildcard certificates.

When there is an **issuewild** CAA record present that does not include an ACM CA value, then no wild cards can be issued by ACM. If there is no **issuewild** present, but there is an **issue** CAA record for ACM, then wild cards may be issued by ACM.

Example CAA Record Examples

In the following examples, your domain name comes first followed by the record type (CAA). The **flags** field is always 0. The **tags** field can be **issue** or **issuewild**. If the field is **issue** and you type the domain name of a CA server in the **value** field, the CAA record indicates that your specified server is permitted to issue your requested certificate. If you type a semicolon ";" in the **value** field, the CAA record indicates that no CA is permitted to issue a certificate. The configuration of CAA records varies by DNS provider.

> **Important**
>
> If you plan to use HTTP validation with CloudFront, you don't need to configure **issuewild** records because HTTP validation doesn't support wildcard certificates. For wildcard certificates, use DNS validation instead.

```
Domain       Record type  Flags  Tag      Value
example.com.   CAA            0        issue      "SomeCA.com"
```

```
Domain       Record type  Flags  Tag      Value
example.com.   CAA            0        issue      "amazon.com"
```

```
Domain       Record type  Flags  Tag      Value
```

```
example.com.   CAA            0        issue      "amazontrust.com"
```

```
Domain       Record type  Flags  Tag      Value
example.com.   CAA            0        issue      "awstrust.com"
```

```
Domain       Record type  Flags  Tag      Value
example.com.   CAA            0        issue      "amazonaws.com"
```

```
Domain       Record type  Flags  Tag      Value
example.com    CAA            0        issue      ";"
```

For more information about how to add or modify DNS records, check with your DNS provider. Route 53 supports CAA records. If Route 53 is your DNS provider, see CAA Format for more information about creating a record.

Types of certificates in AWS Certificate Manager

AWS Certificate Manager (ACM) provides three types of certificates: public, private, and imported. ACM can also issue Amazon N2 certificates from the Amazon N2 CA hierarchy, for authentication outside web browsers, including mutual TLS (mTLS).

Public certificates

ACM issues public certificates from Amazon Trust Services certificate authorities (CAs) in the web public key infrastructure (WebPKI). Major browsers and operating systems include the Amazon Trust Services WebPKI roots in their trust stores.

Private certificates

ACM issues private certificates from a CA in your own AWS Private CA hierarchy. Browsers and operating systems do not include your private roots in their trust stores.

Imported certificates

You obtain certificates from a CA outside ACM, such as a third-party CA or your own AWS Private CA CA, and import them into ACM.

Amazon N2 certificates

ACM issues Amazon N2 certificates from the Amazon N2 CAs, for authentication outside web browsers, such as mutual TLS (mTLS). They chain to the Amazon N2 roots or to the Starfield Services Root Certificate Authority - G2 (SFG2) root certificate. Amazon N2 certificates have the same `Type` as public certificates (`AMAZON_ISSUED`). You request them with the `RootHierarchyType` parameter. For more information, see Amazon N2 certificates.

Choose a certificate type

Certificate	Use it when	Who operates the CA	To get started
Public	Browsers or other clients that you do not control must accept the certifica te without configura	Amazon	Request a public certificate, or use ACME certificate automation

Certificate	Use it when	Who operates the CA	To get started
	tion. This includes certificates on AWS integrated services, such as Elastic Load Balancing or CloudFront, that serve browser clients.		
Amazon N2	You need certificates for authentication outside web browsers, such as mTLS between services or between organizations. They can authenticate a client, a server, or both. ACM validates your domains, and with the Amazon N2 roots a certificate can be valid for up to 395 days.	Amazon	Request an Amazon N2 certificate
Private	You want to operate your own CA hierarchy and control the certificate profile and validity period, for example for internal traffic or mTLS between your services.	You, in AWS Private CA	Request a private certificate

Certificate	Use it when	Who operates the CA	To get started
Imported	You already have a certificate from a CA outside AWS and want to use it with ACM integrated services.	The CA that issued it	Import a certificate

Compare certificate types

Characteristic	Public	Amazon N2	Private	Imported
What the client or server that verifies the certificate needs in its trust store	Nothing. Major browsers and operating systems include the Amazon Trust Services WebPKI roots.	The Amazon N2 roots or SFG2, depending on the root hierarchy. See Compare the Amazon N2 certificates root hierarchies.	The root certificate of your private CA	The root certificate of the CA that issued it
Domain validation	DNS, email, or HTTP	DNS only	None	None
Certificate transparency logging	Yes	No	No	Depends on the CA that issued it
Export from ACM	Only if you request the certificate with `Export` set to `ENABLED`	Yes, unless you request the certificate with `Export` set to `DISABLED`	Yes	No. You already hold the private key.

How ACM identifies each certificate

`DescribeCertificate` returns `Type` for every certificate, and `RootHierarchyType` and `CertificateUsage` only for Amazon N2 certificates. The following table shows how you get each certificate and the values that ACM returns.

Certificate	How you get it	Type	RootHierarchyType	CertificateUsage
Public	`RequestCertificate` without `RootHierarchyType`	`AMAZON_ISSUED`	Not returned	Not returned
Amazon N2	`RequestCertificate` with `RootHierarchyType` set to `N2_AMAZON_MANAGED` or `N2_SFG2_AMAZON_MANAGED`	`AMAZON_ISSUED`	The value that you specified	`CLIENT_AND_SERVER_AUTH` or `CLIENT_AUTH_ONLY`
Private	`RequestCertificate` with `CertificateAuthorityArn`	`PRIVATE`	Not returned	Not returned
Imported	`ImportCertificate`	`IMPORTED`	Not returned	Not returned

In a request, `RootHierarchyType` accepts only `N2_AMAZON_MANAGED` and `N2_SFG2_AMAZON_MANAGED`. Any other value fails with a `ValidationException`. To request a public certificate, omit `RootHierarchyType`. To request a private certificate, specify

`CertificateAuthorityArn` instead. You cannot specify both `RootHierarchyType` and `CertificateAuthorityArn`.

`CertificateUsage` accepts `CLIENT_AND_SERVER_AUTH` and `CLIENT_AUTH_ONLY`, and only with an Amazon N2 `RootHierarchyType`. For the supported combinations, see How Amazon N2 certificates work in ACM.

By default, `ListCertificates` does not return Amazon N2 certificates. To list them, specify `N2_AMAZON_MANAGED`, `N2_SFG2_AMAZON_MANAGED`, or both in the `RootHierarchyTypes` filter. With the filter, `ListCertificates` returns only certificates from the hierarchies that you specify. For an example command, see Listing and finding.

AWS Certificate Manager public certificates

After you request a public certificate you must validate domain ownership, as described in Validate domain ownership for AWS Certificate Manager public certificates.

Public ACM certificates follow the X.509 standard and are subject to the following restrictions:

- **Names:** You must use DNS-compliant subject names. For more information, see Domain Names.
- **Algorithm:** For encryption, the certificate private key algorithm must be either 2048-bit RSA, 256-bit ECDSA, or 384-bit ECDSA.
- **Expiration:** Each certificate is valid for 198 days.
- **Renewal:** ACM attempts to renew a public certificate automatically 45 days before expiration.

> **Note**
>
> To automate public certificate issuance and renewal outside integrated services such as on Amazon EC2 instances, use the ACME protocol. For more information, see ACME certificate automation. Alternatively, if you cannot use ACME, you can automate exportable certificates issued from ACM through AWS Workload Credentials Provider.

Administrators can use ACM Conditional Key Policies to control how end users issue new certificates. These Conditional keys allow restrictions to be placed on domains, validation methods, and other attributes related to a certificate request. If you encounter problems when requesting a certificate, see Troubleshoot certificate requests.

To request a certificate for a private PKI using AWS Private CA, see Request a private certificate in AWS Certificate Manager.

Topics

- AWS Certificate Manager public certificate characteristics and limitations
- Request a public certificate in AWS Certificate Manager
- AWS Certificate Manager exportable public certificates
- Validate domain ownership for AWS Certificate Manager public certificates

AWS Certificate Manager public certificate characteristics and limitations

Public certificates provided by ACM have the following characteristics and limitations. These apply only to certificates provided by ACM. They might not apply to imported certificates. For how these characteristics differ for Amazon N2 certificates, see Compare certificate types.

Browser and application trust

ACM public certificates are trusted by all major browsers including Google Chrome, Microsoft Edge, Mozilla Firefox, and Apple Safari. Browsers display a lock icon when connected by TLS to sites using ACM public certificates. Java also trusts ACM public certificates. For more information, see the repository on the Amazon Trust Services website. Amazon N2 certificates are not intended for browser clients. For more information, see Amazon N2 roots and trust.

Certificate authority and hierarchy

Public certificates requested through ACM come from Amazon Trust Services, an Amazon-managed public certificate authority (CA). Amazon Root CAs 1 to 4 are cross-signed by Starfield G2 Root Certificate Authority – G2. Starfield root is trusted on Android (later Gingerbread versions) and iOS (version 4.1+). Amazon roots are trusted by iOS 11+. Browsers, applications, or OSes including Amazon or Starfield roots will trust ACM public certificates.

ACM issues leaf or end-entity certificates to customers through intermediate CAs, randomly assigned based on the certificate type (RSA or ECDSA). ACM doesn't provide intermediate CA information due to this random selection.

Domain Validation (DV)

ACM certificates are domain validated, identifying only a domain name. When requesting an ACM certificate, you must prove ownership or control of all specified domains. You can validate ownership using DNS. For more information, see AWS Certificate Manager DNS validation.

HTTP validation

ACM supports HTTP validation for domain ownership verification when issuing public TLS certificates for use with CloudFront. This method uses HTTP redirects to prove domain ownership and offers automatic renewal similar to DNS validation. HTTP validation is currently only available through the CloudFront Distribution Tenants feature.

HTTP redirect

For HTTP validation, ACM provides a `RedirectFrom` URL and a `RedirectTo` URL. You must set up a redirect from `RedirectFrom` to `RedirectTo` to demonstrate domain control. The `RedirectFrom` URL includes the validated domain, while `RedirectTo` points to an ACM-controlled location in the CloudFront infrastructure containing a unique validation token.

Managed by

Certificates in ACM managed by another service show that service's identity in the `ManagedBy` field. For certificates using HTTP validation with CloudFront, this field displays "CLOUDFRONT". These certificates can only be used through CloudFront. The `ManagedBy` field appears in the **DescribeCertificate** and **ListCertificates** APIs, and on the certificates inventory and details pages in the ACM console.

The `ManagedBy` field is mutually exclusive with the "Can be used with" attribute. For CloudFront-managed certificates, you can't add new usages through other AWS services. You can only use these certificates with more resources through the CloudFront API.

Intermediate and root CA rotation

Amazon may discontinue an intermediate CA without notice to maintain a resilient certificate infrastructure. These changes don't impact customers. For more information, see "Amazon introduces dynamic intermediate certificate authorities".

If Amazon discontinues a root CA, the change will occur as quickly as needed. Amazon will use all available methods to notify AWS customers, including the Health Dashboard, email, and outreach to technical account managers.

Firewall access for revocation

Revoked end-entity certificates use OCSP and CRLs to verify and publish revocation information. Some customer firewalls may need additional rules to allow these mechanisms.

Use these URL wildcard patterns to identify revocation traffic:

- **OCSP**

  `http://ocsp.?????.amazontrust.com`

  `http://ocsp.*.amazontrust.com`

- **CRL**

  `http://crl.?????.amazontrust.com/?????.crl`

  `http://crl.*.amazontrust.com/*.crl`

An asterisk (*) represents one or more alphanumeric characters, a question mark (?) represents a single alphanumeric character, and a hash mark (#) represents a numeral.

> **Note**
>
> For revocation information for ACME certificates, see Revocation.

Key algorithms

Certificates must specify an algorithm and key size. ACM supports these RSA and ECDSA public key algorithms:

- RSA 1024 bit (`RSA_1024`)
- RSA 2048 bit (`RSA_2048`)*
- RSA 3072 bit (`RSA_3072`)
- RSA 4096 bit (`RSA_4096`)
- ECDSA 256 bit (`EC_prime256v1`)*
- ECDSA 384 bit (`EC_secp384r1`)*
- ECDSA 521 bit (`EC_secp521r1`)

ACM can request new certificates using algorithms marked with an asterisk (*). Other algorithms are for imported certificates only.

> **Note**
>
> For private PKI certificates signed by a AWS Private CA CA, the signing algorithm family (RSA or ECDSA) must match the CA's secret key algorithm family.

ECDSA keys are smaller and more computationally efficient than RSA keys of comparable security, but not all network clients support ECDSA. This table, adapted from NIST, compares RSA and ECDSA key sizes (in bits) for equivalent security strengths:

Comparing security for algorithms and keys

Security strength	RSA key size	ECDSA key size
128	3072	256
192	7680	384
256	15360	521

Security strength, as a power of 2, relates to the number of guesses needed to break the encryption. For example, both a 3072-bit RSA key and a 256-bit ECDSA key can be retrieved with no more than 2^{128} guesses.

For help choosing an algorithm, see the AWS blog post How to evaluate and use ECDSA certificates in AWS Certificate Manager.

> **Important**
>
> Integrated services allow only supported algorithms and key sizes for their resources. Support varies based on whether the certificate is imported into IAM or ACM. For details, see each service's documentation:
>
> - For Elastic Load Balancing, see HTTPS Listeners for Your Application Load Balancer.
> - For CloudFront, see Supported SSL/TLS Protocols and Ciphers.

Managed Renewal and Deployment

ACM manages the renewal and provisioning of ACM certificates. Automatic renewal helps prevent downtime from misconfigured, revoked, or expired certificates. For more information, see Managed certificate renewal in AWS Certificate Manager.

Multiple Domain Names

Each ACM certificate must include at least one fully qualified domain name (FQDN) and can include additional names. For example, a certificate for `www.example.com` can also include `www.example.net`. This applies to bare domains (zone apex or naked domains) too. You can request a certificate for www.example.com and include example.com. For more information, see AWS Certificate Manager public certificates.

Punycode

The following Punycode requirements for Internationalized Domain Names must be met:

1. Domain names beginning with the pattern "<character><character>--" must match "xn--".
2. Domain names beginning with "xn--" must also be valid Internationalized Domain Names.

Punycode examples

Domain Name	Fulfills #1	Fulfills #2	Allowe	Note
example.com	n/a	n/a	✓	Does not start with "<charact er><character>--"
a--exampl e.com	n/a	n/a	✓	Does not start with "<charact er><character>--"
abc--exam ple.com	n/a	n/a	✓	Does not start with "<charact er><character>--"
xn--xyz.com	Yes	Yes	✓	Valid Internationalized Domain Name (resolves to 简.com)
xn--examp le.com	Yes	No	✗	Not a valid Internationalized Domain Name

Domain Name	Fulfills #1	Fulfills #2	Allowe	Note
ab--example.com	No	No	✗	Must start with "xn--"

Validity Period

ACM public certificates are valid for 198 days. The validity of an Amazon N2 certificate depends on its root hierarchy. For more information, see Validity period.

Wildcard Names

ACM allows an asterisk (*) in the domain name to create a wildcard certificate protecting multiple sites in the same domain. For example, `*.example.com` protects `www.example.com` and `images.example.com`.

In a wildcard certificate, the asterisk (*) must be leftmost in the domain name and protects only one subdomain level. For instance, `*.example.com` protects `login.example.com` and `test.example.com`, but not `test.login.example.com`. Also, `*.example.com` protects *only* subdomains, not the bare or apex domain (`example.com`). You can request a certificate for both a bare domain and its subdomains by specifying multiple domain names, such as `example.com` and `*.example.com`.

> **Important**
>
> If you use CloudFront, note that HTTP validation does not support wildcard certificates. For wildcard certificates, you must use either DNS validation or email validation. We recommend DNS validation because it supports automatic certificate renewal.

Request a public certificate in AWS Certificate Manager

> **Important**
>
> Throughout 2027, AWS Certificate Manager will deprecate email validation for new public certificates and renewals. Use DNS validation instead. For more information, see AWS

Certificate Manager will discontinue email validation to prove domain validation for certificates on the AWS Security Blog.

You can request AWS Certificate Manager public certificates from the ACM console, AWS CLI, or API. You can use these certificates with integrated AWS services or export them for use outside of AWS Cloud.

The following list describes the differences between public certificates and exportable public certificates.

Public certificates

Use ACM public certificates with integrated AWS services like Elastic Load Balancing, Amazon CloudFront, and Amazon API Gateway. For more information, see Managed automation with integrated services.

> **Note**
>
> ACM public certificates created prior to June 17, 2025 cannot be exported.

Exportable public certificates

Exportable public certificates work with integrated AWS services and can also be used outside AWS Cloud. For more information, see AWS Certificate Manager exportable public certificates and Managed automation with integrated services. You must create a new ACM public certificate and enable exportable to be able to export the public certificate.

> **Tip**
>
> If you need public certificates for customer-managed infrastructure (such as on-premises servers or Kubernetes clusters) and want to use industry-standard ACME clients, see ACME certificate automation.

The following sections discuss how to request, export, and revoke a public ACM certificate.

Topics

- Request a public certificate using the console
- Request a public certificate using the CLI

Request a public certificate using the console

To request an ACM public certificate (console)

1. Sign in to the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/home.

   Choose **Request a certificate**.

2. In the **Domain names** section, type your domain name.

   You can use a fully qualified domain name (FQDN), such as **`www.example.com`**, or a bare or apex domain name such as **`example.com`**. You can also use an asterisk (**`*`**) as a wild card in the leftmost position to protect several site names in the same domain. For example, **`*.example.com`** protects **`corp.example.com`**, and **`images.example.com`**. The wildcard name will appear in the **Subject** field and in the **Subject Alternative Name** extension of the ACM certificate.

   When you request a wildcard certificate, the asterisk (**`*`**) must be in the leftmost position of the domain name and can protect only one subdomain level. For example, **`*.example.com`** can protect **`login.example.com`**, and **`test.example.com`**, but it cannot protect **`test.login.example.com`**. Also note that **`*.example.com`** protects *only* the subdomains of **`example.com`**, it does not protect the bare or apex domain (**`example.com`**). To protect both, see the next step.

   > **Note**
   >
   > In compliance with RFC 5280, the length of the domain name (technically, the Common Name) that you enter in this step cannot exceed 64 octets (characters), including periods. Each subsequent Subject Alternative Name (SAN) that you provide, as in the next step, can be up to 253 octets in length.

- To add another name, choose **Add another name to this certificate** and type the name in the text box. This is useful for protecting both a bare or apex domain (such as **`example.com`**) and its subdomains such as **`*.example.com`**).

3. If you want to create an ACM exportable public certificates, select the **Enable export** option. You'll be able to access the certificate's private keys and use it outside AWS Cloud. For more information, see AWS Certificate Manager exportable public certificates.
4. In the **Validation method** section, choose either **DNS validation – recommended** or **Email validation**, depending on your needs.

   Before ACM issues a certificate, it validates that you own or control the domain names in your certificate request.

   Use DNS validation for new certificates. With DNS validation, you can renew certificates automatically and avoid email delivery issues. The following substeps describe both DNS validation and email validation for reference while email validation remains available.

   a. If you choose email validation, ACM sends validation email to the domain that you specify in the domain name field. If you specify a validation domain, ACM sends the email to that validation domain instead. For more information about email validation, see AWS Certificate Manager email validation.
   b. If you use DNS validation, you simply add a CNAME record provided by ACM to your DNS configuration. For more information about DNS validation, see AWS Certificate Manager DNS validation.
5. In the **Key algorithm** section, choose an algorithm.
6. In the **Tags** page, you can optionally tag your certificate. Tags are key-value pairs that serve as metadata for identifying and organizing AWS resources. For a list of ACM tag parameters and for instructions on how to add tags to certificates after creation, see Tag AWS Certificate Manager resources.

   When you finish adding tags, choose **Request**.
7. After the request is processed, the console returns you to your certificate list, where information about the new certificate is displayed.

   A certificate enters status **Pending validation** upon being requested, unless it fails for any of the reasons given in the troubleshooting topic Certificate request fails. ACM makes repeated attempts to validate a certificate for 72 hours and then times out. If a certificate shows status

Failed or **Validation timed out**, delete the request, correct the issue with DNS validation or Email validation, and try again. If validation succeeds, the certificate enters status **Issued**.

> **Note**
>
> Depending on how you have ordered the list, a certificate you are looking for might not be immediately visible. You can click the black triangle at right to change the ordering. You can also navigate through multiple pages of certificates using the page numbers at upper-right.

Request a public certificate using the CLI

Use the request-certificate command to request a new public ACM certificate on the command line. Optional values for the validation method are DNS and EMAIL. Optional values for the key algorithm are RSA_2048 (the default if the parameter is not explicitly provided), EC_prime256v1, and EC_secp384r1.

```
aws acm request-certificate \
--domain-name www.example.com \
--key-algorithm EC_Prime256v1 \
--validation-method DNS \
--idempotency-token 1234 \
--options Export=ENABLED
```

> **Note**
>
> The `CertificateTransparencyLoggingPreference` option is deprecated. All ACM public certificates are automatically recorded in certificate transparency logs. Amazon N2 certificates are not recorded, and the option has no effect on them.

This command outputs the Amazon Resource Name (ARN) of your new public certificate.

```
{
    "CertificateArn": "arn:aws:acm:Region:444455556666:certificate/certificate_ID"
}
```

AWS Certificate Manager exportable public certificates

AWS Certificate Manager exportable public certificates lets you provision, manage, and deploy SSL/TLS certificates anywhere - including Amazon EC2 instances, containers, and on-premises hosts. This feature extends ACM issued public certificates beyond integrated AWS services, giving you centralized control over certificates across your entire infrastructure.

Benefits

The following outlines benefits of ACM exportable public certificates:

- *Simplified Certificate Management*: Centrally manage certificates for all your resources with ACM.
- *Faster Certificate Issuance*: Access and use certificates in less time.
- *Automated Renewals*: ACM automatically handles certificate renewals and notifies you when new certificates are ready for deployment. For more information, see Amazon EventBridge support for ACM.
- *Cost Effective*: Pay only for the exportable public certificates you create.
- *Flexible Deployment*: Use certificates with any server or application that supports standard SSL/TLS certificates.

How ACM exportable public certificates works

The following outlines how ACM exportable public certificates work:

1. Request an exportable certificate through ACM for your domain.
2. Validate domain ownership using DNS validation.
3. Export the certificate, private key, and certificate chain.
4. Deploy the certificate to your server or application.
5. ACM manages renewals and sends notifications when new certificates are available.

Security considerations

The following are security considerations when using ACM exportable public certificates. For more information, see Data protection in AWS Certificate Manager.

- Protect exported private keys using secure storage and access controls.

- Use ACM's revocation feature if you suspect key compromise.
- Implement proper key rotation procedures when deploying renewed certificates.

Limitations

The following are some ACM certificate limitations:

- Certificates have a 198 days validity period.
- ACM renews certificates set to expire 45 days before their expiration date.
- You must manage the deployment process for exported certificates.

Pricing

You are subject to an additional charge for exportable public SSL/TLS certificates that you create with AWS Certificate Manager. For the latest ACM pricing information, see the AWS Certificate Manager Service Pricing page on the AWS website.

Best practices

The following are some best practices when using ACM certificates:

- Once a certificate is renewed, you should begin using it immediately.
- Test and implement automated deployment processes for renewed certificates.
- Monitor certificate deployments using Amazon EventBridge metrics and alarms.

Export an AWS Certificate Manager public certificate

The following procedures walks you through how you can export an ACM public certificate in the ACM console. Alternatively, you can use the `export-certificate` AWS CLI or ExportCertificate API action.

> **Tip**
>
> If you need public certificates for customer-managed infrastructure (such as on-premises servers or Kubernetes clusters) and want to use industry-standard ACME clients to automate issuance and renewal, see ACME certificate automation.

> **Note**
>
> ACM public certificates created prior to June 17, 2025 cannot be exported.

Export a public certificate (console)

1. Sign in to the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/.
2. Choose **List certificates** and select the checkbox of the certificate you want to export.
   - Alternatively, you can select the certificate. In the certificate detail page, select **Export**.
3. Choose **More actions** and then choose **Export**.
4. Enter and confirm a passphrase for the private key.
5. You can download or copy the certificate files.

> **Note**
>
> In the ACM console, you're able to export .pem certificate files. You can convert the .pem file to another file format, like .ppk. For more information, see this re:Post article.

Export a public certificate (AWS CLI)

Use the `export-certificate` AWS CLI command or ExportCertificate API action to export a public certificate and private key. You must assign a passphrase when you run the command. For added security, use a file editor to store your passphrase in a file, and then supply the passphrase by supplying the file. This prevents your passphrase from being stored in the command history and prevents others from seeing the passphrase as you type it in.

> **Note**
>
> The file containing the passphrase must not end in a line terminator. You can check your password file like this:

```
$ file -k passphrase.txt
```

```
passphrase.txt: ASCII text, with no line terminators
```

The following examples pipe the command output to `jq` to apply PEM formatting.

```
[Windows/Linux]$ aws acm export-certificate \
    --certificate-arn arn:aws:acm:us-east-1:111122223333:certificate/certificate_ID \
    --passphrase fileb://path-to-passphrase-file  \
    | jq -r '"\(.Certificate)\(.CertificateChain)\(.PrivateKey)"'
```

This outputs a base64-encoded, PEM-format certificate, also containing the certificate chain and encrypted private key, as in the following abbreviated example.

```
-----BEGIN CERTIFICATE-----
MIIDTDCCAjSgAwIBAgIRANWuFpqA16g3IwStE3vVpTwwDQYJKoZIhvcNAQELBQAw
EzERMA8GA1UECgwIdHJvbG9sb2wwHhcNMTkwNzE5MTYxNTU1WhcNMjAwODE5MTcx
NTU1WjAXMRUwEwYDVQQDDAx3d3cuc3B1ZHMuaW8wggEiMA0GCSqGSIb3DQEBAQUA
...
8UNFQvNoo1VtICL4cwWOdLOkxpwkkKWtcEkQuHE1v5Vn6HpbfFmxkdPEasoDhthH
FFWIf4/+VOlbDLgjU4HgtmV4IJDtqM9rGOZ42eFYmmc3eQO0GmigBBwwXp3j6hoi
74YM+igvtILnbYkPYhY9qz8h7lHUmannS8j6YxmtpPY=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIC8zCCAdugAwIBAgIRAM/jQ/6h2/MI1NYWX3dDaZswDQYJKoZIhvcNAQELBQAw
EzERMA8GA1UECgwIdHJvbG9sb2wwHhcNMTkwNjE5MTk0NTE2WhcNMjkwNjE5MjA0
NTE2WjATMREwDwYDVQQKDAh0cm9sb2xvbDCCASIwDQYJKoZIhvcNAQEBBQADggEP
...
j2PAOviqIXjwr08Zo/rTy/8m6LAsmm3LVVYKLyPdl+KB6M/+H93Z1/Bs8ERqqga/
6lfM6iw2JHtkW+q4WexvQSoqRXFhCZWbWPZTUpBS0d4/Y5q92S3iJLRa/JQ0d4U1
tWZyqJ2rj2RL+h7CE71XIAM//oHGcDDPaQBFD2DTisB/+ppGeDuB
-----END CERTIFICATE-----
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFKzBVBgkqhkiG9w0BBQ0wSDAnBgkqhkiG9w0BBQwwGgQUMrZb7kZJ8nTZg7aB
1zmaQh4vwloCAggAMB0GCWCGSAFlAwQBKgQQDViroIHStQgNOjR6nTUnuwSCBNAN
JM4SG202YPUiddWeWmX/RKGg3lIdE+A0WLTPskNCdCAHqdhOSqBwt65qUTZe3gBt
...
ZGipF/DobHDMkpwiaRR5sz6nG4wcki0ryYjAQrdGsR6EVvUUXADkrnrrxuHTWjFl
wEuqyd8X/ApkQsYFX/nhepOEIGWf8Xu0nrjQo77/evhG0sHXborGzgCJwKuimPVy
Fs5kw5mvEoe5DAe3rSKsSUJ1tM4RagJj2WH+BC04SZWNH8kxfOC1E/GSLBCixv3v
+Lwq38CEJRQJLdpta8NcLKnFBwmmVs9OV/VXzNuHYg==
-----END ENCRYPTED PRIVATE KEY-----
```

To output everything to a file, append the > redirect to the previous example, yielding the following command:

```
$ aws acm export-certificate \
     --certificate-arn arn:aws:acm:us-east-1:111122223333:certificate/certificate_ID \
     --passphrase fileb://path-to-passphrase-file \
     | jq -r '"\(.Certificate)\(.CertificateChain)\(.PrivateKey)"' \
     > /tmp/export.txt
```

Secure Kubernetes Workloads with ACM Certificates

You can use AWS Certificate Manager exportable public certificates with AWS Controllers for Kubernetes (ACK) to issue and export public TLS certificates from ACM to your Kubernetes workloads. This integration enables you to secure Amazon Elastic Kubernetes Service (Amazon EKS) pods and terminate TLS at your Kubernetes Ingress. To get started, see the ACM Controller for Kubernetes on GitHub.

Tip

If you use cert-manager or another industry-standard ACME client in your Kubernetes cluster, you can automate certificate issuance from ACM through the ACME protocol instead. In this approach, the ACME client generates and holds the private key. For more information, see ACME certificate automation.

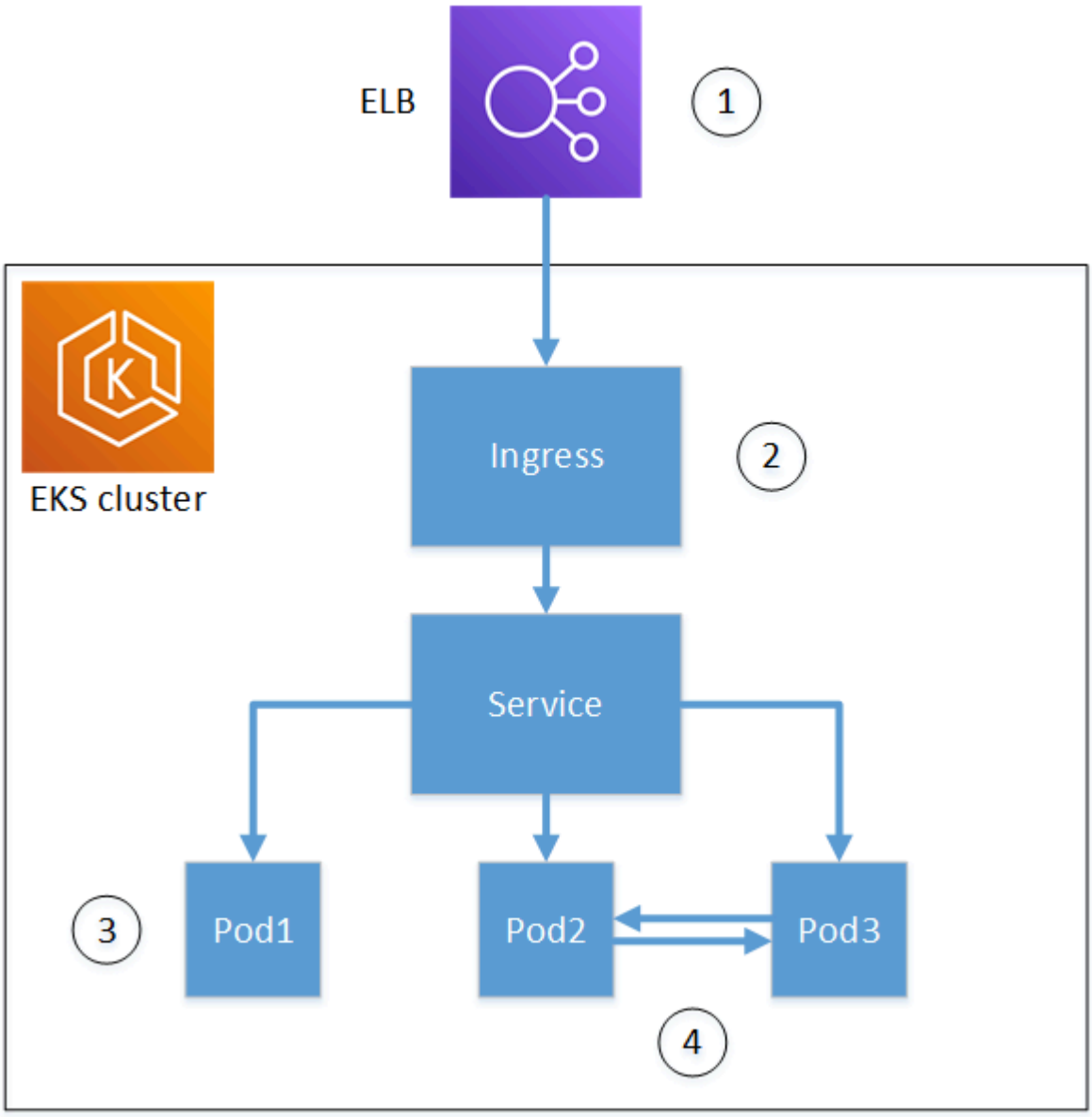
AWS Controllers for Kubernetes (ACK) extends the Kubernetes API to manage AWS resources using native Kubernetes manifests. The ACK service controller for ACM provides automated certificate lifecycle management within your Kubernetes workflow. When you create an ACM Certificate resource in Kubernetes, the ACK controller performs the following actions:

1. Requests a certificate from ACM, which generates the certificate signing request (CSR).
2. Waits for domain validation to complete and for ACM to issue the certificate.
3. If the `exportTo` field is specified, exports the issued certificate and private key and stores them in your specified Kubernetes Secret.
4. If the `exportTo` field is specified and the certificate is eligible for renewal, updates the Kubernetes Secret with renewed certificates before expiration.

Publicly issued certificates require domain validation before ACM can issue them. You can use the ACK service controller for Amazon Route 53 to automatically create the required DNS validation CNAME records in your hosted zone.

Certificate usage options

You can use ACM certificates with Kubernetes in a few ways:



1. *Load balancer termination (without export)*: Issue certificates through ACK and use them to terminate TLS at an AWS load balancer. The certificate remains in ACM and is automatically discovered by the AWS Load Balancer Controller. This approach does not require exporting the certificate.

2. *Ingress termination (with export)*: Export certificates from ACM and store them in Kubernetes Secrets for TLS termination at the Ingress level. This enables you to use certificates directly within your Kubernetes workloads.

> **Note**
>
> For use cases that require private certificates, see AWS Private CA Connector for Kubernetes, a cert-manager plugin.

Prerequisites

Before you install the ACK service controller for ACM, ensure you have the following:

- A Kubernetes cluster.
- Helm installed.
- `kubectl` configured to communicate with your cluster.
- `eksctl` installed for configuring pod identity associations on EKS.

Install the ACK service controller for ACM

Use Helm to install the ACK service controller for ACM in your Amazon EKS cluster.

1. Create a namespace for the ACK controller.

```
$ kubectl create namespace ack-system --dry-run=client -o yaml | kubectl apply -f -
```

2. Create a pod identity association for the ACK controller. Replace *CLUSTER_NAME* with your cluster name and *REGION* with your AWS Region.

```
$ eksctl create podidentityassociation --cluster CLUSTER_NAME --region REGION \
    --namespace ack-system \
    --create-service-account \
    --service-account-name ack-acm-controller \
    --permission-policy-arns arn:aws:iam::aws:policy/
AWSCertificateManagerFullAccess
```

3. Log in to the Amazon ECR Public registry.

```
$ aws ecr-public get-login-password --region us-east-1 | helm registry login --
username AWS --password-stdin public.ecr.aws
```

4. Install the ACK service controller for ACM. Replace *REGION* with your AWS Region.

```
$ helm install -n ack-system ack-acm-controller oci://public.ecr.aws/
aws-controllers-k8s/acm-chart --set serviceAccount.create=false --set
 serviceAccount.name=ack-acm-controller --set aws.region=REGION
```

5. Verify the controller is running.

```
$ kubectl get pods -n ack-system
```

For more information about pod identity associations, see EKS Pod Identity in the *Amazon EKS User Guide*.

Example: Terminate TLS at the Ingress

The following example demonstrates how to export an ACM certificate and use it to terminate TLS at the Kubernetes Ingress level. This configuration creates an ACM certificate, exports it to a Kubernetes Secret, and configures an Ingress resource to use the certificate for TLS termination.

In this example:

- Secret is created to store the exported certificate (`exported-cert-secret`)
- The ACK Certificate resource requests a certificate from ACM for your domain and exports it to the `exported-cert-secret` Secret.
- The Ingress resource references the `exported-cert-secret` to terminate TLS for incoming traffic.

Replace `${HOSTNAME}` with your domain name.

```
apiVersion: v1
kind: Secret
type: kubernetes.io/tls
metadata:
  name: exported-cert-secret
  namespace: demo-app
```

```
data:
  tls.crt: ""
  tls.key: ""
---
apiVersion: acm.services.k8s.aws/v1alpha1
kind: Certificate
metadata:
  name: exportable-public-cert
  namespace: demo-app
spec:
  domainName: ${HOSTNAME}
  options:
    certificateTransparencyLoggingPreference: ENABLED
  exportTo:
    namespace: demo-app
    name: exported-cert-secret
    key: tls.crt
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-traefik
  namespace: demo-app
spec:
  tls:
  - hosts:
    - ${HOSTNAME}
    secretName: exported-cert-secret
  ingressClassName: traefik
  rules:
  - host: ${HOSTNAME}
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: whoami
            port:
              number: 80
```

Once deployed, the ACK service controller for ACM automatically manages the certificate lifecycle, including renewals. When ACM renews the certificate, the controller updates the `exported-cert-`

`secret` Secret with the new certificate, ensuring your Ingress continues to use valid certificates without manual intervention.

Revoke an AWS Certificate Manager public certificate

You can revoke an AWS Certificate Manager exportable public certificates using the ACM console, AWS CLI, or API action.

> **Warning**
>
> After a certificate is revoked, you cannot reuse the certificate. Revoking a certificate is permanent.

You may need to revoke a certificate to comply with your organization's policies or mitigate key compromise. A reason is required when revoking a certificate. The following reasons can be used:

- Unspecified
- Affiliation changed
- Superseded
- Cessation of operation

To learn more see, Amazon Trust Services Certificate Subscriber Agreement and Amazon Trust Service.

AWS provides two services to check certificate revocations: Online Certificate Status Protocol (OCSP) and certificate revocation list. With OCSP, the client queries an authoritative revocation database that returns a status in real-time. OCSP depends on validation information embedded in certificates.

Considerations

The following are considerations before revoking a certificate:

- You can only revoke certificates that were previously exported.
- You cannot revoke non-exportable public certificates. If you no longer need these certificate, you should delete them instead.
- If you no longer need the certificate, you should delete certificates instead of revoking certificates.

- The certificate revocation process is global. All valid certificates you choose to revoke will be revoked along with their associated ARNs.
- Certificate revocation is permanent. You can't retrieve revoked certificates to reuse.
- It can take up to 24 hours for certificate revocation to take effect.

Revoke a certificate (console)

The following procedure walks you through how you can revoke an ACM public or private certificate.

1. Sign in to the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/.
2. Choose **List certificates** and select the checkbox of the certificate you want to revoke.
   - Alternatively, you can select the certificate. In the certificate detail page, select **Revoke**.
3. Choose **More actions** and then choose **Revoke**.
4. A dialog box appears where you must provide a revoke reason, enter **`revoke`**, and then choose **Revoke**.

Revoke a certificate (AWS CLI)

Use the `revoke-certificate` AWS CLI command or `RevokeCertificate` API action to revoke an ACM public or private certificate. You can retrieve the certificate's ARN by calling the `list-certificates` command. For an Amazon N2 certificate, include the `--root-hierarchy-types` option, because `list-certificates` does not return Amazon N2 certificates by default.

```
$ aws acm revoke-certificate \
    --certificate-arn arn:aws:acm:us-
east-1:111122223333:certificate/12345678-1234-1234-1234 \
    --revocation-reason "UNSPECIFIED"
```

> **Warning**
>
> After a certificate is revoked, you cannot reuse the certificate. Revoking a certificate is permanent.

The following would be the output for the `revoke-certificate` command.

```
arn:aws:acm:us-east-1:111122223333:certificate/12345678-1234-1234-1234
```

Configure automatic renewal events

With AWS Certificate Manager exportable public certificates and Amazon EventBridge, you can configure automatic certificate renewals events.

1. Set up an Amazon EventBridge event to monitor certificate renewals. For more information, see Amazon EventBridge support for ACM.
2. Create automation to handle certificate deployment when renewals occur. For more information, see Initiating actions with Amazon EventBridge in ACM.
3. Configure EventBridge events to alert you of any renewal or deployment failures.

Force certificate renewal

You can renew your ACM public and private certificates with the ACM console, renew-certificate AWS CLI, or `RenewCertificate` API action. You can only renew certificates that have been previously exported.

> **Important**
>
> When you renew an ACM exportable public certificates, you're charged an additional fee. For the latest ACM pricing information, see the AWS Certificate Manager Service Pricing page on the AWS website.

Renew a certificate (console)

The following procedure walks you through how you can force the renewal of an ACM public or private certificate.

1. Sign in to the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/.
2. Choose **List certificates** and select the checkbox of the certificate you want to renew.
   - Alternatively, you can select the certificate. In the certificate detail page, select **Renew**.

3. Choose **More actions** and then choose **Renew**.
4. A dialog box appears where you must enter **`renew`**and then choose **Renew**.

Renew a certificate (AWS CLI)

Use the `renew-certificate` AWS CLI command or `RenewCertificate` API action to renew an ACM public or private certificate. You can retrieve the certificate's ARN by calling the `list-certificates` command. The `renew-certificate` command does not return a response. For an Amazon N2 certificate, include the `--root-hierarchy-types` option, because `list-certificates` does not return Amazon N2 certificates by default.

```
$ aws acm renew-certificate \
    --certificate-arn arn:aws:acm:us-
east-1:111122223333:certificate/12345678-1234-1234-1234-123456789012
```

Validate domain ownership for AWS Certificate Manager public certificates

Before AWS Certificate Manager (ACM) issues a certificate from an Amazon certificate authority (CA), it must prove that you own or control all of the domain names that you specify in your request. This applies to every certificate whose `Type` is `AMAZON_ISSUED`: public certificates and Amazon N2 certificates. You can choose to prove your ownership with Domain Name System (DNS) validation, email validation, or HTTP validation when you request a certificate. Amazon N2 certificates support DNS validation only. For more information, see Domain validation.

> **Note**
>
> Validation applies only to certificates that ACM issues from Amazon CAs, whose `Type` is `AMAZON_ISSUED`: public certificates and Amazon N2 certificates. ACM does not validate domain ownership for imported certificates or for certificates signed by a private CA. ACM cannot validate resources in an Amazon VPC private hosted zone or any other private domain. For more information, see Troubleshoot certificate validation.

> **Note**
>
> If you use ACME certificate automation, domain validation is handled differently. ACME domain validations are persistent resources configured by PKI administrators that pre-authorize domains for certificate issuance. ACME clients do not perform live validation challenges. For more information, see ACME domain validation.

We recommend using DNS validation over email validation for the following reasons:

- AWS Certificate Manager will deprecate email validation throughout 2027 for new public certificates and renewals. For more information, see AWS Certificate Manager will discontinue email validation to prove domain validation for certificates on the AWS Security Blog.
- If you use Amazon Route 53 to manage your public DNS records, you can update your records through ACM directly.
- ACM automatically renews DNS-validated certificates for as long as a certificate remains in use and the DNS record is in place.
- Email-validated certificates require an action by the domain owner to be renewed. ACM begins sending renewal notices 45 days before expiration. These notices go to one or more of the domain's five common administrator addresses. The notifications contain a link that the domain owner can click for easy renewal. Once all listed domains are validated, ACM issues a renewed certificate with the same ARN.

HTTP validation is available for certificates used with CloudFront. This method uses HTTP redirects to prove domain ownership and offers automatic renewal similar to DNS validation.

> **Note**
>
> To switch an existing certificate from email validation to DNS validation, use the `UpdateCertificateOptions` API.

Topics

- AWS Certificate Manager DNS validation
- AWS Certificate Manager email validation
- AWS Certificate Manager HTTP validation

- Migrating from email to DNS validation

AWS Certificate Manager DNS validation

The Domain Name System (DNS) is a directory service for resources that are connected to a network. Your DNS provider maintains a database containing records that define your domain. When you choose DNS validation, ACM provides you with one or more CNAME records that must be added to this database. These records contain a unique key-value pair that serves as proof that you control the domain.

> **Note**
>
> To switch an existing certificate from email validation to DNS validation, use the `UpdateCertificateOptions` API.

For example, if you request a certificate for the `example.com` domain with `www.example.com` as an additional name, ACM creates two CNAME records for you. Each record, created specifically for your domain and your account, contains a name and a value. The value is an alias that points to an AWS domain that ACM uses to automatically renew your certificate. The CNAME records must be added to your DNS database only once. ACM automatically renews your certificate as long as the certificate is in use and your CNAME record remains in place.

> **Important**
>
> If you do not use Amazon Route 53 to manage your public DNS records, contact your DNS provider to find out how to add records. If you lack authority to edit your domain's DNS database, you must use email validation instead.

Without the need to repeat validation, you can request additional ACM certificates for your fully qualified domain name (FQDN) for as long as the CNAME record remains in place. That is, you can create replacement certificates that have the same domain name, or certificates that cover different subdomains. Since the CNAME validation token works for any AWS Region, you can re-create the same certificate in multiple Regions. You can also replace a deleted certificate.

You can stop automatic renewal either by removing the certificate from the AWS service with which it is associated or by deleting the CNAME record. If Route 53 is not your DNS provider,

contact your provider to find out how to delete a record. If Route 53 is your provider, see Deleting Resource Record Sets in the *Route 53 Developer Guide*. For more information about managed certificate renewal, see Managed certificate renewal in AWS Certificate Manager.

> **Note**
>
> CNAME resolution will fail if more than eight CNAMEs are chained together in your DNS configuration.

How CNAME records for ACM work

> **Note**
>
> This section is for customers who do not use Route 53 as their DNS provider.

If you are not using Route 53 as your DNS provider, you need to manually enter CNAME records provided by ACM into your provider's database, usually through a website. CNAME records are used for a number of purposes, including as redirect mechanisms and as containers for vendor-specific metadata. For ACM, these records allow initial domain ownership validation and ongoing automated certificate renewal.

The following table shows example CNAME records for six domain names. Each record's **Record Name**-**Record Value** pair serves to authenticate domain name ownership.

In the table, note that the first two **Record Name**-**Record Value** pairs are the same. This illustrates that for a wildcard domain, such as `*.example.com`, the strings created by ACM are the same as those created for its base domain, `example.com`. Otherwise, the paired **Record Name** and **Record Value** differ for each domain name.

Example CNAME records

Domain name	Record Name	Record Value	Comment
*.example.com	_*x1*.example.com.	_*x2*.acm-validations.aws.	Identical
example.com	_*x1*.example.com.	_*x2*.acm-validations.aws.	

Domain name	Record Name	Record Value	Comment
www.example.com	_*x3*.www.example.com.	_*x4*.acm-validations.aws.	Unique
host.example.com	_*x5*.host.example.com.	_*x6*.acm-validations.aws.	Unique
subdomain.example.com	_*x7*.subdomain.example.com.	_*x8*.acm-validations.aws.	Unique
host.subdomain.example.com	_*x9*.host.subdomain.example.com.	_*x10*.acm-validations.aws.	Unique

The *xN* values following the underscore (_) are long strings generated by ACM. For example,

```
_3639ac514e785e898d2646601fa951d5.example.com.
```

is representative of a resulting generated **Record Name**. The associated **Record Value** might be

```
_98d2646601fa951d53639ac514e785e8.acm-validation.aws.
```

for the same DNS record.

> **Note**
>
> If your DNS provider does not support CNAME values with a leading underscore, see Troubleshoot DNS Validation Problems.

When you request a certificate and specify DNS validation, ACM provides CNAME information in the following format:

Domain Name	Record Name	Record Type	Record Value
example.com	_a79865eb4cd1a6ab990a45779b4e0b96.example.com.	CNAME	_424c7224e9b0146f9a8808af955727d0.acm-validations.aws.

Domain Name is the FQDN associated with the certificate. *Record Name* identifies the record uniquely, serving as the key of the key-value pair. *Record Value* serves as the value of the key-value pair.

All three of these values (*Domain Name*, *Record Name*, and *Record Value*) must be entered into the appropriates fields of your DNS provider's web interface for adding DNS records. Providers are inconsistent in their handling of the record name (or just "name") field. In some cases, you are expected to provide the entire string as shown above. Other providers automatically append the domain name to whatever string you enter, meaning (in this example) that you should only enter

```
_a79865eb4cd1a6ab990a45779b4e0b96
```

into the name field. If you guess wrong about this, and enter a record name that contains a domain name (such as *`.example.com`*), you might end up with the following:

```
_a79865eb4cd1a6ab990a45779b4e0b96.example.com.example.com.
```

Validation will fail in this case. Consequently, you should try to determine in advance which type of input your provider expects.

Setting up DNS validation

This section describes how to configure a public certificate to use DNS validation.

To set up DNS validation in the console

> **Note**
>
> This procedure assumes that you have already created at least one certificate and that you are working in the AWS Region where you created it. If you try to open the console and see the first-use screen instead, or you succeed in opening the console and don't see your certificate in the list, confirm that you have specified the correct Region.

1. Open the ACM console at https://console.aws.amazon.com/acm/.
2. In the list of certificates, choose the **Certificate ID** of a certificate with status **Pending validation** that you want to configure. This opens a details page for the certificate.
3. In the **Domains** section, complete one of the following two procedures:

a. (Optional) Validate with Route 53.

An active **Create records in Route 53** button appears if the following conditions are true:

- You use Route 53 as your DNS provider.
- You have permission to write to the zone hosted by Route 53.
- Your FQDN has *not* already been validated.

> **Note**
>
> If you are in fact using Route 53 but the **Create records in Route 53** is missing or disabled, see ACM Console does not display "Create records in Route 53" button.

Choose the **Create records in Route 53**, then choose **Create records**. The **Certificate status** page should open with a status banner reporting **Successfully created DNS records**.

Your new certificate might continue to display a status of **Pending validation** for up to 30 minutes.

> **Tip**
>
> You cannot programmatically request that ACM automatically create your record in Route 53. You can, however, make an AWS CLI or API call to Route 53 to create the record in the Route 53 DNS database. For more information about Route 53 record sets, see Working with Resource Record Sets.

b. (Optional) If you are not using Route 53 as your DNS provider, you must retrieve the CNAME information and add it your DNS database. On the details page for the new certificate, you can do this in either of two ways:

- Copy the CNAME components displayed in the **Domains** section. This information needs to be added manually to your DNS database.
- Alternatively, choose **Export to CSV**. The information in the resulting file needs to be added manually to your DNS database.

> **Important**
>
> To avoid validation problems, review How CNAME records for ACM work before you add information to your DNS provider's database. If you do encounter problems, see Troubleshoot DNS validation problems.

If ACM is not able to validate the domain name within 72 hours from the time it generates a CNAME value for you, ACM changes the certificate status to **Validation timed out**. The most likely reason for this result is that you did not successfully update your DNS configuration with the value that ACM generated. To remedy this issue, you must request a new certificate after reviewing the CNAME instructions.

AWS Certificate Manager email validation

> **Important**
>
> Throughout 2027, AWS Certificate Manager will deprecate email validation for new public certificates and renewals. Use DNS validation instead. For more information, see AWS Certificate Manager will discontinue email validation to prove domain validation for certificates on the AWS Security Blog.

Before the Amazon certificate authority (CA) can issue a certificate for your site, AWS Certificate Manager (ACM) must verify that you own or control all of the domains that you specified in your request. This topic describes email validation for customers who are already using it. For new certificates, use DNS validation (see AWS Certificate Manager DNS validation).

If you encounter problems using email validation, see Troubleshoot email validation problems.

How email validation works

ACM sends validation email messages to the following five common system emails for each domain. Alternatively, you can specify a superdomain as a validation domain if you wish to receive these emails at that domain instead. Any subdomain up to the minimal website address is valid, and is used as the domain for the email address as the suffix after @. For example, you can

receive an email to admin@example.com if you specify example.com as the validation domain for subdomain.example.com.

- administrator@your_domain_name
- hostmaster@your_domain_name
- postmaster@your_domain_name
- webmaster@your_domain_name
- admin@your_domain_name

To prove that you own the domain, you must select the validation link included in these emails. ACM also sends validation emails to these same addresses to renew the certificate when the certificate is 45 days from expiry.

Email validation for multi-domain certificate requests using the ACM API or CLI results in an email message being sent by each requested domain, even if the request includes subdomains of other domains in the request. The domain owner needs to validate an email message for each of these domains before ACM can issue the certificate.

Exception to this process

If you request an ACM certificate for a domain name that begins with **www** or a wildcard asterisk (*****), ACM removes the leading **www** or asterisk and sends email to the administrative addresses. These addresses are formed by pre-pending admin@, administrator@, hostmaster@, postmaster@, and webmaster@ to the remaining portion of the domain name. For example, if you request an ACM certificate for www.example.com, email is sent to admin@example.com rather than to admin@www.example.com. Likewise, if you request an ACM certificate for *.test.example.com, email is sent to admin@test.example.com. The remaining common administrative addresses are similarly formed.

> **Important**
>
> ACM no longer supports WHOIS email validation for new certificates or renewals. Common system addresses remain supported. For details, see blog post.

Considerations

Observe the following considerations about email validation.

- You need a working email address registered in your domain in order to use email validation. Procedures for setting up an email address are outside the scope of this guide.
- Validation applies only to certificates that ACM issues from Amazon CAs, whose `Type` is `AMAZON_ISSUED`: public certificates and Amazon N2 certificates. ACM does not validate domain ownership for imported certificates or for certificates signed by a private CA. ACM cannot validate resources in an Amazon VPC private hosted zone or any other private domain. For more information, see Troubleshoot certificate validation.
- To switch an existing certificate from email validation to DNS validation, use the `UpdateCertificateOptions` API.

Certificate expiration and renewal

ACM certificates are valid for 198 days. Renewing a certificate requires action by the domain owner. ACM begins sending renewal notices to the email addresses associated with the domain 45 days before expiration. The notifications contain a link that the domain owner can click for renewal. Once all listed domains are validated, ACM issues a renewed certificate with the same ARN.

(Optional) Resend validation email

Each validation email contains a token that you can use to approve a certificate request. However, because the validation email required for the approval process can be blocked by spam filters or lost in transit, the token automatically expires after 72 hours. If you do not receive the original email or the token has expired, you can request that the email be resent. For information about how to resend a validation email, see Resend validation email

For persistent problems with email validation, see the Troubleshoot email validation problems section in Troubleshoot issues with AWS Certificate Manager.

Automate AWS Certificate Manager email validation

> **Important**
>
> Throughout 2027, AWS Certificate Manager will deprecate email validation for new public certificates and renewals. Use DNS validation instead. For more information, see AWS Certificate Manager will discontinue email validation to prove domain validation for certificates on the AWS Security Blog.

Email-validated ACM certificates normally require manual action by the domain owner. Organizations dealing with large numbers of email-validated certificates may prefer to create a parser that can automate the required responses. To assist customers using email validation, the information in this section describes the templates used for domain validation email messages and the workflow involved in completing the validation process.

Validation email templates

Validation email messages have one of the two following formats, depending on whether a new certificate is being requested or an existing certificate is being renewed. The content of the highlighted strings should be replaced with values that are specific to the domain being validated.

Validating a new certificate

Email template text:

```
Greetings from Amazon Web Services,

We received a request to issue an SSL/TLS certificate for requested_domain.

Verify that the following domain, AWS account ID, and certificate identifier
 correspond
to a request from you or someone in your organization.

Domain: fqdn
AWS account ID: account_id
AWS Region name: region_name
Certificate Identifier: certificate_identifier

To approve this request, go to Amazon Certificate Approvals
(https://region_name.acm-certificates.amazon.com/approvals?
code=validation_code&context=validation_context)
and follow the instructions on the page.

This email is intended solely for authorized individuals for fqdn. To express any
 concerns
about this email or if this email has reached you in error, forward it along with a
 brief
explanation of your concern to validation-questions@amazon.com.

Sincerely,
Amazon Web Services
```

Validating a certificate for renewal

Email template text:

```
Greetings from Amazon Web Services,

We received a request to issue an SSL/TLS certificate for requested_domain.
This email is a request to validate ownership of the domain in order to renew
the existing, currently in use, certificate. Certificates have defined
validity periods and email validated certificates, like this one, require you
to re-validate for the certificate to renew.

Verify that the following domain, AWS account ID, and certificate identifier
correspond to a request from you or someone in your organization.

Domain: fqdn
AWS account ID: account_id
AWS Region name: region_name
Certificate Identifier: certificate_identifier

To approve this request, go to Amazon Certificate Approvals at
https://region_name.acm-certificates.amazon.com/approvals?code=
$validation_code&context=$validation_context
and follow the instructions on the page.

This email is intended solely for authorized individuals for fqdn. You can see
more about how AWS Certificate Manager validation works here -
https://docs.aws.amazon.com/acm/latest/userguide/email-validation.html.
To express any concerns about this email or if this email has reached you in
error, forward it along with a brief explanation of your concern to
validation-questions@amazon.com.

Sincerely,
Amazon Web Services

--
Amazon Web Services, Inc. is a subsidiary of Amazon.com, Inc. Amazon.com is a
registered trademark of Amazon.com, Inc.

This message produced and distributed by Amazon Web Services, Inc.,
410 Terry Ave. North, Seattle, WA 98109-5210.

(c)2015-2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
```

```
Our privacy policy is posted at https://aws.amazon.com/privacy
```

Once you receive a new validation message from AWS, we recommend that you use it as the most up-to-date and authoritative template for your parser. Customers with message parsers designed before November, 2020, should note the following changes that may have been made to the template:

- The email subject line now reads "`Certificate request for` *`domain name`*" instead of "`Certificate approval for` *`domain name`*".
- The `AWS account ID` is now presented without dashes or hyphens.
- The `Certificate Identifier` now presents the entire certificate ARN instead of a shortened form, for example, *`arn:aws:acm:us-east-1:000000000000:certificate/3b4d78e1-0882-4f51-954a-298ee44ff369`* rather than *`3b4d78e1-0882-4f51-954a-298ee44ff369`*.
- The certificate approval URL now contains `acm-certificates.amazon.com` instead of `certificates.amazon.com`.
- The approval form opened by clicking the certificate approval URL now contains the approval button. The name of the approval button div is now `approve-button` instead of `approval_button`.
- Validation messages for both newly requested certificates and renewing certificates have the same email format.

Validation workflow

This section provides information about the renewal workflow for email-validated certificates.

- When the ACM console processes a multi-domain certificate request, it sends validation email messages to the domain name or the validation domain that you specify when you request a public certificate. The domain owner needs to validate an email message for each domain before ACM can issue the certificate. For more information, see Using Email to Validate Domain Ownership.
- Email validation for multi-domain certificate requests using the ACM API or CLI results in an email message being sent by each requested domain, even if the request includes subdomains of other domains in the request. The domain owner needs to validate an email message for each of these domains before ACM can issue the certificate.

If you resend emails for an existing certificate through the ACM console, emails will be sent to the validation domain specified in the original certificate request, or the exact domain if no validation domain was specified. To receive validation emails at a different domain, you can request a new certificate, specifying the validation domain that you want to use for validation. Alternatively, you can call ResendValidationEmail with the `ValidationDomain` parameter using the API, SDK, or CLI. However, the validation domain specified in the `ResendValidationEmail` request is only used for that call and is not saved to the certificate Amazon Resource Name (ARN) for future validation emails. You must call `ResendValidationEmail` each time you wish to receive a validation email at a domain name that was not specified in the original certificate request.

> **Note**
>
> Prior to November, 2020, customers needed to validate only the apex domain and ACM would issue a certificate that also covered any subdomains. Customers with message parsers designed before that time should note the change to the email validation workflow.

- With the ACM API or CLI, you can force all validation email messages for a multi-domain certificate request to be sent to the apex domain. In the API, use the `DomainValidationOptions` parameter of the RequestCertificate action to specify a value for `ValidationDomain`, which is a member of the DomainValidationOption type. In the CLI, use the **--domain-validation-options** parameter of the request-certificate command to specify a value for `ValidationDomain`.

AWS Certificate Manager HTTP validation

Hypertext Transfer Protocol (HTTP) is a foundational protocol for data communication on the World Wide Web. When you choose HTTP validation for certificates used with CloudFront, ACM leverages this protocol to verify your domain ownership. ACM works in conjunction with CloudFront to provide you with a specific URL and a unique token that must be made accessible at that URL on your domain. This token serves as proof that you control the domain. By setting up a redirect from your domain to an ACM-controlled location within the CloudFront infrastructure, you demonstrate your ability to modify content on the domain, thus validating your ownership. This seamless integration between ACM and CloudFront simplifies the certificate issuance process, especially for CloudFront distributions.

Important

HTTP validation does not support wildcard domain certificates (such as *.example.com). For wildcard certificates, you must use either DNS validation or email validation instead.

For example, if you request a certificate for the `example.com` domain with `www.example.com` as an additional name using CloudFront, ACM provides you with two sets of URLs for HTTP validation. Each set contains a `redirectFrom` URL and a `redirectTo` URL, created specifically for your domain and your AWS account. The `redirectFrom` URL is a path on your domain (for example, `http://example.com/.well-known/pki-validation/example.txt`) that you need to configure. The `redirectTo` URL points to an ACM-controlled location within the CloudFront infrastructure where a unique validation token is stored. You need to set up these redirects only once. When a certificate authority attempts to validate your domain ownership, it will request the file from the `redirectFrom` URL, which CloudFront redirects to the `redirectTo` URL, allowing access to the validation token. ACM automatically renews your certificate as long as the certificate is in use with CloudFront and your redirect remains in place.

Once you've set up HTTP validation for a fully qualified domain name (FQDN) with CloudFront, you can request additional ACM certificates for that FQDN without repeating the validation process, as long as the HTTP redirect remains in place. This means you can create replacement certificates with the same domain name. You can also replace a deleted certificate without going through the validation process again, provided the redirect is still active.

To stop automatic renewal of your HTTP-validated certificate, you have two options. You can either remove the certificate from the CloudFront distribution with which it is associated, or you can delete the HTTP redirect you set up for validation. If you're using a content delivery network (CDN) or web server other than CloudFront to manage your redirects, consult their documentation to learn how to remove a redirect. If you're using CloudFront to manage your redirects, you can remove the redirect by updating your distribution's configuration. For more information about managed certificate renewal, see Managed certificate renewal in AWS Certificate Manager. Remember that stopping automatic renewal may lead to certificate expiration, which could interrupt your HTTPS traffic.

How HTTP redirects for ACM work

> **Note**
>
> This section is for customers who are using CloudFront for content delivery and ACM for SSL/TLS certificate management.

When using HTTP validation with ACM and CloudFront, you need to set up HTTP redirects. These redirects allow ACM to verify your domain ownership for initial certificate issuance and ongoing automated renewal. The redirect mechanism works by pointing a specific URL on your domain to an ACM-controlled location within the CloudFront infrastructure where a unique validation token is stored.

The following table shows example redirect configurations for domain names. Note that HTTP validation does not support wildcard domains (such as *.example.com). Each configuration's **Redirect From**-**Redirect To** pair serves to authenticate domain name ownership.

Example HTTP redirect configurations

Domain name	Redirect From	Redirect To	Comment
example.com	`http://example.com/.well-known/pki-validation/ x2.txt`	`https://validation. region.acm-validations.aws/ y2/.well-known/pki-validation/ x2.txt`	Unique
www.example.com	`http://www.example.com/.well-known/pki-validation/ x3.txt`	`https://validation. region.acm-validations.aws/ y3/.well-known/pki-validation/ x3.txt`	Unique
host.example.com	`http://host.example.com/.well-`	`https://validation. region.acm-`	Unique

Domain name	Redirect From	Redirect To	Comment
	known/pki-validation/ *x4*.txt	validations.aws/ *y4*/.well-known/pki-validation/ *x4*.txt	
subdomain.example.com	http://subdomain.example.com/.well-known/pki-validation/ *x5*.txt	https://validation. *region*.acm-validations.aws/ *y5*/.well-known/pki-validation/ *x5*.txt	Unique
host.subdomain.example.com	http://host.subdomain.example.com/.well-known/pki-validation/ *x6*.txt	https://validation. *region*.acm-validations.aws/ *y6*/.well-known/pki-validation/ *x6*.txt	Unique

The *xN* values in the file names and the *yN* values in the ACM-controlled domains are unique identifiers generated by ACM. For example,

```
http://example.com/.well-known/pki-validation/3639ac514e785e898d2646601fa951d5.txt
```

is representative of a resulting generated **Redirect From** URL. The associated **Redirect To** URL might be

```
https://validation.region.acm-validations.aws/98d2646601fa/.well-known/pki-
validation/3639ac514e785e898d2646601fa951d5.txt
```

for the same validation record.

> **Note**
>
> If your web server or content delivery network does not support setting up redirects at the specified path, see Troubleshoot HTTP Validation Problems.

When you request a certificate and specify HTTP validation, ACM provides redirect information in the following format:

Domain Name	Redirect To
example.com	https://validation.*region*.acm-validations.aws/*a424c7224e9b* /.well-known/pki-validation/*a79865eb4cd1a6ab990a45779b4e0b96* .txt

Domain Name is the FQDN associated with the certificate. *Redirect From* is the URL on your domain where ACM will look for the validation file. *Redirect To* is the ACM-controlled URL where the actual validation file is hosted.

You need to configure your web server or CloudFront distribution to redirect requests from the *Redirect From* URL to the *Redirect To* URL. The exact method for setting up this redirect depends on your web server software or CloudFront configuration. Ensure that the redirect is set up correctly to allow ACM to validate your domain ownership and issue or renew your certificate.

Setting up HTTP validation

ACM uses HTTP validation to verify your domain ownership when issuing public SSL/TLS certificates for use with CloudFront. This section describes how to configure a public certificate to use HTTP validation.

To set up HTTP validation in the console

> **Note**
>
> This procedure assumes that you have already requested a certificate through CloudFront and that you're working in the AWS Region where you created it. HTTP validation is available only through the CloudFront Distribution Tenants feature.

1. Open the ACM console at https://console.aws.amazon.com/acm/.
2. In the list of certificates, choose the **Certificate ID** of a certificate with status **Pending validation** that you want to configure. This opens a details page for the certificate.
3. In the **Domains** section, you can see the **Redirect From** and **Redirect To** values for each domain in your certificate request.
4. For each domain, set up an HTTP redirect from the **Redirect From** URL to the **Redirect To** URL. You can do this through your CloudFront distribution configuration.
5. Configure your CloudFront distribution to redirect requests from the **Redirect From** URL to the **Redirect To** URL. The method for setting up this redirect depends on your CloudFront configuration.
6. After you set up the redirects, ACM automatically attempts to validate your domain ownership. This process can take up to 30 minutes.

If ACM can't validate the domain name within 72 hours from the time it generates the redirect values for you, ACM changes the certificate status to **Validation timed out**. The most likely reason for this result is that you didn't successfully set up the HTTP redirects. To fix this issue, you must request a new certificate after reviewing the redirect instructions.

Important

To avoid validation problems, make sure that the content at the **Redirect From** location matches the content at the **Redirect To** location. If you encounter problems, see Troubleshooting HTTP validation problems.

Note

Unlike DNS validation, you can't programmatically request that ACM automatically create your HTTP redirects. You must configure these redirects through your CloudFront distribution settings.

For more information about how HTTP validation works, see How HTTP redirects for ACM work.

Migrating from email to DNS validation

You can migrate an existing email-validated public AWS Certificate Manager (ACM) certificate to DNS validation. The certificate ARN does not change during migration. Your existing infrastructure references—such as load balancer configurations and CI/CD pipelines— continue to work without modification.

DNS-validated certificates can be renewed automatically by ACM. Email-validated certificates require you to manually approve a validation email for every renewal. As certificate validity periods decrease, migrating to DNS validation reduces the operational overhead of certificate renewal.

How migration works

When you initiate migration, ACM generates one CNAME record for each domain on the certificate. You have up to 72 hours to add these records to your DNS configuration. After ACM verifies the records, the certificate switches from email validation to DNS validation. The certificate ARN remains the same.

- **Certificate ARN** — The ARN is preserved. You don't need to update any references to the certificate.
- **Certificate validity** — In most cases, the existing expiration date is preserved. If the certificate is within the renewal window, ACM also renews the certificate during migration and extends the expiration date.

- **Auto-renewal** — After migration, the certificate behaves like any other DNS-validated ACM certificate. As long as DNS validation records remain in place, ACM automatically renews the certificate before expiration without requiring you to approve validation emails.

> **Important**
>
> You can only migrate from email to DNS validation. Migration from DNS to email validation is not supported.

You can initiate a migration through the ACM console or by calling the UpdateCertificateOptions API operation with the `ValidationMethod` option set to `DNS`. For step-by-step instructions, see Migrating a certificate (console or AWS CLI).

Monitor migration

To monitor migration progress, use the certificate details page in the ACM console or call the ListCertificateDomainValidations API operation. The response returns a domain validation summary for each domain on the certificate. The summary includes the following fields:

- **ActiveValidationConfiguration**—Shows the validation method currently in effect. While migration is in progress, the active method is `EMAIL`.
- **RequestedValidationConfiguration**—Shows the validation configuration that ACM is attempting to switch to (`DNS`), including the CNAME record you must add and the current validation status (`PENDING_VALIDATION` until verified).

Migration is complete when ACM completes DNS validation for every domain in the certificate. For a Java code example, see Listing domain validation status.

If ACM doesn't verify the CNAME records within 72 hours, the migration request expires and the certificate remains email-validated. You can reinitiate migration at any time, which starts a new 72-hour window. If your migration timed out and you can't determine why, see My email-to-DNS migration timed out.

Migration eligibility

You can migrate a certificate when all of the following conditions are true:

- The certificate is a public certificate issued by ACM.

- The certificate currently uses email validation.

> **Note**
>
> To find your email-validated certificates, use certificate search to filter by validation method. For more information, see certificate search.

- The certificate status is **Issued**.
- The certificate does not have another migration request already in progress.

You cannot migrate the following certificates:

- Private certificates issued by AWS Private Certificate Authority.
- Certificates that already use DNS validation.
- Imported certificates.
- Certificates with a status other than **Issued**.

Migrating a certificate (console or AWS CLI)

To migrate a certificate from email to DNS validation (console)

> **Note**
>
> The exact label of the migration action in the console might vary. Look for the option to switch the validation method to DNS in the **Actions** menu on the certificate details page.

1. Open the ACM console at https://console.aws.amazon.com/acm/.
2. In the list of certificates, choose the **Certificate ID** of an issued, email-validated certificate.
3. Choose **Actions**, and then choose **Migrate to DNS validation**.
4. Review the generated CNAME records for each domain on the certificate.
5. Add each CNAME record to the DNS configuration for its domain. If your domain is hosted in Route 53 and you have permission to write to the zone, you can choose **Create records in Route 53** to add the records automatically.
6. Choose **Migrate** to confirm the request.

7. Monitor the certificate's domain validation status on the certificate details page. Migration completes after ACM verifies all CNAME records.

> **Tip**
>
> To migrate multiple certificates at the same time, select more than one certificate from the certificate list and then choose **Migrate to DNS validation**.

To migrate a certificate from email to DNS validation (AWS CLI)

1. Initiate the migration by calling the update-certificate-options command with the `ValidationMethod` option set to `DNS`. Replace the example ARN with your certificate ARN.

```
aws acm update-certificate-options \
    --certificate-arn arn:aws:acm:us-
east-1:111122223333:certificate/12345678-1234-1234-1234-123456789012 \
    --options ValidationMethod=DNS
```

2. Retrieve the CNAME records that you must add to your DNS configuration by calling the list-certificate-domain-validations command.

```
aws acm list-certificate-domain-validations \
    --certificate-arn arn:aws:acm:us-
east-1:111122223333:certificate/12345678-1234-1234-1234-123456789012
```

   For each domain, the response includes a `RequestedValidationConfiguration` with a CNAME record under `DnsValidationChallenge.ResourceRecord`.

3. Add each CNAME record to the DNS configuration for its domain.

4. Wait for ACM to verify the CNAME records. Verify the migration progress by calling `list-certificate-domain-validations` again. When migration completes, the `ActiveValidationConfiguration` for each domain shows `ValidationMethod` set to `DNS`.

Private certificates in AWS Certificate Manager

If you have access to an existing private CA created by AWS Private CA, AWS Certificate Manager (ACM) can request a certificate suited for use in your private key infrastructure (PKI). The CA may either reside in your account or be shared with you by a different account. For information about creating a private CA, see Create a Private Certificate Authority.

Certificates signed by a private CA are not trusted by default, and ACM does not support any form of validation for them. Consequently, an administrator must take action to install them in your organization's client trust stores.

Private ACM certificates follow the X.509 standard and are subject to the following restrictions:

- **Names:** You must use DNS-compliant subject names. For more information, see Domain Names.
- **Algorithm:** For encryption, the certificate private key algorithm must be either 2048-bit RSA, 256-bit ECDSA, or 384-bit ECDSA.

  > **Note**
  >
  > The specified signing algorithm family (RSA or ECDSA) must match the algorithm family of the CA's secret key.

- **Expiration:** Each private certificate is valid for 13 months (395 days). The end date of the signing CA certificate must exceed the end date of the requested certificate, or else the certificate request will fail.

  > **Note**
  >
  > Private certificates have a longer validity period than public certificates. Public ACM certificates are valid for 198 days. For more information about public certificates, see Request a public certificate in AWS Certificate Manager.

- **Renewal:** ACM attempts to renew a private certificate automatically after 11 months.

The private CA used to sign the end-entity certificates is subject to its own restrictions:

- The CA must have a status of Active.

> **Note**
>
> Unlike publicly trusted certificates, certificates signed by a private CA do not require validation.

Topics

- Conditions for using AWS Private CA to sign ACM private certificates
- Request a private certificate in AWS Certificate Manager
- Export an AWS Certificate Manager private certificate

Conditions for using AWS Private CA to sign ACM private certificates

You can use AWS Private CA to sign your ACM certificates in either of two cases:

- **Single account**: The signing CA and the AWS Certificate Manager (ACM) certificate that is issued reside in the same AWS account.

  For single-account issuance and renewals to be enabled, the AWS Private CA administrator must grant permission to the ACM service principal to create, retrieve, and list certificates. This is done using the AWS Private CA API action CreatePermission or the AWS CLI command create-permission. The account owner assigns these permissions to an IAM user, group, or role that is responsible for issuing certificates.
- **Cross-account**: The signing CA and the ACM certificate that is issued reside in different AWS accounts, and access to the CA has been granted to the account where the certificate resides.

  To enable cross-account issuance and renewals, the AWS Private CA administrator must attach a resource-based policy to the CA using the AWS Private CA API action PutPolicy or the AWS CLI command put-policy. The policy specifies principals in other accounts that are allowed limited access to the CA. For more information, see Using a Resource Based Policy with ACM Private CA.

  The cross-account scenario also requires ACM to set up a service-linked role (SLR) to interact as a principal with the PCA policy. ACM creates the SLR automatically while issuing the first certificate.

  ACM might alert you that it cannot determine whether an SLR exists on your account. If the required `iam:GetRole` permission has already been granted to the ACM SLR for your account,

then the alert will not recur after the SLR is created. If it does recur, then you or your account administrator might need to grant the `iam:GetRole` permission to ACM, or associate your account with the ACM-managed policy `AWSCertificateManagerFullAccess`.

For more information, see Using a Service Linked Role with ACM.

Important

Your ACM certificate must be actively associated with a supported AWS service before it can be automatically renewed. For information about the resources that ACM supports, see Managed automation with integrated services.

Request a private certificate in AWS Certificate Manager

Request a private certificate (console)

1. Sign into the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/home.

   Choose **Request a certificate**.
2. On the **Request certificate** page, choose **Request a private certificate** and **Next** to continue.
3. In the **Certificate authority details** section, select the **Certificate authority** menu and choose one of the available private CAs. If the CA is shared from another account, the ARN is prefaced by ownership information.

   Details about the CA are displayed to help you verify that you have chosen the correct one:

   - **Owner**
   - **Type**
   - **Common name (CN)**
   - **Organization (O)**
   - **Organization unit (OU)**
   - **Country name (C)**
   - **State or province**
   - **Locality name**

4. In the **Domain names** section, type your domain name. You can use a fully qualified domain name (FQDN), such as **`www.example.com`**, or a bare or apex domain name such as **`example.com`**. You can also use an asterisk (**`*`**) as a wild card in the leftmost position to protect several site names in the same domain. For example, **`*.example.com`** protects **`corp.example.com`**, and **`images.example.com`**. The wildcard name will appear in the **Subject** field and in the **Subject Alternative Name** extension of the ACM certificate.

   > **Note**
   >
   > When you request a wildcard certificate, the asterisk (**`*`**) must be in the leftmost position of the domain name and can protect only one subdomain level. For example, **`*.example.com`** can protect **`login.example.com`**, and **`test.example.com`**, but it cannot protect **`test.login.example.com`**. Also note that **`*.example.com`** protects *only* the subdomains of **`example.com`**, it does not protect the bare or apex domain (**`example.com`**). To protect both, see the next step

   Optionally, choose **Add another name to this certificate** and type the name in the text box. This is useful for authenticating both a bare or apex domain (such as **`example.com`**) and its subdomains such as **`*.example.com`**).

5. In the **Key algorithm** section, choose an algorithm.

   For information to help you choose an algorithm, see the AWS blog post How to evaluate and use ECDSA certificates in AWS Certificate Manager.

6. In the **Tags** section, you can optionally tag your certificate. Tags are key-value pairs that serve as metadata for identifying and organizing AWS resources. For a list of ACM tag parameters and for instructions on how to add tags to certificates after creation, see Tag AWS Certificate Manager resources.

7. In the **Certificate renewal permissions** section, acknowledge the notice about certificate renewal permissions. These permissions allow automatic renewal of private PKI certificates that you sign with the selected CA. For more information, see Using a Service Linked Role with ACM.

8. After providing all of the required information, choose **Request**. The console returns you to the certificate list, where you can view your new certificate.

> **Note**
>
> Depending on how you have ordered the list, a certificate you are looking for might not be immediately visible. You can click the black triangle at right to change the ordering. You can also navigate through multiple pages of certificates using the page numbers at upper-right.

Request a private certificate (CLI)

Use the request-certificate command to request a private certificate in ACM.

> **Note**
>
> When you request a private PKI certificate signed by a CA from AWS Private CA, the specified signing algorithm family (RSA or ECDSA) must match the algorithm family of the CA's secret key.

```
aws acm request-certificate \
--domain-name www.example.com \
--idempotency-token 12563 \
--certificate-authority-arn arn:aws:acm-pca:Region:444455556666:\
certificate-authority/CA_ID
```

This command outputs the Amazon Resource Name (ARN) of your new private certificate.

```
{
    "CertificateArn": "arn:aws:acm:Region:444455556666:certificate/certificate_ID"
}
```

In most cases, ACM automatically attaches a service-linked role (SLR) to your account the first time that you use a shared CA. The SLR enables automatic renewal of end-entity certificates that you issue. To check whether the SLR is present, you can query IAM with the following command:

```
aws iam get-role --role-name AWSServiceRoleForCertificateManager
```

If the SLR is present, the command output should resemble the following:

```
{
   "Role":{
      "Path":"/aws-service-role/acm.amazonaws.com/",
      "RoleName":"AWSServiceRoleForCertificateManager",
      "RoleId":"AAAAAAA0000000BBBBBBB",
      "Arn":"arn:aws:iam::{account_no}:role/aws-service-role/acm.amazonaws.com/
AWSServiceRoleForCertificateManager",
      "CreateDate":"2020-08-01T23:10:41Z",
      "AssumeRolePolicyDocument":{
         "Version":"2012-10-17",
         "Statement":[
            {
               "Effect":"Allow",
               "Principal":{
                  "Service":"acm.amazonaws.com"
               },
               "Action":"sts:AssumeRole"
            }
         ]
      },
      "Description":"SLR for ACM Service for accessing cross-account Private CA",
      "MaxSessionDuration":3600,
      "RoleLastUsed":{
         "LastUsedDate":"2020-08-01T23:11:04Z",
         "Region":"ap-southeast-1"
      }
   }
}
```

If the SLR is missing, see Using a Service Linked Role with ACM.

Export an AWS Certificate Manager private certificate

You can export a certificate issued by AWS Private CA for use anywhere in your private PKI environment. The exported file contains the certificate, the certificate chain, and the encrypted private key. This file must be stored securely. For more information about AWS Private CA, see AWS Private Certificate Authority User Guide.

> **Note**
>
> If you want to export public certificates issued through ACM, see ACM exportable public certificates.

Topics

- Export a private certificate (console)
- Export a private certificate (CLI)

Export a private certificate (console)

1. Sign into the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/home.
2. Choose **Certificate Manager**
3. Choose the link of the certificate that you want to export.
4. Choose **Export**.
5. Enter and confirm a passphrase for the private key.

   > **Note**
   >
   > When creating your passphrase, you can use any ASCII character except #, $, or %.

6. Choose **Generate PEM Encoding**.
7. You can copy the certificate, certificate chain, and encrypted key to memory or choose **Export to a file** for each.
8. Choose **Done**.

Export a private certificate (CLI)

Use the export-certificate command to export a private certificate and private key. You must assign a passphrase when you run the command. For added security, use a file editor to store your passphrase in a file, and then supply the passphrase by supplying the file. This prevents your passphrase from being stored in the command history and prevents others from seeing the passphrase as you type it in.

Note

The file containing the passphrase must not end in a line terminator. You can check your password file like this:

```
$ file -k passphrase.txt
passphrase.txt: ASCII text, with no line terminators
```

The following examples pipe the command output to `jq` to apply PEM formatting.

```
[Windows/Linux]
$ aws acm export-certificate \
     --certificate-arn arn:aws:acm:Region:444455556666:certificate/certificate_ID \
     --passphrase fileb://path-to-passphrase-file  \
     | jq -r '"\(.Certificate)\(.CertificateChain)\(.PrivateKey)"'
```

This outputs a base64-encoded, PEM-format certificate, also containing the certificate chain and encrypted private key, as in the following abbreviated example.

```
-----BEGIN CERTIFICATE-----
MIIDTDCCAjSgAwIBAgIRANWuFpqA16g3IwStE3vVpTwwDQYJKoZIhvcNAQELBQAw
EzERMA8GA1UECgwIdHJvbG9sb2wwHhcNMTkwNzE5MTYxNTU1WhcNMjAwODE5MTcx
NTU1WjAXMRUwEwYDVQQDDAx3d3cuc3B1ZHMuaW8wggEiMA0GCSqGSIb3DQEBAQUA
...
8UNFQvNoo1VtICL4cwWOdLOkxpwkkKWtcEkQuHE1v5Vn6HpbfFmxkdPEasoDhthH
FFWIf4/+VOlbDLgjU4HgtmV4IJDtqM9rGOZ42eFYmmc3eQO0GmigBBwwXp3j6hoi
74YM+igvtILnbYkPYhY9qz8h7lHUmannS8j6YxmtpPY=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIC8zCCAdugAwIBAgIRAM/jQ/6h2/MI1NYWX3dDaZswDQYJKoZIhvcNAQELBQAw
EzERMA8GA1UECgwIdHJvbG9sb2wwHhcNMTkwNjE5MTk0NTE2WhcNMjkwNjE5MjA0
NTE2WjATMREwDwYDVQQKDAh0cm9sb2xvbDCCASIwDQYJKoZIhvcNAQEBBQADggEP
...
j2PAOviqIXjwr08Zo/rTy/8m6LAsmm3LVVYKLyPdl+KB6M/+H93Z1/Bs8ERqqga/
6lfM6iw2JHtkW+q4WexvQSoqRXFhCZWbWPZTUpBS0d4/Y5q92S3iJLRa/JQ0d4U1
tWZyqJ2rj2RL+h7CE71XIAM//oHGcDDPaQBFD2DTisB/+ppGeDuB
-----END CERTIFICATE-----
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFKzBVBgkqhkiG9w0BBQ0wSDAnBgkqhkiG9w0BBQwwGgQUMrZb7kZJ8nTZg7aB
1zmaQh4vwloCAggAMB0GCWCGSAFlAwQBKgQQDViroIHStQgNOjR6nTUnuwSCBNAN
```

```
JM4SG202YPUiddWeWmX/RKGg3lIdE+A0WLTPskNCdCAHqdhOSqBwt65qUTZe3gBt
...
ZGipF/DobHDMkpwiaRR5sz6nG4wcki0ryYjAQrdGsR6EVvUUXADkrnrrxuHTWjFl
wEuqyd8X/ApkQsYFX/nhepOEIGWf8Xu0nrjQo77/evhG0sHXborGzgCJwKuimPVy
Fs5kw5mvEoe5DAe3rSKsSUJ1tM4RagJj2WH+BC04SZWNH8kxfOC1E/GSLBCixv3v
+Lwq38CEJRQJLdpta8NcLKnFBwmmVs9OV/VXzNuHYg==
-----END ENCRYPTED PRIVATE KEY-----
```

To output everything to a file, append the > redirect to the previous example, yielding the following.

```
$ aws acm export-certificate \
     --certificate-arn arn:aws:acm:Region:444455556666:certificate/certificate_ID \
     --passphrase fileb://path-to-passphrase-file \
     | jq -r '"\(.Certificate)\(.CertificateChain)\(.PrivateKey)"' \
     > /tmp/export.txt
```

Import certificates into AWS Certificate Manager

In addition to requesting SSL/TLS certificates provided by AWS Certificate Manager (ACM), you can import certificates that you obtained outside of AWS. You might do this because you already have a certificate from a third-party certificate authority (CA), or because you have application-specific requirements that are not met by ACM issued certificates.

You can use an imported certificate with any AWS service that is integrated with ACM. The certificates that you import work the same as those provided by ACM, with one important exception: ACM does not provide managed renewal for imported certificates.

To renew an imported certificate, you can obtain a new certificate from your certificate issuer and then manually reimport it into ACM. This action preserves the certificate's association and its Amazon Resource name (ARN). Alternatively, you can import a completely new certificate. Multiple certificates with the same domain name can be imported, but they must be imported one at a time.

Important

You are responsible for monitoring the expiration date of your imported certificates and for renewing them before they expire. You can simplify this task by using Amazon CloudWatch

Events to send notices when your imported certificates approach expiration. For more information, see Using Amazon EventBridge.

All certificates in ACM are regional resources, including the certificates that you import. To use the same certificate with Elastic Load Balancing load balancers in different AWS Regions, you must import the certificate into each Region where you want to use it. To use a certificate with Amazon CloudFront, you must import it into the US East (N. Virginia) Region. For more information, see Supported Regions.

For information about how to import certificates into ACM, see the following topics. If you encounter problems importing a certificate, see Certificate import problems.

Topics

- Prerequisites for importing ACM certificates
- Certificate and key format for importing
- Import a certificate
- Reimport a certificate

Prerequisites for importing ACM certificates

To import a self–signed SSL/TLS certificate into ACM, you must provide both the certificate and its private key. To import a certificate signed by a non-AWS certificate authority (CA), you must also include the private and public keys of certificate. Your certificate must satisfy all of the criteria described in this topic.

For all imported certificates, you must specify a cryptographic algorithm and a key size. ACM supports the following algorithms (API name in parentheses):

- RSA 1024 bit (`RSA_1024`)
- RSA 2048 bit (`RSA_2048`)
- RSA 3072 bit (`RSA_3072`)
- RSA 4096 bit (`RSA_4096`)
- ECDSA 256 bit (`EC_prime256v1`)
- ECDSA 384 bit (`EC_secp384r1`)

- ECDSA 521 bit (`EC_secp521r1`)

Note also the following additional requirements:

- ACM integrated services allow only the algorithms and key sizes that they support to be associated with their resources. For example, CloudFront only supports 1024-bit RSA, 2048-bit RSA, 3072-bit RSA, 4096-bit RSA, and Elliptic Prime Curve 256-bit keys, while Application Load Balancer supports all of the algorithms available from ACM. For more information, see the documentation for the service you are using.
- A certificate must be an SSL/TLS X.509 version 3 certificate. It must contain a public key, the fully qualified domain name (FQDN) or IP address for your website, and information about the issuer.
- A certificate can be self-signed by a private key that you own, or signed by the private key of an issuing CA. You must provide the private key, which may be no larger than 5 KB (5,120 bytes) and must be unencrypted.
- If the certificate is signed by a CA, and you choose to provide the certificate chain, the chain must be PEM–encoded.
- A certificate must be valid at the time of import. You cannot import a certificate before its validity period begins or after it expires. The `NotBefore` certificate field contains the validity start date, and the `NotAfter` field contains the end date.
- All of the required certificate materials (certificate, private key, and certificate chain) must be PEM–encoded. Uploading DER–encoded materials results in an error. For more information and examples, see Certificate and key format for importing.
- When you renew (reimport) a certificate, you cannot remove a `KeyUsage` or `ExtendedKeyUsage` extension that was present in the previously imported certificate.

  The following exceptions apply:

  - You can reimport a certificate missing the Client Authentication ExtendedKeyUsage when compared to the previous certificate. This accommodates industry changes where certificate authorities no longer issue certificates with ClientAuth EKU to comply with Chrome's root program requirements.
  - You can remove the `keyEncipherment` Key Usage from ECDSA certificates. This accommodates RFC 5480 Section 3, which does not include `keyEncipherment` as a permitted Key Usage for ECDSA keys.

> **Important**
>
> If you require Client Authentication functionality, you must implement additional validations on your side, as ACM does not support rollback to previously imported certificates.

- AWS CloudFormation does not support the import of certificates into ACM.

Certificate and key format for importing

ACM requires you to separately import the certificate, certificate chain, and private key (if any), and to encode each component in PEM format. PEM stands for Privacy Enhanced Mail. The PEM format is often used to represent certificates, certificate requests, certificate chains, and keys. The typical extension for a PEM–formatted file is `.pem`, but it doesn't need to be.

> **Note**
>
> AWS does not provide utilities for manipulating PEM files or other certificate formats. The following examples rely on a generic text editor for simple operations. If you need to perform more complex tasks (such as converting file formats or extracting keys), free and open-source tools such as OpenSSL are readily available.

The following examples illustrate the format of the files to be imported. If the components come to you in a single file, use a text editor (carefully) to separate them into three files. Note that if you edit any of the characters in a PEM file incorrectly or if you add one or more spaces to the end of any line, the certificate, certificate chain, or private key will be invalid.

Example 1. PEM–encoded certificate

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

Example 2. PEM–encoded certificate chain

A certificate chain contains one or more certificates. You can use a text editor, the `copy` command in Windows, or the Linux `cat` command to concatenate your certificate files into a chain. The

certificates must be concatenated in order so that each directly certifies the one preceding. If importing a private certificate, copy the root certificate last. The following example contains three certificates, but your certificate chain might contain more or fewer.

> **Important**
>
> Do not copy your certificate into the certificate chain.

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

Example 3. PEM–encoded private keys

X.509 version 3 certificates use public key algorithms. When you create an X.509 certificate or certificate request, you specify the algorithm and the key bit size that must be used to create the private–public key pair. The public key is placed in the certificate or request. You must keep the associated private key secret. Specify the private key when you import the certificate. The key must be unencrypted. The following example shows an RSA private key.

```
-----BEGIN PRIVATE KEY-----
Base64-encoded private key
-----END PRIVATE KEY-----
```

The next example shows a PEM–encoded elliptic curve private key. Depending on how you create the key, the parameters block might not be included. If the parameters block is included, ACM removes it before using the key during the import process.

```
-----BEGIN EC PARAMETERS-----
Base64-encoded parameters
-----END EC PARAMETERS-----
-----BEGIN EC PRIVATE KEY-----
```

```
Base64-encoded private key
-----END EC PRIVATE KEY-----
```

Import a certificate

You can import an externally obtained certificate (that is, one provided by a third-party trust services provider) into ACM by using the AWS Management Console, the AWS CLI, or the ACM API. The following topics show you how to use the AWS Management Console and the AWS CLI. Procedures for obtaining a certificate from a non-AWS issuer are outside the scope of this guide.

> **Important**
>
> Your selected signature algorithm must meet the Prerequisites for importing ACM certificates.

Topics

- Import (console)
- Import (AWS CLI)

Import (console)

The following example shows how to import a certificate using the AWS Management Console.

1. Open the ACM console at https://console.aws.amazon.com/acm/home. If this is your first time using ACM, look for the **AWS Certificate Manager** heading and choose the **Get started** button under it.
2. Choose **Import a certificate**.
3. Do the following:

   a. For **Certificate body**, paste the PEM-encoded certificate to import. It should begin with `-----BEGIN CERTIFICATE-----` and end with `-----END CERTIFICATE-----`.

   b. For **Certificate private key**, paste the certificate's PEM-encoded, unencrypted private key. It should begin with `-----BEGIN PRIVATE KEY-----` and end with `-----END PRIVATE KEY-----`.

   c. (Optional) For **Certificate chain**, paste the PEM-encoded certificate chain.

4. (Optional) To add tags to your imported certificate, choose **Tags**. A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value, both of which you define. You can use tags to organize your resources or track your AWS costs.
5. Choose **Import**.

Import (AWS CLI)

The following example shows how to import a certificate using the AWS Command Line Interface (AWS CLI). The example assumes the following:

- The PEM-encoded certificate is stored in a file named `Certificate.pem`.
- The PEM-encoded certificate chain is stored in a file named `CertificateChain.pem`.
- The PEM-encoded, unencrypted private key is stored in a file named `PrivateKey.pem`.

To use the following example, replace the file names with your own and type the command on one continuous line. The following example includes line breaks and extra spaces to make it easier to read.

```
$ aws acm import-certificate --certificate fileb://Certificate.pem \
      --certificate-chain fileb://CertificateChain.pem \
      --private-key fileb://PrivateKey.pem
```

If the `import-certificate` command is successful, it returns the Amazon Resource Name (ARN) of the imported certificate.

Reimport a certificate

If you imported a certificate and associated it with other AWS services, you can reimport that certificate before it expires while preserving the AWS service associations of the original certificate. For more information about AWS services integrated with ACM, see Managed automation with integrated services.

The following conditions apply when you reimport a certificate:

- You can add or remove domain names.
- You cannot remove all of the domain names from a certificate.

- If **Key Usage** extensions are present in the originally imported certificate, you can add new extension values, but you cannot remove existing values.

  Exception: You can remove the `keyEncipherment` Key Usage from ECDSA certificates. This accommodates RFC 5480 Section 3, which does not include `keyEncipherment` as a permitted Key Usage for ECDSA keys.
- If **Extended Key Usage** extensions are present in the originally imported certificate, you can add new extension values, but you cannot remove existing values.

  Exception: You can remove the Client Authentication Extended Key Usage. This accommodates industry changes where certificate authorities no longer issue certificates with ClientAuth EKU to comply with Chrome's root program requirements.

  > **Important**
  >
  > If you require Client Authentication functionality, you must implement additional validations on your side, as ACM does not support rollback to previously imported certificates.

- The key type and size cannot be changed.
- You cannot apply resource tags when reimporting a certificate.

Topics

- Reimport (console)
- Reimport (AWS CLI)

Reimport (console)

The following example shows how to reimport a certificate using the AWS Management Console.

1. Open the ACM console at https://console.aws.amazon.com/acm/home.
2. Select or expand the certificate to reimport.
3. Open the details pane of the certificate and choose the **Reimport certificate** button. If you selected the certificate by checking the box beside its name, choose **Reimport certificate** on the **Actions** menu.
4. For **Certificate body**, paste the PEM-encoded end-entity certificate.

5. For **Certificate private key**, paste the unencrypted PEM-encoded private key associated with the certificate's public key.
6. (Optional) For **Certificate chain**, paste the PEM-encoded certificate chain. The certificate chain includes one or more certificates for all intermediate issuing certification authorities, and the root certificate. If the certificate to be imported is self-assigned, no certificate chain is necessary.
7. Review the information about your certificate. If there are no errors, choose **Reimport**.

Reimport (AWS CLI)

The following example shows how to reimport a certificate using the AWS Command Line Interface (AWS CLI). The example assumes the following:

- The PEM-encoded certificate is stored in a file named `Certificate.pem`.
- The PEM-encoded certificate chain is stored in a file named `CertificateChain.pem`.
- (Private certificates only) The PEM-encoded, unencrypted private key is stored in a file named `PrivateKey.pem`.
- You have the ARN of the certificate you want to reimport.

To use the following example, replace the file names and the ARN with your own and type the command on one continuous line. The following example includes line breaks and extra spaces to make it easier to read.

> **Note**
>
> To reimport a certificate, you must specify the certificate ARN.

```
$ aws acm import-certificate --certificate fileb://Certificate.pem \
      --certificate-chain fileb://CertificateChain.pem \
      --private-key fileb://PrivateKey.pem \
      --certificate-
arn arn:aws:acm:region:123456789012:certificate/12345678-1234-1234-1234-12345678901
```

If the `import-certificate` command is successful, it returns the Amazon Resource Name (ARN) of the certificate.

Amazon N2 certificates

Amazon N2 certificates are AWS Certificate Manager (ACM) certificates from the Amazon N2 certificate authorities (CAs), an Amazon-operated CA hierarchy. Use them for authentication outside web browsers, such as mutual TLS (mTLS). Choose them when WebPKI certificates do not fit, for example because you need a validity period longer than WebPKI allows (up to 395 days with the Amazon N2 roots), and private certificates do not fit either, for example because the clients or servers that verify your certificates are in another organization. In mTLS, the client and the server each present a certificate and verify the other's. This guide calls the client or server that verifies a certificate the *verifying peer*. Use Amazon N2 certificates between services in a mesh, between an application and an API endpoint, between an on-premises client and a service that you operate, or between your service and a partner's client.

An Amazon N2 certificate chains to the root hierarchy of your choice. Choose the Amazon N2 roots when you can add them to the trust store of each verifying peer. Choose the Starfield Services Root Certificate Authority - G2 (SFG2) when the verifying peers already include it in their trust stores, as many existing clients and devices do. For the root programs that include SFG2, see the repository on the Amazon Trust Services website. To compare the root hierarchies, see Compare the Amazon N2 certificates root hierarchies.

Do not use Amazon N2 certificates for browser clients. ACM does not record Amazon N2 certificates in certificate transparency (CT) logs, so browsers that require CT reject them.

Topics

- How Amazon N2 certificates work in ACM
- Amazon N2 roots and trust
- Compare the Amazon N2 certificates root hierarchies
- When to use Amazon N2 certificates
- Amazon N2 certificate characteristics
- Request an Amazon N2 certificate

How Amazon N2 certificates work in ACM

You request an Amazon N2 certificate with the `RequestCertificate` action. Two optional parameters determine which certificate ACM issues: `RootHierarchyType` and `CertificateUsage`.

`RootHierarchyType`

The root hierarchy that ACM issues the certificate from. To request an Amazon N2 certificate, specify one of these values:

- `N2_AMAZON_MANAGED`: chains to the Amazon N2 roots.
- `N2_SFG2_AMAZON_MANAGED`: chains to the Starfield Services Root Certificate Authority - G2 (SFG2).

If you omit this parameter, ACM issues an ACM public certificate. Any other value fails with a `ValidationException`. You cannot specify `RootHierarchyType` together with `CertificateAuthorityArn`.

This parameter selects a type of hierarchy, not a specific root or intermediate CA. ACM chooses the CA within the hierarchy, and that choice can change over time without any change to your request.

`CertificateUsage`

The extended key usage (EKU) profile of the certificate: whether it can authenticate both a client and a server, or only a client. Specify this parameter only with an Amazon N2 `RootHierarchyType`. If you omit it, ACM issues the certificate with `CLIENT_AND_SERVER_AUTH`.

A request fails with a `ValidationException` if you specify `CertificateUsage` without an Amazon N2 `RootHierarchyType`, or if the combination is not supported. The following table lists the supported combinations.

`RootHierarchyType`	**`CertificateUsage`**	**Certificate can authenticate**
`N2_AMAZON_MANAGED`	`CLIENT_AND_SERVER_AUTH` (default when omitted)	A client and a server

RootHierarchyType	CertificateUsage	Certificate can authenticate
N2_AMAZON_MANAGED	CLIENT_AUTH_ONLY	A client only
N2_SFG2_AMAZON_MANAGED	CLIENT_AND_SERVER_AUTH (default when omitted)	A client and a server

N2_SFG2_AMAZON_MANAGED does not support CLIENT_AUTH_ONLY, because certificates that chain to SFG2 always include server authentication.

DescribeCertificate returns RootHierarchyType and CertificateUsage for Amazon N2 certificates. The Type of an Amazon N2 certificate is AMAZON_ISSUED, the same as other ACM-issued certificates.

For the RootHierarchyType and CertificateUsage values that ACM reports for every certificate type, see Types of certificates in AWS Certificate Manager.

Amazon N2 roots and trust

A peer validates an Amazon N2 certificate only if its trust store includes the root that the certificate chains to. Each verifying peer needs the following:

- N2_AMAZON_MANAGED: all of the Amazon N2 roots. Browsers and operating systems do not include them by default, so add them to the trust store of each verifying peer. Add all of them, not only the root that matches your key algorithm, because ACM chooses the CA within the hierarchy and that choice can change over time.
- N2_SFG2_AMAZON_MANAGED: the Starfield Services Root Certificate Authority - G2 (SFG2) root certificate. The peer must also not require certificate transparency.

The certificate chain that ACM returns with an Amazon N2 certificate depends on the root hierarchy:

- N2_AMAZON_MANAGED: the intermediate CA that issued the certificate. The chain does not include the Amazon N2 roots, so each verifying peer must already have them in its trust store.

- N2_SFG2_AMAZON_MANAGED: the intermediate CA that issued the certificate, and an Amazon SFG2 N2 root, which SFG2 cross-signs. That root chains to SFG2 in the verifying peer's trust store.

To download the Amazon N2 roots and the Amazon SFG2 N2 roots in PEM or DER format, see the repository on the Amazon Trust Services website. The repository also lists test sites that serve valid, revoked, and expired certificates for each root that has them.

The two root hierarchies are governed by different Certificate Policy and Certification Practice Statement (CP/CPS) documents. For the CP/CPS that applies to each root, see the repository on the Amazon Trust Services website.

> **Note**
>
> The Amazon N2 roots are separate from the Amazon Trust Services WebPKI roots that ACM public certificates chain to. For those roots, see AWS Certificate Manager public certificate characteristics and limitations.

Compare the Amazon N2 certificates root hierarchies

The following table compares the two root hierarchies for Amazon N2 certificates. To compare Amazon N2 certificates with public, private, and imported certificates, see Compare certificate types.

Characteristic	Amazon N2 roots (N2_AMAZON_MANAGED)	SFG2 (N2_SFG2_AMAZON_MANAGED)
Choose it when	You can add the Amazon N2 roots to the trust store of each verifying peer, or you need a validity period longer than 98 days	The verifying peers already include SFG2 in their trust stores
Chains to	The Amazon N2 roots, with no cross-signing	The Starfield Services Root Certificate Authority - G2 (SFG2) root certificate,

Characteristic	Amazon N2 roots (`N2_AMAZON_MANAGED`)	SFG2 (`N2_SFG2_AMAZON_MANAGED`)
		through an Amazon SFG2 N2 root, which SFG2 cross-signs
What each verifying peer needs	The Amazon N2 roots in its trust store	SFG2 in its trust store, and no certificate transparency requirement
`CertificateUsage` values	`CLIENT_AND_SERVER_AUTH` (default) or `CLIENT_AUTH_ONLY`	`CLIENT_AND_SERVER_AUTH` only
Maximum validity	395 days	98 days
Chain that ACM returns	The issuing intermediate CA	The issuing intermediate CA and the cross-signed Amazon SFG2 N2 root

For domain validation, key algorithms, export, certificate transparency logging, listing, integrated services, quotas, and renewal, see Amazon N2 certificate characteristics.

When to use Amazon N2 certificates

Use an Amazon N2 certificate when:

- A certificate must authenticate a client, for example in mTLS. A certificate with `CLIENT_AND_SERVER_AUTH` also includes server authentication, so the same certificate can authenticate a server.
- You want Amazon to operate the CAs, including for connections between your services and a partner's.

To choose a root hierarchy, see Compare the Amazon N2 certificates root hierarchies. To compare Amazon N2 certificates with public, private, and imported certificates, see Types of certificates in AWS Certificate Manager.

Amazon N2 certificate characteristics

The following characteristics apply to Amazon N2 certificates. Where a characteristic differs from an ACM public certificate, this page describes the difference.

Validity period

The maximum validity of an Amazon N2 certificate depends on its root hierarchy.

`N2_AMAZON_MANAGED`

395 days

`N2_SFG2_AMAZON_MANAGED`

98 days. This hierarchy follows CA/Browser Forum validity requirements, so its maximum decreases as those requirements change.

Certificate usage

Every Amazon N2 certificate can authenticate a client. With `CLIENT_AND_SERVER_AUTH` (the default), the certificate can also authenticate a server. With `CLIENT_AUTH_ONLY`, available only with `N2_AMAZON_MANAGED`, it authenticates only a client. For the supported combinations, see How Amazon N2 certificates work in ACM.

Key algorithms

ACM issues Amazon N2 certificates with the following key algorithms.

- RSA 2048 bit (`RSA_2048`)
- ECDSA 256 bit (`EC_prime256v1`)
- ECDSA 384 bit (`EC_secp384r1`)

Imported certificates also support RSA 3072 bit, RSA 4096 bit, and ECDSA 521 bit keys. For a comparison of RSA and ECDSA key sizes at equivalent security strengths, see Key algorithms.

Domain validation

Amazon N2 certificates support DNS validation only. Set `ValidationMethod` to `DNS` and create the CNAME record that ACM provides. For more information, see AWS Certificate Manager DNS validation.

HTTP validation and email validation are not available for Amazon N2 certificates. The domain names in an Amazon N2 certificate must be public DNS names that ACM can resolve.

The validation CNAME record for an Amazon N2 certificate is specific to the AWS Region and account, and it differs from the record for an ACM public certificate for the same domain. Create the record that ACM provides for your Amazon N2 certificate request, even if you already validated the domain for a public certificate.

Certificate transparency logging

ACM does not record Amazon N2 certificates in certificate transparency (CT) logs, because they are not intended for browser clients. If your request includes a CT logging preference, ACM accepts the request and ignores the preference.

Listing and finding

By default, `ListCertificates` does not return Amazon N2 certificates. As a result, the certificate pickers in integrated service consoles might not show Amazon N2 certificates. To list Amazon N2 certificates, specify `N2_AMAZON_MANAGED`, `N2_SFG2_AMAZON_MANAGED`, or both in the `RootHierarchyTypes` filter. With the filter, `ListCertificates` returns only certificates from the hierarchies that you specify. To list all of your certificates, call it once without the filter and once with it. The following AWS CLI command lists Amazon N2 certificates from both hierarchies.

```
aws acm list-certificates \
    --root-hierarchy-types N2_AMAZON_MANAGED N2_SFG2_AMAZON_MANAGED
```

`SearchCertificates` and `DescribeCertificate` return Amazon N2 certificates without a filter. To search for Amazon N2 certificates from one hierarchy, filter on `RootHierarchyType`:

```
aws acm search-certificates \
    --filter-statement '{"Filter": {"AcmCertificateMetadataFilter":
 {"RootHierarchyType": "N2_AMAZON_MANAGED"}}}'
```

AWS integrated services

You can associate an Amazon N2 certificate with the resource of an integrated service by specifying the certificate ARN. You do not need to export the certificate. For the list of integrated services, see Managed automation with integrated services.

Associate an Amazon N2 certificate only with a resource whose clients include the root in their trust store. If you associate an Amazon N2 certificate with a resource that serves browser clients, such as a public CloudFront distribution or an internet-facing Elastic Load Balancing listener, browsers reject the connection.

Quotas

Amazon N2 certificates count toward the **Number of ACM certificates** quota, the same as other ACM-issued certificates. For more information, see Quotas.

Export and revocation

Amazon N2 certificates are exportable by default. If you omit the `Export` option from the request, ACM stores the export preference as `ENABLED`. To issue an Amazon N2 certificate that cannot be exported, set `Export` to `DISABLED` in the request. You cannot change the export setting after the certificate is issued. ACM public certificates, by contrast, are not exportable unless you request them with `Export` set to `ENABLED`. For more information, see AWS Certificate Manager exportable public certificates.

You can revoke an exported Amazon N2 certificate with the console, the AWS CLI, or the `RevokeCertificate` API action. For more information, see Revoke an AWS Certificate Manager public certificate.

Amazon N2 certificates include certificate revocation list (CRL) and Online Certificate Status Protocol (OCSP) locations, so verifying peers that check revocation status detect a revoked certificate. For the certificate chain that ACM returns, see Amazon N2 roots and trust.

Renewal

ACM renews Amazon N2 certificates automatically while the DNS validation records stay in place. Renewal starts 60 days before expiration for `N2_AMAZON_MANAGED` and 30 days before expiration for `N2_SFG2_AMAZON_MANAGED`. Integrated services that use the certificate use the renewed certificate. ACM does not update copies that you exported, so export the certificate and install it again after each renewal. For more information, see Managed certificate renewal in AWS Certificate Manager.

Request an Amazon N2 certificate

You can request an Amazon N2 certificate with the AWS Command Line Interface (AWS CLI) or the ACM API.

To request an Amazon N2 certificate (AWS CLI)

1. Request the certificate. The following command requests an Amazon N2 certificate that chains to the Amazon N2 roots and can authenticate both a client and a server. Replace *`service.example.com`* with your domain name.

```
aws acm request-certificate \
    --domain-name service.example.com \
    --validation-method DNS \
    --key-algorithm EC_prime256v1 \
    --root-hierarchy-type N2_AMAZON_MANAGED \
    --certificate-usage CLIENT_AND_SERVER_AUTH
```

The command uses the following parameters:

- `--validation-method DNS`: required. Amazon N2 certificates support DNS validation only.
- `--key-algorithm`: `RSA_2048`, `EC_prime256v1`, or `EC_secp384r1`. For more information, see Key algorithms.
- `--root-hierarchy-type`: `N2_AMAZON_MANAGED` chains to the Amazon N2 roots. `N2_SFG2_AMAZON_MANAGED` chains to the Starfield Services Root Certificate Authority - G2 (SFG2). To choose, see Compare the Amazon N2 certificates root hierarchies.
- `--certificate-usage`: optional. `CLIENT_AND_SERVER_AUTH` is the default. To issue a certificate that authenticates only a client, specify `CLIENT_AUTH_ONLY`, which is available only with `N2_AMAZON_MANAGED`.

Amazon N2 certificates are exportable by default. To prevent export, for example when you use the certificate only with AWS integrated services, add `--options Export=DISABLED`.

The command returns the ARN of the new certificate.

```
{
    "CertificateArn": "arn:aws:acm:us-
east-1:111122223333:certificate/12345678-1234-1234-1234-123456789012"
}
```

2. Validate your domain. The following command returns the CNAME record that ACM created for validation. Replace the ARN with the ARN from the previous step.

```
aws acm describe-certificate \
    --certificate-arn arn:aws:acm:us-
east-1:111122223333:certificate/12345678-1234-1234-1234-123456789012 \
    --query "Certificate.DomainValidationOptions[].ResourceRecord"
```

Create the CNAME record with your DNS provider. For more information, see AWS Certificate Manager DNS validation.

3. Confirm issuance. Run the following command to check the status of the certificate.

```
aws acm describe-certificate \
    --certificate-arn arn:aws:acm:us-
east-1:111122223333:certificate/12345678-1234-1234-1234-123456789012 \
    --query "Certificate.[Status,RootHierarchyType,CertificateUsage]"
```

When the status is ISSUED, the certificate is ready to use.

4. Use the certificate in either of these ways:

- **With an AWS integrated service.** Specify the certificate ARN when you configure the resource. You do not need to export the certificate. The certificate pickers in some service consoles might not list Amazon N2 certificates, so enter the ARN directly. For more information, see AWS integrated services.
- **On your own clients or servers.** Export the certificate, and then install it. To export the certificate, run the following command. The response contains the certificate, the certificate chain, and the private key, encrypted with your passphrase.

```
aws acm export-certificate \
    --certificate-arn arn:aws:acm:us-
east-1:111122223333:certificate/12345678-1234-1234-1234-123456789012 \
    --passphrase fileb://passphrase.txt
```

In both cases, make sure that each verifying peer includes the root that the certificate chains to. For more information, see Amazon N2 roots and trust.

Certificate lifecycle automation

AWS Certificate Manager (ACM) can renew your certificates and deploy them where you use them. How you automate depends on where the certificate runs.

Method	Use it when
Managed certificate renewal in AWS Certificate Manager	You request a certificate from ACM. ACM issues a new certificate before the current one expires and keeps the same ARN, so you do not track expiration dates or request replacements.
Managed automation with integrated services	An integrated AWS service, such as Elastic Load Balancing or CloudFront, uses the certificate. The service uses each renewed certificate for you.
ACME certificate automation	You need public certificates on hosts that you manage, and you can run an ACME client such as Certbot or cert-manager. The client requests, renews, and installs the certificates.
AWS Workload Credentials Provider	You export ACM certificates to hosts that you manage and cannot use ACME. The provider deploys each renewed certificate to your hosts.

Topics

- Managed certificate renewal in AWS Certificate Manager
- Managed automation with integrated services
- ACME certificate automation
- AWS Workload Credentials Provider

Managed certificate renewal in AWS Certificate Manager

ACM provides managed renewal for your Amazon-issued SSL/TLS certificates. This means that ACM will either renew your certificates automatically (if you are using DNS validation), or it will send you email notices when expiration is approaching. These services are provided for both public and private ACM certificates.

A certificate is eligible for automatic renewal subject to the following considerations:

- ELIGIBLE if associated with another AWS service, such as Elastic Load Balancing or CloudFront.
- ELIGIBLE if exported since being issued or last renewed.
- ELIGIBLE if it is a private certificate issued by calling the ACM RequestCertificate API *and* then exported or associated with another AWS service.
- ELIGIBLE if it is a private certificate issued through the management console *and* then exported or associated with another AWS service.
- NOT ELIGIBLE if it is a private certificate issued by calling the AWS Private CA IssueCertificate API.
- NOT ELIGIBLE if imported.
- NOT ELIGIBLE if already expired.

> **Note**
>
> Certificates issued through ACME certificate automation are not eligible for ACM managed renewal. Renewal of ACME certificates is managed by the ACME client. For more information, see Issuing certificates through ACME.

Additionally, the following Punycode requirements relating to Internationalized Domain Names must be fulfilled:

1. Domain names beginning with the pattern "<character><character>--" must match "xn--".
2. Domain names beginning with "xn--" must also be valid Internationalized Domain Names.

Punycode examples

Domain Name	Fulfills #1	Fulfills #2	Allowe	Note
example.com	n/a	n/a	✓	Does not start with "<charact er><character>--"
a--exampl e.com	n/a	n/a	✓	Does not start with "<charact er><character>--"
abc--exam ple.com	n/a	n/a	✓	Does not start with "<charact er><character>--"
xn--xyz.com	Yes	Yes	✓	Valid Internationalized Domain Name (resolves to 簡.com)
xn--examp le.com	Yes	No	✗	Not a valid Internationalized Domain Name
ab--examp le.com	No	No	✗	Must start with "xn--"

When ACM renews a certificate, the certificate's Amazon Resource Name (ARN) remains the same. Also, ACM certificates are regional resources. If you have certificates for the same domain name in multiple AWS Regions, each of these certificates must be renewed independently.

Topics

- Renew ACM public certificates
- Private certificate renewal in AWS Certificate Manager
- Check a certificate's renewal status

Renew ACM public certificates

When issuing a managed, publicly trusted certificate, AWS Certificate Manager requires you to prove that you are the domain owner. This happens by means of DNS validation. Email validation is being deprecated; for certificates already validated by email, see Renewal for email-validated domains. When a certificate comes up for renewal, ACM uses the same method that you chose

earlier to re-validate your ownership. The following topics describe how the renewal process works in each case.

Topics

- Renewal for domains validated by DNS
- Renewal for email-validated domains
- Renewal for domains validated by HTTP

Renewal for domains validated by DNS

Managed renewal is fully automated for ACM certificates that were originally issued using DNS validation.

At 45 days prior to expiration, ACM checks for the following renewal criteria:

> **Note**
>
> Previously issued certificates with a 395-day validity period renew 60 days before expiration and receive a renewed validity period of 198 days. Certificates with a 198-day validity period renew 45 days before expiration.

- The certificate is currently in use by an AWS service.
- All required ACM-provided DNS CNAME records (one for each unique Subject Alternative Name) are present and accessible via public DNS.

If these criteria are met, ACM considers the domain names validated and renews the certificate.

ACM sends AWS Health events and Amazon EventBridge events if it can't automatically validate a domain during renewal. These events are sent at 30 days, 15 days, seven days, three days, and one day prior to expiration. For more information, see Amazon EventBridge support for ACM.

Renewal for email-validated domains

> **Important**
>
> Throughout 2027, AWS Certificate Manager will deprecate email validation for new public certificates and renewals. Use DNS validation instead. For more information, see AWS

Certificate Manager will discontinue email validation to prove domain validation for certificates on the AWS Security Blog.

ACM certificates are valid for 198 days. Renewing a certificate requires action by the domain owner. ACM begins sending renewal notices to the email addresses associated with the domain 45 days before expiration. The notifications contain a link that the domain owner can click for renewal. Once all listed domains are validated, ACM issues a renewed certificate with the same ARN.

ACM sends AWS Health events and Amazon EventBridge events if it can't automatically validate a domain during renewal. These events are sent at 30 days, 15 days, seven days, three days, and one day prior to expiration. For more information, see Amazon EventBridge support for ACM.

For more information about validation email messages, see AWS Certificate Manager email validation

To learn how you can respond programmatically to validation email, see Automate AWS Certificate Manager email validation.

Resend validation email

After you configure email validation for your domain when you request a certificate (see AWS Certificate Manager email validation), you can use the AWS Certificate Manager API to request that ACM send you a domain validation email for your certificate renewal. You should do this in the following circumstances:

- You used email validation when initially requesting your ACM certificate.
- Your certificate's renewal status is **pending validation**. For information about determining a certificate's renewal status, see Check a certificate's renewal status.
- You didn't receive or can't find the original domain validation email message that ACM sent for certificate renewal.

To send validation emails to a different domain than what you originally configured in your certificate request, you can use the ResendValidationEmail operation in the ACM API, AWS CLI, or AWS SDKs. ACM will send emails to the specified validation domain. You can access the AWS CLI in browser by using AWS CloudShell in supported Regions.

To request that ACM resend the domain validation email message (console)

1. Open the AWS Certificate Manager console at https://console.aws.amazon.com/acm/home.
2. Choose the **Certificate ID** of the certificate that requires validation.
3. Choose **Resend validation email**.

To request that ACM resend the domain validation email (ACM API)

Use the ResendValidationEmail operation in the ACM API. In doing so, pass the ARN of the certificate, the domain that requires manual validation, and domain where you want to receive the domain validation emails. The following example shows how to do this with the AWS CLI. This example contains line breaks to make it easier to read.

```
$ aws acm resend-validation-email \
 --certificate-arn arn:aws:acm:region:account:certificate/certificate_ID \
 --domain subdomain.example.com \
 --validation-domain example.com
```

Renewal for domains validated by HTTP

ACM provides automated managed renewal for certificates that were originally issued using HTTP validation through CloudFront.

At 45 days prior to expiration, ACM checks for the following renewal criteria:

> **Note**
>
> Previously issued certificates with a 395-day validity period renew 60 days before expiration and receive a renewed validity period of 198 days. Certificates with a 198-day validity period renew 45 days before expiration.

- The certificate is currently in use by CloudFront.
- All required HTTP validation records are accessible and contain the expected content.

If these criteria are met, ACM considers the domain names validated and renews the certificate.

ACM sends AWS Health events and Amazon EventBridge events if it can't automatically validate a domain during renewal. These events are sent at 30 days, 15 days, seven days, three days, and one day prior to expiration. For more information, see Amazon EventBridge support for ACM.

To ensure successful renewal, make sure that the content at the `RedirectFrom` location matches the content at the `RedirectTo` location for each domain in the certificate.

Private certificate renewal in AWS Certificate Manager

ACM certificates that were signed by a private CA from AWS Private CA are eligible for managed renewal. Unlike publicly trusted ACM certificates, a certificate for a private PKI requires no validation. Trust is established when an administrator installs the appropriate root CA certificate in client trust stores.

> **Note**
>
> Only certificates obtained using the ACM console or the RequestCertificate action of the ACM API are eligible for managed renewal. Certificates issued directly from AWS Private CA using the IssueCertificate action of the AWS Private CA API are not managed by ACM.

When a managed certificate is 60 days away from expiration, ACM automatically attempts to renew it. This includes certificates that were exported and installed manually (for example, in an on-premises data center). Customers can also force renewal at any time using the RenewCertificate action of the ACM API. For a sample Java implementation of forced renewal, see Renewing a certificate.

After renewal, a certificate's deployment into service occurs in one of the following ways:

- If the certificate **is** associated with an ACM integrated service, the new certificate replaces the old one without additional customer action.
- If the certificate **is not** associated with an ACM integrated service, customer action is required to export and install the renewed certificate. You can perform these actions manually, or with assistance from AWS Health, Amazon EventBridge, and AWS Lambda as follows. For more information, see Automate export of renewed certificates

Automate export of renewed certificates

The following procedure provides an example solution for automating export of your private PKI certificates when ACM renews them. This example only exports a certificate and its private key out of ACM; after export, the certificate must still be installed on its target device.

To automate certificate export using the console

1. Following procedures in the AWS Lambda Developer Guide, create and configure a Lambda function that calls ACM export API.

   a. Create a Lambda function.

   b. Create a Lambda execution role for your function and add the following trust policy to it. The policy grants permission to the code in your function to retrieve the renewed certificate and private key by calling the ExportCertificate action of the ACM API.

   JSON

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

2. Create a rule in Amazon EventBridge to listen for ACM health events and call your Lambda function when it detects one. ACM writes to an AWS Health event each time it attempts to renew a certificate. For more information about these notices, see Check the status using Personal Health Dashboard (PHD).

   Configure the rule by adding the following event pattern.

```
{
   "source":[
      "aws.health"
   ],
```

```
   "detail-type":[
      "AWS Health Event"
   ],
   "detail":{
      "service":[
         "ACM"
      ],
      "eventTypeCategory":[
         "scheduledChange"
      ],
      "eventTypeCode":[
         "AWS_ACM_RENEWAL_STATE_CHANGE"
      ]
   },
   "resources":[
      "arn:aws:acm:region:account:certificate/certificate_ID"
   ]
}
```

3. Complete the renewal process by manually installing the certificate on the target system.

Test managed renewal of private PKI certificates

You can use the ACM API or AWS CLI to manually test the configuration of your ACM managed renewal workflow. By doing so, you can confirm that your certificates will be renewed automatically by ACM prior to expiration.

> **Note**
>
> You can only test the renewal of certificates issued and exported by AWS Private CA.

When you use API actions or CLI commands described below, ACM attempts to renew the certificate. If the renewal succeeds, ACM updates the certificate metadata displayed in the management console or in API output. If the certificate is associated with an ACM integrated services, the new certificate is deployed and a renewal event is generated in Amazon CloudWatch Events. If the renewal fails, ACM returns a error and suggests remedial action. (You can view this information using the describe-certificate command.) If the certificate is not deployed through an integrated service, you still need to export it and manually install it on your resource.

> **Important**
>
> In order to renew your AWS Private CA certificates with ACM, you must first grant the ACM service principal permissions to do so. For more information, see Assigning Certificate Renewal Permissions to ACM.

To manually test certificate renewal (AWS CLI)

1. Use the renew-certificate command to renew a private exported certificate.

```
aws acm renew-certificate \
 --certificate-arn arn:aws:acm:region:account:certificate/certificate_ID
```

2. Then use the describe-certificate command to confirm that the certificate's renewal details have been updated.

```
aws acm describe-certificate \
 --certificate-arn arn:aws:acm:region:account:certificate/certificate_ID
```

To manually test certificate renewal (ACM API)

- Send a RenewCertificate request, specifying the ARN of the private certificate to renew. Then use the DescribeCertificate operation to confirm that the certificate's renewal details have been updated.

Check a certificate's renewal status

When you have attempted to renew a certificate, ACM provides a *Renewal status* information field in the certificate details. You can use the AWS Certificate Manager console, the ACM API, the AWS CLI, or the Health Dashboard to check the renewal status of an ACM certificate. If you use the console, AWS CLI, or ACM API, the renewal status can have one of the four possible status values listed below. Similar values are displayed if you use the Health Dashboard.

Pending automatic renewal

ACM is attempting to automatically validate the domain names in the certificate. For more information, see Renewal for domains validated by DNS. No further action is required.

Pending validation

ACM couldn't automatically validate one or more domain names in the certificate. You must take action to validate these domain names or the certificate won't be renewed. If you originally used email validation for the certificate, look for an email from ACM and then follow the link in that email to perform the validation. If you used DNS validation, check to make sure your DNS record exists and that your certificate remains in use.

Success

All domain names in the certificate are validated, and ACM renewed the certificate. No further action is required.

Failed

One or more domain names were not validated before the certificate expired, and ACM did not renew the certificate. You can request a new certificate.

A certificate is eligible for renewal if it is associated with another AWS service, such as Elastic Load Balancing or CloudFront, or if it has been exported since being issued or last renewed.

> **Note**
>
> It can take up to several hours for changes to the renewal status to become available. If a problem is encountered, the renewal request times out after 72 hours, and the renewal process must be repeated from the beginning. For troubleshooting help, see Troubleshoot certificate requests.

Topics

- Check the status (console)
- Check the status (API)
- Check the status (CLI)
- Check the status using Personal Health Dashboard (PHD)

Check the status (console)

The following procedure discusses how to use the ACM console to check the renewal status of an ACM certificate.

1. Open the AWS Certificate Manager console at https://console.aws.amazon.com/acm/home.
2. Expand a certificate to view its details.
3. Find the **Renewal status** in the **Details** section. If you don't see the status, ACM hasn't started the managed renewal process for this certificate.

Check the status (API)

For a Java example that shows how to use the DescribeCertificate action to check the status, see Describing a certificate.

Check the status (CLI)

The following example shows how to check the status of your ACM certificate renewal with the AWS Command Line Interface (AWS CLI).

```
aws acm describe-certificate \
 --certificate-arn arn:aws:acm:region:account:certificate/certificate_ID
```

In the response, note the value in the `RenewalStatus` field. If you don't see the `RenewalStatus` field, ACM hasn't started the managed renewal process for your certificate.

Check the status using Personal Health Dashboard (PHD)

ACM attempts to automatically renew your ACM certificate 45 days prior to expiration for public certificates and 60 days prior for private certificates. If ACM cannot automatically renew your certificate, it sends certificate renewal event notices to your Health Dashboard at 45 day (for private only), 30 day, 15 day, 7 day, 3 day, and 1 day intervals from expiration to inform you that you need to take action. The Health Dashboard is part of the AWS Health service. It requires no setup and can be viewed by any user that is authenticated in your account. For more information, see AWS Health User Guide.

> **Note**
>
> ACM writes successive renewal event notices to a single event in your PHD time line. Each notice overwrites the previous one until the renewal succeeds.

To use the Health Dashboard:

1. Log in to the Health Dashboard at https://phd.aws.amazon.com/phd/home#/.
2. Choose **Event log**.
3. For **Filter by tags or attributes**, choose **Service**.
4. Choose **Certificate Manager**.
5. Choose **Apply**.
6. For **Event category** choose **Scheduled Change**.
7. Choose **Apply**.

Managed automation with integrated services

AWS Certificate Manager supports a growing number of AWS services that can use ACM certificates directly.

> **Note**
>
> To automate public certificate issuance and renewal outside integrated services such as on Amazon EC2 instances, use the ACME protocol. For more information, see ACME certificate automation. Alternatively, if you cannot use ACME, you can automate exportable certificates issued from ACM through AWS Workload Credentials Provider.

> **Note**
>
> ACME certificates (certificates with `CertificateKeyPairOrigin` of `ACME`) cannot be used with AWS integrated services. To use certificates with the services listed below, use ACM-managed certificates with a key pair origin of `AWS_MANAGED` or `CUSTOMER_PROVIDED`.

> **Note**
>
> You can associate an Amazon N2 certificate with an integrated service by specifying the certificate ARN. You do not need to export the certificate.

`ListCertificates` excludes Amazon N2 certificates by default, so the certificate pickers in integrated service consoles might not show them. For more information about Amazon N2 certificates, see Amazon N2 certificates.

ACM certificates are supported by the following services:

Elastic Load Balancing

Elastic Load Balancing automatically distributes your incoming application traffic across multiple Amazon EC2 instances. It detects unhealthy instances and reroutes traffic to healthy instances until the unhealthy instances have been restored. Elastic Load Balancing automatically scales its request handling capacity in response to incoming traffic. For more information about load balancing, see the Elastic Load Balancing User Guide.

In general, to serve secure content over SSL/TLS, load balancers require that SSL/TLS certificates be installed on either the load balancer or the back-end Amazon EC2 instance. ACM is integrated with Elastic Load Balancing to deploy ACM certificates on the load balancer. For more information, see Create an Application Load Balancer

Amazon CloudFront

Amazon CloudFront is a web service that speeds up distribution of your dynamic and static web content to end users by delivering your content from a worldwide network of edge locations. When an end user requests content that you're serving through CloudFront, the user is routed to the edge location that provides the lowest latency. This ensures that content is delivered with the best possible performance. If the content is currently at that edge location, CloudFront delivers it immediately. If the content is not currently at that edge location, CloudFront retrieves it from the Amazon S3 bucket or web server that you have identified as the definitive content source. For more information about CloudFront, see the Amazon CloudFront Developer Guide.

To serve secure content over SSL/TLS, CloudFront requires that SSL/TLS certificates be installed on either the CloudFront distribution or on the backed content source. ACM is integrated with CloudFront to deploy ACM certificates on the CloudFront distribution. For more information, see Getting an SSL/TLS Certificate.

> **Note**
>
> To use an ACM certificate with CloudFront, you must request or import the certificate in the US East (N. Virginia) region.

Amazon Elastic Kubernetes Service

Amazon Elastic Kubernetes Service is a managed Kubernetes service that makes it easy to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane. For more information about Amazon EKS, see the Amazon Elastic Kubernetes Service User Guide.

You can use ACM with AWS Controllers for Kubernetes (ACK) to issue and export TLS certificates to your Kubernetes workloads. This integration enables you to secure Amazon EKS pods and terminate TLS at your Kubernetes Ingress or at an AWS load balancer. ACM automatically renews certificates and the ACK controller updates your Kubernetes Secrets with renewed certificates. For more information, see Secure Kubernetes Workloads with ACM Certificates.

Amazon Cognito

Amazon Cognito provides authentication, authorization, and user management for your web and mobile applications. Users can sign in directly with your AWS account credentials or through a third party such as Facebook, Amazon, Google, or Apple. For more information about Amazon Cognito, see Amazon Cognito Developer Guide.

When you configure a Cognito user pool to use an Amazon CloudFront proxy, CloudFront may put an ACM certificate in place to secure the custom domain. When this is the case, be aware that you must remove the certificate's association with CloudFront before you can delete it.

AWS Elastic Beanstalk

Elastic Beanstalk helps you deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications. AWS Elastic Beanstalk reduces management complexity. You simply upload your application and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and health monitoring. Elastic Beanstalk uses the Elastic Load Balancing service to create a load balancer. For more information about Elastic Beanstalk, see the AWS Elastic Beanstalk Developer Guide.

To choose a certificate, you must configure the load balancer for your application in the Elastic Beanstalk console. For more information, see Configuring Your Elastic Beanstalk Environment's Load Balancer to Terminate HTTPS.

AWS App Runner

App Runner is an AWS service that provides a fast, simple, and cost-effective way to deploy from source code or a container image directly to a scalable and secure web application in the AWS Cloud. You don't need to learn new technologies, decide which compute service to use, or know how to provision and configure AWS resources. For more information about App Runner, see the AWS App Runner Developer Guide.

When you associate custom domain names with your App Runner service, App Runner internally creates certificates that track domain validity. They're stored in ACM. App Runner doesn't delete these certificates for seven days after a domain is disassociated from your service or after the service is deleted. This entire process is automated and you don't need to add or manage any certificates yourself. For more information, see Managing custom domain names for an App Runner service in the *AWS App Runner Developer Guide*.

Amazon API Gateway

With the proliferation of mobile devices and growth of the Internet of Things (IoT), it has become increasingly common to create APIs that can be used to access data and interact with back-end systems on AWS. You can use API Gateway to publish, maintain, monitor, and secure your APIs. After you deploy your API to API Gateway, you can set up a custom domain name to simplify access to it. To set up a custom domain name, you must provide an SSL/TLS certificate. You can use ACM to generate or import the certificate. For more information about Amazon API Gateway, see the Amazon API Gateway Developer Guide.

AWS Nitro Enclaves

AWS Nitro Enclaves is an Amazon EC2 feature that allows you to create isolated execution environments, called *enclaves*, from Amazon EC2 instances. Enclaves are separate, hardened, and highly constrained virtual machines. They provide only secure local socket connectivity with their parent instance. They have no persistent storage, interactive access, or external networking. Users cannot SSH into an enclave, and the data and applications inside the enclave cannot be accessed by the parent instance's processes, applications, or users (including root or admin).

EC2 instances connected to Nitro Enclaves support ACM certificates. For more information, see AWS Certificate Manager for Nitro Enclaves.

> **Note**
>
> For publicly trusted certificates on Amazon EC2 instances, we recommend the ACME protocol, which automates certificate issuance and renewal directly on the instance, including instances that are not connected to a Nitro Enclave. For more information, see ACME certificate automation. Associating an ACM-managed certificate directly with an Amazon EC2 instance requires a Nitro Enclave.

AWS CloudFormation

CloudFormation helps you model and set up your Amazon Web Services resources. You create a template that describes the AWS resources that you want to use, such as Elastic Load Balancing or API Gateway. Then CloudFormation takes care of provisioning and configuring those resources for you. You don't need to individually create and configure AWS resources and figure out what's dependent on what; CloudFormation handles all of that. ACM certificates are included as a template resource, which means that CloudFormation can request ACM certificates that you can use with AWS services to enable secure connections. In addition, ACM certificates are included with many of the AWS resources that you can set up with CloudFormation.

For general information about CloudFormation, see the CloudFormation User Guide. For information about ACM resources supported by CloudFormation, see AWS::CertificateManager::Certificate.

With the powerful automation provided by CloudFormation, it is easy to exceed your certificate quota, especially with new AWS accounts. We recommend that you follow the ACM best practices for CloudFormation.

> **Note**
>
> If you create an ACM certificate with CloudFormation, the CloudFormation stack remains in the **CREATE_IN_PROGRESS** state. Any further stack operations are delayed until you act upon the instructions in the certificate validation email. For more information, see Resource Failed to Stabilize During a Create, Update, or Delete Stack Operation.

AWS Amplify

Amplify is a set of purpose-built tools and features that enables front-end web and mobile developers to quickly and easily build full-stack applications on AWS. Amplify provides two services: Amplify Hosting and Amplify Studio. Amplify Hosting provides a git-based workflow for hosting full-stack serverless web apps with continuous deployment. Amplify Studio is a visual development environment that simplifies the creation of scalable, full-stack web and mobile apps. Use Studio to build your front-end UI with a set of ready-to-use UI components, create an app backend, and then connect the two together. For more information about Amplify, see the AWS Amplify User Guide.

If you connect a custom domain to your application, the Amplify console issues an ACM certificate to secure it.

Amazon OpenSearch Service

Amazon OpenSearch Service is a search and analytics engine for use cases such as log analytics, real-time application monitoring, and click stream analysis. For more information, see the Amazon OpenSearch Service Developer Guide.

When you create an OpenSearch Service cluster that contains a custom domain and endpoint, you can use ACM to provision the associated Application Load Balancer with a certificate.

AWS Network Firewall

AWS Network Firewall is a managed service that makes it easy to deploy essential network protections for all of your Amazon Virtual Private Clouds (VPCs). For more information about Network Firewall, see the AWS Network Firewall Developer Guide.

Network Firewall firewall integrates with ACM for TLS inspection. If you use TLS inspection in Network Firewall, you must configure an ACM certificate for the decryption and re-encryption of the SSL/TLS traffic going through your firewall. For information about how Network Firewall works with ACM for TLS inspection, see Requirements for using SSL/TLS certificates with TLS inspection configurations in the *AWS Network Firewall Developer Guide*.

ACME certificate automation

AWS Certificate Manager (ACM) supports the Automated Certificate Management Environment (ACME) protocol, an industry-standard method for automating certificate issuance and lifecycle management. With ACME certificate automation, you can issue publicly trusted certificates for

workloads running on customer-managed infrastructure such as on-premises servers, Kubernetes clusters, and hybrid environments.

Topics

- What is ACME?
- How ACME works with ACM
- ACME certificate characteristics
- Working with ACME-issued certificates
- When to use ACME vs. RequestCertificate
- ACME persona model
- End-to-end workflow
- ACME endpoints
- ACME domain validation
- External account bindings
- Issuing certificates through ACME
- Issuing ACME certificates over AWS PrivateLink

What is ACME?

ACME is an internet protocol defined in RFC 8555 that automates certificate issuance through machine-to-machine communication. Instead of manual certificate requests, ACME clients communicate directly with an ACME server to request and obtain certificates automatically.

ACME clients such as Certbot and cert-manager are widely available and integrate with many application platforms. These clients handle the entire certificate lifecycle, including initial issuance and renewal before expiration.

How ACME works with ACM

ACM provides a managed ACME server that you access through ACME endpoints. The architecture separates into two planes:

- **Control plane:** PKI administrators use ACM APIs to create ACME endpoints, configure domain validations, and generate external account binding (EAB) credentials.

- **Data plane:** ACME clients interact with the ACME protocol endpoint to register accounts and request certificates for domains that the administrator has already validated.

Unlike a typical public ACME service, where the client proves domain ownership each time it requests a certificate, the ACM ACME server uses domains that an administrator approves in advance. This separates the administrator who controls which domains an endpoint can issue for from the application owners who request certificates. For more information, see ACME domain validation.

Certificates issued through ACME receive standard ACM ARNs and appear in your ACM certificate inventory.

ACME certificate characteristics

Certificates issued through ACME are publicly trusted ACM certificates. They share the same trust chain, key algorithms, and certificate transparency logging as other ACM public certificates. For information about the characteristics that apply to all ACM public certificates, see AWS Certificate Manager public certificate characteristics and limitations.

The following sections describe where ACME certificates differ from other ACM public certificates.

Revocation

Revoked end-entity certificates use OCSP and CRLs to verify and publish revocation information. Some customer firewalls might need additional rules to allow these mechanisms.

Use these URL wildcard patterns to identify revocation traffic:

- **OCSP**

  ```
  http://*.amazontrust.com
  ```

- **CRL**

  ```
  http://*.amazontrust.com/*
  ```

If more restrictive rules are needed, see the Amazon Trust Services website.

Certificate validity period

The validity period for ACME-issued certificates is 45 days, which is shorter than the validity period for other ACM public certificates. This shorter validity is offset by the fact that ACME clients renew certificates automatically. Using short-lived certificates also positions you for upcoming industry changes, because public certificate maximum validity periods are mandated to reduce over time and will be no longer than 47 days by early 2029.

Working with ACME-issued certificates

The following operational behaviors and constraints apply to ACME-issued certificates as ACM resources:

- **Private key:** The private key is generated and held by the ACME client. ACM never sees the private key.
- **AWS integrated services:** ACME-issued certificates cannot be bound to Managed automation with integrated services such as Elastic Load Balancing, CloudFront, or API Gateway. Use the certificate on your managed infrastructure where the ACME client holds the private key.
- **Renewal:** The ACME client requests a new certificate before the existing one expires. ACM managed renewal does not apply.
- **Revocation:** Revocation is initiated by the ACME client through the ACME endpoint's `revoke-cert` URL.
- **Deletion:** You can delete expired ACME-issued certificates by calling `DeleteCertificate`. ACM automatically removes ACME-issued certificates one year after expiration.
- **Key pair origin:** `CertificateKeyPairOrigin` is `ACME` on `DescribeCertificate`, `ListCertificates`, and `SearchCertificates` responses.
- **Unsupported ACM APIs:** `ExportCertificate`, `RevokeCertificate`, `RenewCertificate`, and `ResendValidationEmail` are not supported for ACME-issued certificates. Lifecycle is managed by the ACME client.

When to use ACME vs. RequestCertificate

Use the following guidance to determine which certificate issuance method fits your use case.

Use RequestCertificate when:

- You need certificates for AWS integrated services (Elastic Load Balancing, CloudFront, API Gateway).
- You want ACM to manage the private key.
- You want ACM managed renewal.

Use ACME when:

- You want to use industry-standard ACME clients (Certbot, cert-manager).
- You need certificates for customer-managed infrastructure (on-premises, Kubernetes, hybrid).
- You need automated certificate lifecycle without AWS SDK dependencies.
- You must keep the private key on your own systems. For example, you might terminate TLS directly on your servers, or you might need to satisfy compliance or regulatory requirements that the private key never be held by a third party.

ACME persona model

ACME certificate automation involves two roles:

PKI administrator

: Creates and manages ACME endpoints, configures domain validations, creates external account bindings, controls which domains can issue certificates, sets up IAM roles, and monitors certificate issuance through CloudWatch metrics and the ACM console.

Application owner

: Receives EAB credentials from the administrator, uses an ACME client to request certificates, and manages certificate renewal on their systems.

End-to-end workflow

The following steps describe the complete setup and usage flow for ACME certificate automation:

1. The PKI administrator creates an ACME endpoint and receives an endpoint URL.
2. The administrator creates domain validations to pre-approve domains and provisions the required CNAME records.
3. The administrator creates external account bindings to generate credentials for ACME clients.

4. The administrator distributes EAB credentials to application owners.
5. The application owner configures an ACME client with the EAB credentials, registers an account, and requests certificates.
6. Issued certificates appear in ACM with standard ARNs.
7. The PKI administrator monitors issuance success and failure through CloudWatch metrics and the ACM console's certificates dashboard. For more information, see Supported CloudWatch metrics.

ACME endpoints

An ACME endpoint is a customer-specific, managed ACME server with a unique URL. Each endpoint is configured to issue certificates from a specific Certificate Authority. When you create an endpoint, ACM provisions an RFC 8555-compliant ACME server that your ACME clients can connect to.

Endpoint directory URL

When you create an ACME endpoint, ACM assigns it a unique ACME directory URL. You provide this URL to your ACME clients so they can connect to the endpoint. Retrieve the directory URL from the endpoint details in the console, or from the `EndpointUrl` field returned by `DescribeAcmeEndpoint`. The directory URL has the following form:

```
https://acm-acme-enroll.region.api.aws/00000000-0000-0000-0000-000000000000/directory
```

Endpoint configuration

When you create an ACME endpoint, you configure the following settings. For ACME quotas, see Quotas.

`AuthorizationBehavior`

: `PRE_APPROVED`: Domain validation is handled by pre-configured domain validation resources rather than live ACME challenges.

`Contact`

: `REQUIRED` or `NOT_REQUIRED`: Specifies whether ACME clients must provide contact information when they register an account with the endpoint. In the console, this setting is **Account email registration**.

By default, an ACME account contains no identifying information. The contact field is an email address that a client can supply during account registration to associate the account with an identity. Setting `Contact` to `REQUIRED` lets a PKI administrator enforce that the endpoint only accepts account registrations that include contact information, and rejects registrations that omit it. Registered contact information is purely informational and is used by the PKI administrator to manage ACME accounts.

`CertificateAuthority`

`PublicCertificateAuthority`: Opts the endpoint in to issuing publicly trusted (WebPKI) certificates from Amazon Trust Services. You do not specify a particular CA; this field declares that the endpoint is associated with public certificate issuance.

`AllowedKeyAlgorithms`

`RSA_2048`, `EC_prime256v1`, `EC_secp384r1`: Adds an enforcement layer to the endpoint. If a certificate request contains an embedded public key whose algorithm does not match one of the allowed algorithms, the endpoint rejects the request. When you don't set `AllowedKeyAlgorithms`, the endpoint applies no key algorithm enforcement.

In the console, these algorithms appear as **Certificate key types**: `EC_prime256v1` is **ECDSA P-256**, `RSA_2048` is **RSA 2048**, and `EC_secp384r1` is **ECDSA P-384**.

`CertificateTags`

Tags that ACM automatically attaches to every certificate issued through this endpoint. Use certificate tags to organize certificates, track costs, or control access to certificates issued through ACME. In the console, these appear as **Certificate tags**, separate from the **Tags** applied to the endpoint resource itself.

Endpoint status lifecycle

An ACME endpoint transitions through the following statuses:

`ACTIVE`

The endpoint is operational and can accept ACME requests.

`DELETING`

The endpoint is being deleted.

`FAILED`

Endpoint creation failed. Check the `FailureReason` field for details.

Creating an ACME endpoint

You can create an ACME endpoint by using the ACM console or the AWS CLI.

To create an endpoint (console)

1. Sign in to the AWS Management Console and open the ACM console.
2. In the left navigation pane, under **ACME**, choose **Endpoints**.
3. Choose **Create ACME endpoint**.
4. For **Endpoint name**, enter a name for the endpoint. The name is stored as a tag on the endpoint, and you can change it later.
5. Note that **Endpoint type** is **Public**. ACME clients connect to the endpoint over the internet by using HTTPS.
6. (Optional) Under **Account email registration**, select **Enable contact information during ACME account registration** to require ACME clients to provide a contact email address when they register an account.
7. Note that **Certificate type** is **Public**. The endpoint issues publicly trusted certificates from Amazon Trust Services.
8. For **Certificate key types**, select at least one key algorithm: **ECDSA P-256** (default), **RSA 2048**, or **ECDSA P-384**.
9. (Optional) Under **Domains**, add one or more domains to validate for this endpoint. You can also add domains later from the endpoint details page. For more information, see ACME domain validation.
10. (Optional) Under **Tags**, add tags to the endpoint.
11. (Optional) Under **Certificate tags**, add tags that ACM automatically attaches to every certificate issued through this endpoint.
12. Choose **Create ACME endpoint**.

After the endpoint status changes to `ACTIVE`, retrieve the endpoint's directory URL and provide it to your ACME clients.

To create an endpoint (AWS CLI)

Run the following command to create an ACME endpoint:

```
aws acm create-acme-endpoint \
    --authorization-behavior PRE_APPROVED \
    --contact REQUIRED \
    --certificate-authority '{
        "PublicCertificateAuthority": {
            "AllowedKeyAlgorithms": ["RSA_2048", "EC_prime256v1", "EC_secp384r1"]
        }
    }'
```

The response includes the `AcmeEndpointArn`:

```
{
    "AcmeEndpointArn": "arn:aws:acm:region:111122223333:acme-
endpoint/00000000-0000-0000-0000-000000000000"
}
```

To retrieve the endpoint URL and configuration, run the following command:

```
aws acm describe-acme-endpoint \
    --acme-endpoint-arn arn:aws:acm:region:111122223333:acme-
endpoint/00000000-0000-0000-0000-000000000000
```

The following shows an example response. The `EndpointUrl` is the ACME directory URL that you provide to your ACME clients.

```
{
    "AcmeEndpoint": {
        "AcmeEndpointArn": "arn:aws:acm:region:111122223333:acme-
endpoint/00000000-0000-0000-0000-000000000000",
        "EndpointUrl": "https://acm-acme-
enroll.region.api.aws/00000000-0000-0000-0000-000000000000/directory",
        "Status": "ACTIVE",
        "AuthorizationBehavior": "PRE_APPROVED",
        "Contact": "REQUIRED",
        "CertificateAuthority": {
            "PublicCertificateAuthority": {
                "AllowedKeyAlgorithms": [
```

```
                    "RSA_2048",
                    "EC_prime256v1",
                    "EC_secp384r1"
                ]
            }
        },
        "CreatedAt": "2026-06-18T20:35:06.331000-04:00",
        "UpdatedAt": "2026-06-18T20:35:06.331000-04:00"
    }
}
```

Managing ACME endpoints

You can perform the following management operations on ACME endpoints:

Describe

View endpoint details including status and configuration.

List

View all ACME endpoints in your account.

Update

Modify authorization behavior, contact requirements, or certificate authority settings.

Delete

Remove an endpoint. Deleting an endpoint also deletes its external account binding and domain validation resources, along with any ACME accounts registered with the endpoint.

Monitoring ACME endpoints

You can monitor ACME endpoint activity and certificate issuance through the ACM console and Amazon CloudWatch.

Console monitoring tab

In the ACM console, select an ACME endpoint and choose the **Monitoring** tab to view issuance metrics for that endpoint. The monitoring tab displays graphs for `CertificateIssuanceSuccess` and `CertificateIssuanceFailed` metrics over time.

Certificates dashboard

The ACM console **Certificates dashboard** provides an overview of all certificates in your account, including ACME-issued certificates. Use the dashboard to track certificate counts, expiration timelines, and renewal status across your inventory.

CloudWatch metrics

ACM publishes the following metrics to the `AWS/CertificateManager` namespace for each ACME endpoint:

- `CertificateIssuanceSuccess` – Count of certificates successfully issued through the endpoint.
- `CertificateIssuanceFailed` – Count of failed issuance attempts for the endpoint.

Both metrics use the `AcmeEndpointArn` dimension. You can create CloudWatch alarms on these metrics to be notified of issuance failures. For more information, see Supported CloudWatch metrics.

ACME domain validation

ACME domain validation resources pre-authorize which domains an ACME endpoint can issue certificates for. Unlike standard ACM domain validation (which you set up as part of a certificate request), ACME domain validations are persistent resources that the PKI administrator configures in advance. This separation enables application owners to request certificates without having to perform domain validation themselves.

Each domain validation requires a CNAME record in DNS. This is the same type of CNAME record used for standard ACM DNS validation. However, ACME domain validations are specific to individual endpoints. Different endpoints require separate CNAME records, even for the same domain.

How ACME domain validation relates to standard ACM validation

Both mechanisms use the same CNAME record format and purpose: proving domain ownership by placing a specific record in DNS, as described in AWS Certificate Manager DNS validation. In both cases, the CNAME delegates ongoing domain validation to ACM. Because the record points to a target that ACM manages, ACM can re-validate domain ownership over time without further action from you. This is what lets ACM renew certificates automatically. The following list describes the key differences:

- With standard ACM validation, you establish the CNAME as part of a certificate request, such as a call to `RequestCertificate`.
- ACME domain validation is a persistent ACM resource that an administrator configures in advance, independent of any individual certificate request.
- ACME domain validation includes a configurable scope that lets you control whether the endpoint can issue certificates for the exact domain, its subdomains, or wildcard names. For more information, see Domain validation scope.

Domain validation scope

When you create a domain validation, you configure a scope that controls what certificates can be issued using this validation. The scope has three independent settings. For definitions of *apex domain* and *subdomain*, see Domain Names.

`ExactDomain` (`ENABLED`/`DISABLED`)

Allow certificates for the exact domain that you specify. For example, if you specify the apex domain `example.com`, this setting allows certificates for `example.com`.

`Subdomains` (`ENABLED`/`DISABLED`)

Allow certificates for subdomains of the domain that you specify (for example, `www.example.com` or `api.example.com`).

`Wildcards` (`ENABLED`/`DISABLED`)

Allow wildcard certificates for the domain that you specify (for example, `*.example.com`).

You can combine these settings. The following table shows example scope combinations.

Domain validation scope combinations

DomainName	ExactDomain	Subdomains	Wildcards	Certificates allowed
example.com	ENABLED	DISABLED	DISABLED	example.com only
example.com	DISABLED	ENABLED	DISABLED	sub.example.com, api.example.com, and so on

DomainName	ExactDomain	Subdomains	Wildcards	Certificates allowed
example.com	DISABLED	DISABLED	ENABLED	*.example.com only
example.com	ENABLED	ENABLED	ENABLED	example.com, any subdomain, and *.example.com
internal.example.com	ENABLED	ENABLED	DISABLED	internal.example.com and its subdomains

Status lifecycle

After you create a domain validation, ACM attempts to verify the CNAME record for up to 72 hours. If the record is not detected within this period, the domain validation transitions to `INVALID` status. Make sure you provision the CNAME record promptly after creating the domain validation.

An ACME domain validation transitions through the following statuses:

`VALIDATING`

The CNAME record is being verified. ACM attempts to verify the record for up to 72 hours. If the record is not confirmed within this period, the status transitions to `INVALID` with a `TIMED_OUT` failure reason.

`VALID`

The CNAME record is confirmed. The domain validation is active and can be used for issuance.

`INVALID`

CNAME record verification failed. See the following failure reasons.

`DELETING`

The domain validation is being removed.

Failure reasons

`ACCESS_DENIED`

Insufficient permissions to verify the DNS record.

`DOMAIN_MISMATCH`

The CNAME record does not match expected values.

`HOSTED_ZONE_NOT_FOUND`

The specified hosted zone could not be found.

`INTERNAL_FAILURE`

An internal error occurred. Try creating the domain validation again.

`DOMAIN_NOT_ALLOWED`

The domain is not permitted for issuance. The domain may be on a restricted list or may not meet issuance requirements.

`CAA_ERROR`

A Certification Authority Authorization (CAA) DNS record prevents ACM from issuing for this domain. Ensure your CAA records allow Amazon to issue certificates.

`TIMED_OUT`

The CNAME record was not detected within 72 hours. Verify that the record has propagated in DNS and that it matches the expected name and value exactly.

Creating a domain validation

You can create an ACME domain validation by using the ACM console or the AWS CLI.

To create a domain validation (console)

1. Sign in to the AWS Management Console and open the ACM console.
2. In the left navigation pane, under **ACME**, choose **Endpoints**.
3. Select the endpoint to configure.
4. Choose the **Domains** tab.
5. Choose **Add domain**.
6. For **Domain name**, enter the domain name (for example, `example.com`).

7. Configure the scope settings for exact domain, subdomains, and wildcards.
8. (Optional) For **Hosted zone ID**, enter a Route 53 hosted zone ID for automatic CNAME provisioning.
9. (Optional) Under **Tags**, add one or more tags to the domain configuration.
10. Choose **Add domain configuration**.
11. If you are not using Route 53 automatic provisioning, provision the CNAME record in your DNS. The required CNAME name and value are shown in the domain configuration details.
12. Wait for the status to change to `VALID`.

To create a domain validation (AWS CLI)

Run the following command to create an ACME domain validation:

```
aws acm create-acme-domain-validation \
    --acme-endpoint-arn arn:aws:acm:region:111122223333:acme-
endpoint/00000000-0000-0000-0000-000000000000 \
    --domain-name example.com \
    --prevalidation-options '{
        "DnsPrevalidation": {
            "DomainScope": {
                "ExactDomain": "ENABLED",
                "Subdomains": "ENABLED",
                "Wildcards": "DISABLED"
            },
            "HostedZoneId": "Z1234567890"
        }
    }'
```

To check the status and get CNAME details, run the following command:

```
aws acm describe-acme-domain-validation \
    --acme-domain-validation-arn arn:aws:acm:region:111122223333:acme-
endpoint/00000000-0000-0000-0000-000000000000/acme-domain-
validation/11111111-1111-1111-1111-111111111111
```

Managing domain validations

You can perform the following management operations on ACME domain validations:

Describe

View status and CNAME record details.

List

View all domain validations for an endpoint.

Update

Modify the scope configuration.

Delete

Remove a domain validation. Certificates already issued are not affected.

External account bindings

External account bindings (EABs) are credentials that authorize ACME clients to register accounts with an ACME endpoint. An ACME client cannot request certificates from an endpoint without first registering an account, and registration requires valid EAB credentials. When a PKI administrator creates an EAB, they associate it with an IAM role that controls what certificate operations the ACME client can perform. The administrator then distributes the EAB credentials out-of-band to the teams or systems that need certificates.

Each EAB produces a key identifier (key ID) and a MAC key. The ACME client uses these credentials during account registration to prove authorization. After the ACME account is created, the account operates independently of the EAB. Revoking or deleting an EAB does not affect existing ACME accounts created with it.

IAM role requirements

Each EAB is associated with an IAM role. This role determines what the ACME client can do when issuing or revoking certificates. The role must meet the following requirements:

Trust policy

The role must allow the ACME service principal (`acm-acme.amazonaws.com`) to assume it. The trust policy must grant `sts:AssumeRole`, `sts:TagSession`, and `sts:SetSourceIdentity`. The condition on `sts:SourceIdentity` allows only the sessions that ACM establishes for ACME:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "acm-acme.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringLikeIfExists": {
                    "sts:SourceIdentity": "acm-acme-*"
                }
            }
        }
    ]
}
```

For the role session name, source identity, and session tag values that ACM sets, see IAM for ACME certificate automation.

Permissions

The role needs `acm:RequestCertificate` (for issuing) or `acm:RevokeCertificate` (for revoking), or both. These are the same permissions and condition keys used with standard ACM APIs. For more information, see Use condition keys with ACM.

PassRole

The administrator creating the EAB needs `iam:PassRole` permission for the role.

For complete IAM policy examples, see IAM for ACME certificate automation.

How the EAB role is used

When an ACME client issues or revokes a certificate, ACM uses the IAM role associated with the client's EAB to authorize the operation. Because issuance and revocation run under this role, ACME behaves like the standard ACM API:

- The same IAM policies and condition keys you use for direct ACM API calls apply to ACME issuance and revocation.
- AWS Organizations Service Control Policies (SCPs) are enforced at issuance time.
- Issuance and revocation operations are recorded in CloudTrail.
- ACM sets a role session name and source identity (and session tags) when it assumes the role, which you can reference with the `sts:RoleSessionName` and `sts:SourceIdentity` condition keys in the role's trust policy. For the values ACM uses, see IAM for ACME certificate automation.

Credential retrieval

After creating an EAB, retrieve its credentials by using `GetAcmeExternalAccountBindingCredentials`. This returns:

- **Key ID:** A unique identifier for the EAB (used as the `kid` in ACME EAB).
- **MAC Key:** A secret key used to create the HMAC signature during account registration (used as the `hmacKey`).

> **Important**
>
> Treat the MAC key as a secret. Distribute it only to the ACME clients that you authorize to use the endpoint.

Optional expiration

You can configure an expiration for the EAB credentials:

- **Value:** A positive integer.
- **Type:** `MINUTES`, `HOURS`, or `DAYS`.

If the credentials are not used before expiration, they become invalid. If no expiration is set, credentials remain valid until the EAB is revoked or deleted.

EAB and ACME account lifecycle

Once an ACME client registers an account using EAB credentials, the ACME account and EAB have independent lifecycles:

- Revoking an EAB does not affect existing ACME accounts created with it.
- Deleting an EAB does not deactivate accounts.
- An ACME account can continue issuing certificates after its originating EAB is revoked.

> **Note**
>
> To stop an ACME account from issuing more certificates, manage the account directly. You can revoke the account with the `RevokeAcmeAccount` API, or the account holder can deactivate it through the ACME protocol.

Creating an external account binding

You can create an EAB by using the ACM console or the AWS CLI.

To create an EAB (console)

1. Sign in to the AWS Management Console and open the ACM console.
2. In the left navigation pane, under **ACME**, choose **Endpoints**.
3. Select the endpoint.
4. Choose the **External account bindings** tab.
5. Choose **Create external account binding**.
6. For **IAM role**, enter or select the ARN of the role. The role must have the correct trust policy.
7. (Optional) Configure an expiration for the credentials.
8. (Optional) Under **Tags**, add one or more tags to the external account binding.
9. Choose **Create**.
10. Copy the Key ID and MAC key.

> **Important**
>
> Treat the MAC key as a secret. You can retrieve the key identifier and MAC key again at any time by calling `GetAcmeExternalAccountBindingCredentials`.

To create an EAB (AWS CLI)

Run the following command to create an EAB:

```
aws acm create-acme-external-account-binding \
    --acme-endpoint-arn arn:aws:acm:region:111122223333:acme-
endpoint/00000000-0000-0000-0000-000000000000 \
    --role-arn arn:aws:iam::111122223333:role/AcmeIssuanceRole \
    --expiration '{"Value": 7, "Type": "DAYS"}'
```

Then retrieve the credentials:

```
aws acm get-acme-external-account-binding-credentials \
    --acme-external-account-binding-arn arn:aws:acm:region:111122223333:acme-
endpoint/00000000-0000-0000-0000-000000000000/acme-external-account-
binding/22222222-2222-2222-2222-222222222222
```

The response contains the key identifier and MAC key that you provide to the ACME client. You can retrieve these credentials again at any time by calling `GetAcmeExternalAccountBindingCredentials`.

```
{
    "KeyId": "e06b7974-a684-4cc2-b4b1-1fd87ac81baa",
    "MacKey": "txvweHF1-COvXAKNlGk-58SX6ZVzEaaXaBPrwNAKCTo"
}
```

Managing external account bindings

You can perform the following management operations on EABs:

Describe

View EAB details including role ARN, expiration, creation time, and last used.

List

View all EABs for an endpoint.

Revoke

Invalidate the EAB credentials. This does not affect existing ACME accounts created with the EAB. To manage those accounts, revoke them with the `RevokeAcmeAccount` API, or have the account holder deactivate them through the ACME protocol.

Delete

Remove the EAB resource entirely. Any ACME accounts already registered with the EAB continue to exist; deleting the EAB only prevents new ACME accounts from being registered with it.

Issuing certificates through ACME

After your PKI administrator sets up an ACME endpoint with domain validations and external account bindings, application owners can use ACME clients to issue certificates. This section describes the issuance flow, provides a brief example using Certbot, and explains how ACME-issued certificates appear in ACM.

How issuance works

The ACME certificate issuance flow follows the RFC 8555 protocol:

1. **Account registration:** The ACME client registers an account using the EAB credentials (key ID and MAC key). This is a one-time step per ACME client.
2. **Certificate order:** The client creates an order specifying the domain names for the certificate.
3. **Authorization:** The ACME server checks pre-configured domain validations. Because the endpoint uses `PRE_APPROVED` authorization, no live challenges are required from the client.
4. **Finalize:** The client submits a Certificate Signing Request (CSR) to finalize the order, and ACM issues the certificate.
5. **Download:** The client downloads the issued certificate chain.

> **Important**
>
> The private key is generated by the ACME client and never leaves the client's system.

Example: Issuing a certificate with Certbot

This example shows how to register an ACME account and request a certificate using Certbot.

Prerequisites

- An ACME endpoint in `ACTIVE` status (you have the directory URL)
- At least one domain validation in `VALID` status for the domain you want
- EAB credentials (key ID and MAC key) from your PKI administrator
- Certbot installed on your system

To register an account and request a certificate

> **Important**
>
> Certificate issuance through the ACM ACME endpoint can take up to two minutes. Configure your ACME client's issuance timeout to at least 120 seconds (2 minutes) to prevent the client from timing out before the certificate is delivered. Many ACME clients default to 30-90 seconds, which is not sufficient.

Run the following command:

```
certbot certonly \
    --standalone \
    --non-interactive \
    --agree-tos \
    --email user@example.com \
    --server https://acm-acme-
enroll.region.api.aws/00000000-0000-0000-0000-000000000000/directory \
    --eab-kid AAABBBCCC111222333 \
    --eab-hmac-key base64encodedmackey \
    --issuance-timeout 120 \
    --domain www.example.com
```

After successful issuance, Certbot stores the certificate and private key in its default location (typically `/etc/letsencrypt/live/www.example.com/`).

ACME certificates in ACM

Certificates issued through ACME automatically receive a standard ACM certificate ARN and appear in the ACM certificate inventory. However, ACME-issued certificates have important differences from certificates issued through `RequestCertificate`.

Key differences

- **Private key is client-side:** The certificate private key is generated and stored by the ACME client. ACM does not have access to the private key.
- **Cannot bind to integrated services:** Because AWS does not hold the private key, ACME certificates cannot be used with Elastic Load Balancing, CloudFront, or API Gateway.
- **Not in default list output:** ACME certificates do not appear in `ListCertificates` results by default. You must include the `CertificateKeyPairOrigins` filter with value `ACME` to include them. `SearchCertificates` returns ACME certificates by default.
- **Cannot export through ACM:** The `ExportCertificate` API cannot be used because the certificate is already exported by definition since the client has it.
- **Revocation through ACME only:** Use the ACME endpoint's revoke-cert URL. The ACM `RevokeCertificate` API does not work for ACME-issued certificates.
- **Renewal is client-driven:** ACM does not auto-renew ACME certificates. The ACME client must request a new certificate before the existing one expires.

Viewing ACME certificates

In the ACM console, use the filter to include certificates with an ACME key pair origin. For more information about finding certificates, see Search certificates.

To list ACME certificates (AWS CLI)

Use `list-certificates` with the `CertificateKeyPairOrigins` filter:

```
aws acm list-certificates \
    --certificate-key-pair-origins ACME
```

To search for ACME certificates (AWS CLI)

Use `search-certificates` with the `CertificateKeyPairOrigin` filter to find ACME certificates by additional criteria:

```
aws acm search-certificates \
    --filter-statement '{"Filter": {"AcmCertificateMetadataFilter":
 {"CertificateKeyPairOrigin": "ACME"}}}'
```

You can also search for certificates issued through a specific endpoint or ACME account by filtering on `AcmeEndpointArn` or `AcmeAccountId`.

To view certificate details (AWS CLI)

Run the following command:

```
aws acm describe-certificate \
    --certificate-arn arn:aws:acm:region:account-id:certificate/certificate-id
```

Revoking ACME certificates

To revoke a certificate issued through ACME, use the ACME protocol's revoke-cert endpoint. The ACME client handles this.

> **Note**
>
> ACME-issued certificates cannot be revoked using the ACM `RevokeCertificate` API. You must revoke these certificates through the ACME endpoint that issued them.

The following example shows revocation using Certbot:

```
certbot revoke \
    --cert-path /etc/letsencrypt/live/www.example.com/cert.pem \
    --server https://acm-acme-
enroll.region.api.aws/00000000-0000-0000-0000-000000000000/directory
```

The revocation is authorized by the IAM role associated with the client's external account binding, so the same IAM policies and SCPs that apply to the standard `acm:RevokeCertificate` action apply here. For more information about the required IAM role configuration, see IAM for ACME certificate automation.

Certificate renewal

ACME certificates are not renewed by ACM. Your ACME client is responsible for requesting a new certificate before the current one expires. Most ACME clients (including Certbot) support automatic renewal through scheduled tasks (such as cron jobs or systemd timers).

ARN stability across renewals

When an ACME client renews a certificate, ACM determines whether the new issuance is a renewal of an existing certificate or a distinct certificate. If the issuance comes from the same ACME account with the same certificate metadata (such as domain names and key algorithm), ACM updates the existing certificate ARN rather than creating a new one. This keeps the ARN stable across renewals, which simplifies tracking and automation.

If the original certificate has been removed from ACM (for example, because it expired more than one year ago and was automatically deleted), a subsequent issuance with the same metadata creates a new certificate ARN.

Certbot auto-renewal example (typically configured automatically during Certbot installation):

```
certbot renew --quiet
```

Issuing ACME certificates over AWS PrivateLink

AWS Certificate Manager (ACM) supports issuing certificates through the Automated Certificate Management Environment (ACME) protocol over AWS PrivateLink, so that ACME clients in your VPC can issue certificates without routing traffic over the public internet. AWS PrivateLink connects your VPC directly to ACM through the Amazon network without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection.

For more information about AWS PrivateLink and Amazon VPC endpoints, see Interface Amazon VPC endpoints (AWS PrivateLink) in the *Amazon Virtual Private Cloud User Guide*.

ACM offers two Amazon VPC endpoint services for ACME, one for each side of the protocol:

- `com.amazonaws.`*`region`*`.acm-acme` serves the ACME management operations that a PKI administrator uses to configure ACME, such as `CreateAcmeEndpoint`, `CreateAcmeDomainValidation`, and `CreateAcmeExternalAccountBinding`.
- `com.amazonaws.`*`region`*`.acm-acme-enroll` serves the ACME protocol itself, which ACME clients use to issue and revoke certificates.

The two are independent: an endpoint for one does not provide access to the other, and each takes its own endpoint policy. The rest of this section describes the endpoint for certificate issuance.

Considerations for ACME issuance VPC endpoints

Before you set up interface Amazon VPC endpoints for ACME certificate issuance, be aware of the following considerations:

- ACM might not support Amazon VPC endpoints for ACME in some Availability Zones. When you create an Amazon VPC endpoint, first check support in the management console. Unsupported Availability Zones are marked *Service not supported in this Availability Zone*.
- Amazon VPC endpoints do not support cross-Region requests. Ensure that you create your endpoint in the same Region where your ACME endpoint is configured.
- Amazon VPC endpoints only support Amazon provided DNS through Amazon Route 53. If you want to use your own DNS, you can use conditional DNS forwarding. For more information, see DHCP option sets in the *Amazon Virtual Private Cloud User Guide*.
- The security group attached to the Amazon VPC endpoint must allow incoming connections on port 443 from the private subnet of the VPC.

Creating the Amazon VPC endpoint for ACME certificate issuance

You can create an Amazon VPC endpoint for ACME certificate issuance using either the VPC console or the AWS Command Line Interface. For more information, see Create an interface endpoint in the *Amazon Virtual Private Cloud User Guide*.

When creating the endpoint, specify `com.amazonaws.`*`region`*`.acm-acme-enroll` as the service name.

> **Important**
>
> You must enable private DNS host names for the Amazon VPC endpoint. Without them, ACME clients cannot reach your ACME endpoint through the Amazon VPC endpoint.

Creating an Amazon VPC endpoint policy for ACME certificate issuance

You can attach a policy to your Amazon VPC endpoint to control access to ACME operations. The policy specifies the principals that can perform actions, the actions they can perform, and

the resources on which they can act. For more information, see Control access to Amazon VPC endpoints using endpoint policies in the *Amazon Virtual Private Cloud User Guide*.

The ACME issuance flow (account registration, order placement, authorization, finalize, and download) spans several operations. Most of them are evaluated against the ACME endpoint resource and do not correspond to a named IAM action. The finalize and revocation steps are evaluated against the certificate resource, as `acm:RequestCertificate` and `acm:RevokeCertificate`. An Amazon VPC endpoint policy that restricts access to a specific ACME endpoint must therefore cover both resource types.

Example Amazon VPC endpoint policy for ACME certificate issuance on a specific endpoint

The following policy restricts all ACME operations to a single endpoint. The first statement covers account registration, order placement, authorization, and download. Because those operations have no named IAM action for a policy to match, `"Action": "*"` is required. The second statement covers finalize and revocation.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": "*",
         "Resource": "arn:aws:acm:region:account-id:acme-endpoint/endpoint-id"
      },
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "acm:RequestCertificate",
            "acm:RevokeCertificate"
         ],
         "Resource": "arn:aws:acm:region:account-id:certificate/*"
      }
   ]
}
```

A request that arrives through the Amazon VPC endpoint must be allowed by both the endpoint policy and the IAM policies on the IAM role associated with the EAB credentials. Because the endpoint policy cannot identify the calling principal, use the role's identity policies to control

which principals can issue or revoke certificates. For more information, see IAM for ACME certificate automation.

AWS Workload Credentials Provider

The AWS Workload Credentials Provider automates the use of public, private, and Amazon N2 certificates exported from ACM. The provider periodically retrieves certificates and their private keys, writes them to configured paths, and optionally runs a command to reload dependent services such as web servers.

> **Tip**
>
> For publicly trusted certificates, you can instead use the ACME protocol to automate issuance and renewal directly on your hosts with industry-standard ACME clients, without exporting certificates from ACM. For more information, see ACME certificate automation.

You can use the Workload Credentials Provider with the following compute environments:

- Amazon Elastic Compute Cloud (Amazon EC2)
- On-premises servers with AWS credentials
- Containers on Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and Docker on Amazon EC2, where the provider runs as a sidecar

The Workload Credentials Provider uses IAM role assumption to retrieve certificates. On hosts, it runs as a system service on Linux and Windows under a dedicated low-privilege user and writes certificate files with restricted permissions. In containers, it runs as the container's process and writes certificate files to a shared volume.

> **Important**
>
> Only exportable certificates can be used with this provider. For more information, see AWS Certificate Manager exportable public certificates and Exporting a private certificate.

The AWS Workload Credentials Provider is open source. For source code and contributions, see the GitHub repository. Container deployments don't use the installer. For container setup steps, see Running as a Certificate Manager Sidecar in the repository's README.

How certificate automation works

The provider uses a scheduler that refreshes each configured certificate on a fixed interval:

- The refresh interval is 24 hours.
- On each cycle, the provider exports the certificate, compares it against the files on disk, and writes new files only if the content has changed.
- When certificate files are updated, the configured `refresh_command` runs (for example, to reload NGINX or Apache).
- If certificate content hasn't changed, the refresh command is skipped.

The provider performs an initial refresh on every system boot and shortly after service startup, with a small random jitter to avoid synchronized API calls when multiple providers start simultaneously across the fleet.

The provider also supports dynamic configuration reload, allowing you to add, remove, or modify certificates without reinstalling. For more information, see the section called “Dynamic configuration reload”.

Prerequisites

Before you install the provider, ensure you have the following:

- A Linux instance with systemd (Amazon Linux 2023, Ubuntu 20.04+, or RHEL 8+)
- Or a Windows instance running Windows Server 2016 or later with PowerShell 5.1 or later
- Administrator access to run the installer
- An exportable certificate in ACM
- An IAM role with ACM export permissions (see the section called “Required permissions”)
- AWS credentials available on the instance (instance profile, environment variables, or credential file)

Install the provider

The following table lists download links for pre-built provider binaries. Releases for Windows are signed. Once you have downloaded the binary for your platform, proceed to running the installer.

Platform	Architecture	Download URL	Checksum (SHA-256)
Linux	x86-64	Download for Linux x86-64 (3.2.0)	bccc74d27cd6da97937c131b028e6a9deae945b4cbec77c6fe63a27d465447e3
Linux	Aarch64	Download for Linux AArch64 (3.2.0)	3d939c48b9f1cc061027522398f744d2bc34900c862d6bdfb94fcab37aee9dc3
Windows	x86-64	Download for Windows x86-64 (3.2.0)	06e97a10dd58d88ec54d6dc57b3f1aade43fab9164e72c813373d5532daa4849

Linux

On Amazon EC2 and other Linux instances, you can install the provider using the RPM package (available for AL2023). You can also build and install from source.

Option A: Install the RPM package (AL2023)

1. **Install the package**

   Install the RPM package. The package manager finds the latest version for you:

   ```
   sudo dnf install aws-workload-credentials-provider
   ```

   The RPM package installs the binary and helper scripts to `/opt/aws/workload-credentials-provider/`. It creates the `aws-wcp` service user, then installs and starts

the AWS Secrets Manager and token services. It also stages the ACM certificate automation service, which you enable in a later step.

2. **Create a configuration file**

   Create a TOML configuration file with your certificate details. For examples, see the section called "Configure the provider".

3. **Enable certificate automation**

   Run the ACM setup script with your configuration file to install, enable, and start the certificate automation service:

   ```
   sudo /opt/aws/workload-credentials-provider/bin/install-acm.sh --config /path/to/your/config.toml
   ```

   The setup script accepts the following options:

   - `--config <file>` – The configuration file with ACM certificate entries.
   - `--no-start` – Install the service but don't start it.
   - `--no-privileges` – Skip Linux capabilities on the service.
   - `--no-sudoers` – Skip sudoers generation for refresh commands.

Option B: Build and install from source

1. **Build the provider from source**

   Alternatively, you can build the provider from source. Follow the instructions at Install Rust on the Rust website to install the Rust toolchain.

   ```
   cargo build --release
   ```

   The executable is at `target/release/aws-workload-credentials-provider`.

2. **Create a configuration file**

   Create a TOML configuration file with your certificate details. For examples, see the section called "Configure the provider". The installer copies this file to `/etc/aws-workload-credentials-provider/config.toml` automatically when you use the `--config` option.

3. **Run the installer**

   Run the install script as root:

   ```
   cd aws_workload_credentials_provider_common/configuration
   sudo ./install --config /path/to/your/config.toml
   ```

4. **(Optional) Provide AWS credentials**

   On Amazon EC2 instances with an attached instance profile, the provider obtains credentials automatically. For other environments, create a systemd override file to inject credentials:

   ```
   sudo mkdir -p /etc/systemd/system/aws-workload-credentials-provider-
   acm.service.d
   sudo tee /etc/systemd/system/aws-workload-credentials-provider-acm.service.d/
   creds.conf > /dev/null <<EOF
   [Service]
   Environment="AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE"
   Environment="AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
   Environment="AWS_REGION=us-west-2"
   EOF
   sudo systemctl daemon-reload
   sudo systemctl restart aws-workload-credentials-provider-acm
   ```

Windows

1. **Build the provider from source**

   Alternatively, you can build the provider from source for Windows. Follow the instructions at Install Rust on the Rust website to install the Rust toolchain.

   ```
   cargo build --release
   ```

   The executable is at `target\release\aws-workload-credentials-provider.exe`.

2. **Create a configuration file**

   Create a TOML configuration file with your certificate details. For examples, see the section called “Configure the provider”. The installer copies this file to `C:\ProgramData\AWS`

`\WorkloadCredentialsProvider\config.toml` automatically when you use the `-Config` parameter.

3. **Run the installer**

   Run the install script as Administrator:

   ```
   cd aws_workload_credentials_provider_common\configuration
   .\install.ps1 -Config C:\path\to\your\config.toml
   ```

   To install without starting the service:

   ```
   .\install.ps1 -Config C:\path\to\your\config.toml -NoStart
   ```

4. **(Optional) Provide AWS credentials**

   On Amazon EC2 instances with an attached instance profile, the provider obtains credentials automatically through IMDS. For other environments, set environment variables for the service through the Windows Registry:

   ```
   $regPath = "HKLM:\SYSTEM\CurrentControlSet\Services
   \AWSWorkloadCredentialsProvider-ACM"
   $envVars = @(
       "AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE",
       "AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
       "AWS_REGION=us-west-2"
   )
   New-ItemProperty -Path $regPath -Name "Environment" -Value $envVars -
   PropertyType MultiString -Force
   Restart-Service AWSWorkloadCredentialsProvider-ACM
   ```

Configure the provider

Create a TOML configuration file with your certificate details. The installer copies this file to `/etc/aws-workload-credentials-provider/config.toml` when you use the `--config` option.

NGINX

```
[logging]
log_level = "info"
```

```
log_to_file = true

[capabilities.acm]
enabled = true

[[capabilities.acm.certificates]]
certificate_arn = "arn:aws:acm:us-west-2:123456789012:certificate/
abcd1234-5678-90ab-cdef-EXAMPLE11111"
role_arn = "arn:aws:iam::123456789012:role/ACMExportRole"
certificate_path = "/etc/pki/tls/certs/example.com.crt"
private_key_path = "/etc/pki/tls/private/example.com.key"
chain_path = "/etc/pki/tls/certs/example.com-chain.pem"
refresh_command = "/usr/sbin/nginx -s reload"
```

Apache

```
[logging]
log_level = "info"
log_to_file = true

[capabilities.acm]
enabled = true

[[capabilities.acm.certificates]]
certificate_arn = "arn:aws:acm:us-west-2:123456789012:certificate/
abcd1234-5678-90ab-cdef-EXAMPLE11111"
role_arn = "arn:aws:iam::123456789012:role/ACMExportRole"
certificate_path = "/etc/ssl/certs/example.com.crt"
private_key_path = "/etc/ssl/private/example.com.key"
chain_path = "/etc/ssl/certs/example.com-chain.pem"
refresh_command = "/bin/systemctl reload httpd"
```

Apache (Windows)

```
[logging]
log_level = "info"
log_to_file = true

[capabilities.acm]
enabled = true

[[capabilities.acm.certificates]]
```

```
certificate_arn = "arn:aws:acm:us-west-2:123456789012:certificate/
abcd1234-5678-90ab-cdef-EXAMPLE11111"
role_arn = "arn:aws:iam::123456789012:role/ACMExportRole"
certificate_path = 'C:\Apache24\conf\ssl\example.com.crt'
private_key_path = 'C:\Apache24\conf\ssl\example.com.key'
chain_path = 'C:\Apache24\conf\ssl\example.com-chain.pem'
refresh_command = 'C:\Apache24\bin\httpd.exe -k restart'
```

Note

When `chain_path` is omitted, the certificate chain is appended to the file at `certificate_path` to produce a fullchain file. This is compatible with web servers that expect a single file containing the certificate and its chain.

Dynamic configuration reload

You can update the provider's configuration without reinstalling by running the `acm reload` command. This validates the new configuration, updates permissions to match the new certificate paths, and restarts the service.

Certificates removed from the configuration stop being refreshed. New certificates begin their first export immediately. Each certificate runs as an independent task, so a failure in one does not affect others.

Linux

```
aws-workload-credentials-provider acm reload --config /path/to/new-config.toml
```

Windows

```
& 'C:\Program Files\AWS\WorkloadCredentialsProvider\bin\aws-workload-credentials-
provider.exe' acm reload -Config C:\path\to\new-config.toml
```

Required permissions

Base credentials

The provider's base credentials (instance profile or environment) must be able to assume the role specified in each certificate's `role_arn`:

```
{
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::123456789012:role/ACMExportRole"
}
```

Certificate export role

The role specified in `role_arn` requires the following permissions:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "acm:ExportCertificate",
            "Resource": "arn:aws:acm:us-west-2:123456789012:certificate/*"
        }
    ]
}
```

The role's trust policy must allow the provider's base identity to assume it:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/EC2InstanceRole"
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
```

Refresh command permissions (Linux)

On Linux, the provider executes the configured `refresh_command` through sudo. The installer generates a sudoers entry at `/etc/sudoers.d/aws-workload-credentials-provider` that permits the provider user to run the exact configured command without a password prompt.

> **Important**
>
> Ensure `/etc/sudoers` includes the `/etc/sudoers.d` directory. The installer warns if this include directive is not present. Without it, the generated sudoers file has no effect and refresh commands will fail.

On Windows, the provider triggers the `refresh_command` as a SYSTEM scheduled task.

Verify the installation

After installation, verify the provider is running and certificates are being written:

Linux

1. **Check service status:**

   ```
   sudo systemctl status aws-workload-credentials-provider-acm
   ```

2. **Review provider logs:**

   ```
   cat /opt/aws/workload-credentials-provider/logs/acm_provider.log
   ```

3. **Verify certificate files exist:**

   ```
   ls -la /etc/pki/tls/certs/example.com.crt
   ls -la /etc/pki/tls/private/example.com.key
   ```

4. **Validate certificate content:**

   ```
   openssl x509 -in /etc/pki/tls/certs/example.com.crt -noout -subject -dates
   ```

Windows

1. **Check service status:**

```
Get-Service AWSWorkloadCredentialsProvider-ACM | Format-List Name, Status,
 StartType
```

2. **Review provider logs:**

```
Get-Content "C:\ProgramData\AWS\WorkloadCredentialsProvider\logs\acm_provider.log"
 -Tail 20
```

3. **Verify certificate files exist:**

```
Get-Item C:\certs\example.com.crt
Get-Item C:\certs\example.com.key
```

4. **Validate certificate content:**

```
$cert = New-Object
 System.Security.Cryptography.X509Certificates.X509Certificate2("C:\certs
\example.com.crt")
$cert | Select-Object Subject, NotBefore, NotAfter
```

Configuration reference

Field	Required	Default	Description
`logging.log_level`	No	`info`	Logging verbosity: `debug`, `info`, `warn`, `error`, `none`
`logging.log_to_file`	No	`true`	Write logs to file (`true`) or stdout/stderr (`false`)
`capabilities.acm.enabled`	No	`false`	Enable or disable the ACM capability
`certificates[].certificate_arn`	Yes	—	ARN of the ACM certificate to export

Field	Required	Default	Description
`certificates[].role_arn`	Yes	—	IAM role ARN to assume for the `ExportCertificate` call
`certificates[].certificate_path`	Yes	—	Absolute path to write the certificate file
`certificates[].private_key_path`	Yes	—	Absolute path to write the private key file
`certificates[].chain_path`	No	—	Absolute path to write the certificate chain. If omitted, chain is appended to `certificate_path`
`certificates[].refresh_command`	No	—	Command to run after certificate files are updated. Must be an absolute path
`certificates[].certificate_and_chain_permission`	No	`0600`	File permissions for cert and chain files (Linux `mode` format)
`certificates[].key_permission`	No	`0600`	File permissions for the private key file (Linux `mode` format)

Container deployments use additional fields, such as `watcher_id`, `watcher_config_dir`, and `watcher_requests_dir`. For container setup, see the README in the GitHub repository linked earlier on this page.

Logging

On Linux, the provider logs to `/opt/aws/workload-credentials-provider/logs/acm_provider.log`.

On Windows, the provider logs to `C:\ProgramData\AWS\WorkloadCredentialsProvider\logs\acm_provider.log`. Service start and stop events are also recorded in the Windows Application Event Log under the source `AWSWorkloadCredentialsProvider-ACM`.

Log rotation — The provider creates a new log file when the current file reaches 10 MB, and stores up to five archived log files.

AWS service logging — When the provider calls `ExportCertificate`, that call is recorded in AWS CloudTrail with a user agent string containing `aws-workload-credentials-provider`. The provider's internal operations (scheduler cycles, file writes) appear only in the local log.

You can configure logging with the `log_level` and `log_to_file` settings. For more information, see the section called “Configuration reference”.

Certificate management

You can use the ACM console or the AWS CLI to manage the certificates in your account.

- Search certificates to find certificates managed by ACM. You can filter and sort results using advanced search criteria.
- View certificate details to see the details of an individual certificate.
- Delete certificates to remove them from your account. Deleted certificates may appear in lists for a short time after they are deleted.

Search certificates managed by AWS Certificate Manager

With the ACM console or AWS CLI, you can search for certificates in your account. The SearchCertificates operation provides advanced filtering capabilities. You can filter certificates by ARN, X.509 attributes, and ACM specific properties like certificate status, type and renewal eligibility. You can also combine filters with AND, OR, and NOT logical operators.

> **Note**
>
> The ListCertificates operation still exists for basic listing with limited filtering options such as key type, key usage, certificate status, and certificate key pair origin. By default, `ListCertificates` does not return Amazon N2 certificates. To include them, use the `RootHierarchyTypes` filter. For more information, see Listing and finding.

Filter categories

You can filter certificates using the following categories:

- **Certificate ARN** – Filter by the Amazon Resource Name of the certificate.
- **X.509 attributes** – Filter by subject common name, subject alternative name, extended key usage, key usage, key algorithm, serial number, expiration date range (NotAfter), or validity start date range (NotBefore). Date range filters are inclusive of both the start and end dates. For example, to find all certificates for a specific domain, use the common name and DNS filters together with an OR operator.

- **ACM metadata** – Filter by status, type, in-use, exported, export option, managed-by, validation method, renewal status, renewal eligibility, certificate key pair origin, root hierarchy type, ACME endpoint, or ACME account.

The certificate key pair origin filter accepts the values `AWS_MANAGED`, `CUSTOMER_PROVIDED`, and `ACME`. Certificates issued through ACME are returned only when you set the certificate key pair origin filter to `ACME`.

String filters support the CONTAINS and EQUALS comparison operators. You can combine multiple filters using AND, OR, and NOT logical operators.

Sorting

You can sort results by the following fields:

- `CREATED_AT`
- `NOT_AFTER`
- `STATUS`
- `RENEWAL_STATUS`
- `EXPORTED`
- `IN_USE`
- `NOT_BEFORE`
- `KEY_ALGORITHM`
- `TYPE`
- `CERTIFICATE_ARN`
- `COMMON_NAME`
- `REVOKED_AT`
- `RENEWAL_ELIGIBILITY`
- `ISSUED_AT`
- `MANAGED_BY`
- `EXPORT_OPTION`
- `VALIDATION_METHOD`
- `IMPORTED_AT`

Sort order can be `ASCENDING` or `DESCENDING`.

Pagination

Results are paginated. The `MaxResults` parameter defaults to 100 and accepts a maximum value of 500. Use the `NextToken` parameter to retrieve additional pages of results.

To search your certificates using the console

1. Open the ACM console at https://console.aws.amazon.com/acm/.
2. Use the property filter bar to search and filter certificates. You can filter by properties such as common name, status, type, key algorithm, and more. Combine filters with AND or OR operators to narrow your results.
3. Review the information in the certificate list. You can navigate through multiple pages of certificates by using the page numbers. Each certificate occupies a row. The following columns are displayed by default:

- **Domain name** – The fully qualified domain name (FQDN) for the certificate.
- **Type** – The type of certificate. Possible values are: **Amazon issued | Private | Imported**
- **Status** – Certificate status. Possible values are: **Pending validation | Issued | Inactive | Expired | Revoked | Failed | Validation timed out**
- **In use?** – Whether the ACM certificate is actively associated with an AWS service such as Elastic Load Balancing or CloudFront. The value can be **No** or **Yes**.
- **Renewal eligibility** – Whether the certificate can be renewed automatically by ACM when it approaches expiration. Possible values are: **Eligible | Ineligible**. For eligibility rules, see Managed certificate renewal in AWS Certificate Manager.

To customize the certificate list display, choose the settings icon. You can change the number of certificates shown on a page, specify the line-wrapping behavior of cell contents, and display additional information fields. The following optional fields are available:

- **Additional domain names** – One or more domain names (subject alternative names) included in the certificate.
- **Requested at** – The time when ACM requested the certificate.
- **Issued at** – The time when the certificate was issued. This information is available only for Amazon-issued certificates, not for imports.

- **Not before** – The time before which the certificate is not valid.
- **Not after** – The time after which the certificate is not valid.
- **Revoked at** – For revoked certificates, the time of the revocation.
- **Name tag** – The value of a tag on this certificate called *Name*, if such a tag exists.
- **Renewal status** – Status of the requested renewal of a certificate. This field is displayed and has a value only when renewal was requested. Possible values are: **Pending automatic renewal | Pending validation | Success | Failure**.

> **Note**
>
> It can take up to several hours for changes to the certificate status to become available. If a problem is encountered, a certificate request times out after 72 hours, and the issuance or renewal process must be repeated from the beginning.

The **Page size** preference specifies the number of certificates returned on each console page.

For more information about the available certificate details, see View AWS Certificate Manager certificate details.

To search your certificates using the AWS CLI

- Use the search-certificates command to search for ACM-managed certificates. The following example filters for certificates with a status of `ISSUED`:

```
$ aws acm search-certificates \
    --filter-statement '{"Filter": {"AcmCertificateMetadataFilter": {"Status":
 "ISSUED"}}}' \
    --max-results 10
```

The command returns information similar to the following:

```
{
    "Results": [
        {
            "CertificateArn":
 "arn:aws:acm:Region:444455556666:certificate/certificate_ID",
            "X509Attributes": {
```

```
                "Issuer": {
                    "CommonName": "Example CA",
                    "Country": "US",
                    "Organization": "Example Corp"
                },
                "Subject": {
                    "CommonName": "example.com"
                },
                "SubjectAlternativeNames": [
                    {
                        "DnsName": "example.com"
                    },
                    {
                        "DnsName": "www.example.com"
                    }
                ],
                "ExtendedKeyUsages": [
                    "TLS_WEB_SERVER_AUTHENTICATION",
                    "TLS_WEB_CLIENT_AUTHENTICATION"
                ],
                "KeyAlgorithm": "RSA_2048",
                "KeyUsages": [
                    "DIGITAL_SIGNATURE",
                    "KEY_ENCIPHERMENT"
                ],
                "SerialNumber": "serial_number",
                "NotAfter": "2025-02-14T23:59:59+00:00",
                "NotBefore": "2024-01-15T00:00:00+00:00"
            },
            "CertificateMetadata": {
                "AcmCertificateMetadata": {
                    "CreatedAt": "2024-01-15T12:00:00+00:00",
                    "IssuedAt": "2024-01-15T12:05:00+00:00",
                    "Exported": false,
                    "InUse": true,
                    "RenewalEligibility": "ELIGIBLE",
                    "Status": "ISSUED",
                    "Type": "AMAZON_ISSUED",
                    "ValidationMethod": "DNS",
                    "CertificateKeyPairOrigin": "AWS_MANAGED"
                }
            }
        }
    ],
```

```
    "NextToken": "nextToken"
}
```

View AWS Certificate Manager certificate details

You can use the ACM console or the AWS CLI to list detailed metadata about your certificates.

To view certificate details in the console

1. Open the ACM console at https://console.aws.amazon.com/acm/ to display your certificates. You can navigate through multiple pages of certificates using the page numbers at upper-right.
2. To show detailed metadata for a listed certificate, choose the Certificate ID. A page opens, displaying the following information:

   - **Certificate status**
     - **Identifier** – 32-byte hexadecimal unique identifier of the certificate
     - **ARN** – An Amazon Resource Name (ARN) in the form `arn:aws:acm:Region:444455556666:certificate/certificate_ID`
     - **Type** – Identifies the management category of an ACM certificate. Possible values are: **Amazon Issued | Private | Imported**. For more information, see AWS Certificate Manager public certificates, Request a private certificate in AWS Certificate Manager, or Import certificates into AWS Certificate Manager.
     - **Status** – The certificate status. Possible values are: **Pending validation | Issued | Inactive | Expired | Revoked | Failed | Validation timed out**
     - **Detailed status** – Date and time when the certificate was issued or imported
   - **Domains**
     - **Domain** – The fully qualified domain name (FQDN) for the certificate.
     - **Status** – The domain validation status. Possible values are: **Pending validation | Revoked | Failed | Validation timed out | Success**
   - **Details**
     - **In use?** – Whether the certificate is associated with an AWS integrated service Possible values are: **Yes | No**
     - **Domain name** – The first fully qualified domain name (FQDN) for the certificate.
     - **Managed by** – Identifies the AWS service that manages the certificate with ACM.

- **Number of additional names** – Number of domain names for which the certificate is valid
- **Serial number** – 16-byte hexadecimal serial number of the certificate
- **Public key info** – The cryptographic algorithm that generated the key pair
- **Signature algorithm** – The cryptographic algorithm used to sign the certificate.
- **Can be used with** – A list of ACM integrated services that support a certificate with these parameters
- **Requested at** – Date and time of issuance request
- **Issued at** – If applicable, the date and time of issuance
- **Imported at** – If applicable, the date and time of import
- **Not before** – The start of the validity period of the certificate
- **Not after** – The expiration date and time of the certificate
- **Renewal eligibility** – Possible values are: **Eligible | Ineligible**. For eligibility rules, see Managed certificate renewal in AWS Certificate Manager.
- **Renewal status** – Status of the requested renewal of a certificate. This field is displayed and has a value only when renewal was requested. Possible values are: **Pending automatic renewal | Pending validation | Success | Failure**.

  > **Note**
  >
  > It can take up to several hours for changes to the certificate status to become available. If a problem is encountered, a certificate request times out after 72 hours, and the issuance or renewal process must be repeated from the beginning.

- **CA** – The ARN of the signing CA

- **Tags**
  - **Key**
  - **Value**
- **Validation state** – If applicable, possible values are:
  - **Pending** – Validation has been requested and has not completed.
  - **Validation timed out** – A requested validation timed out, but you can repeat the request.
  - **None** – The certificate is for a private PKI or is self-signed, and does not need validation.

To view certificate details using the AWS CLI

Use the describe-certificate in the AWS CLI to display certificate details, as shown in the following command:

```
$ aws acm describe-certificate --certificate-arn
 arn:aws:acm:Region:444455556666:certificate/certificate_ID
```

The command returns information similar to the following:

```
{
    "Certificate": {
        "CertificateArn": "arn:aws:acm:Region:444455556666:certificate/certificate_ID",
        "Status": "EXPIRED",
        "Options": {
            "CertificateTransparencyLoggingPreference": "ENABLED"
        },
        "SubjectAlternativeNames": [
            "example.com",
            "www.example.com"
        ],
        "DomainName": "gregpe.com",
        "NotBefore": 1450137600.0,
        "RenewalEligibility": "INELIGIBLE",
        "NotAfter": 1484481600.0,
        "KeyAlgorithm": "RSA-2048",
        "InUseBy": [
            "arn:aws:cloudfront::account:distribution/E12KXPQHVLSYVC"
        ],
        "SignatureAlgorithm": "SHA256WITHRSA",
        "CreatedAt": 1450212224.0,
        "IssuedAt": 1450212292.0,
        "KeyUsages": [
            {
                "Name": "DIGITAL_SIGNATURE"
            },
            {
                "Name": "KEY_ENCIPHERMENT"
            }
        ],
        "Serial": "07:71:71:f4:6b:e7:bf:63:87:e6:ad:3c:b2:0f:d0:5b",
        "Issuer": "Amazon",
        "Type": "AMAZON_ISSUED",
        "ExtendedKeyUsages": [
            {
```

```
                "OID": "1.3.6.1.5.5.7.3.1",
                "Name": "TLS_WEB_SERVER_AUTHENTICATION"
            },
            {
                "OID": "1.3.6.1.5.5.7.3.2",
                "Name": "TLS_WEB_CLIENT_AUTHENTICATION"
            }
        ],
        "DomainValidationOptions": [
            {
                "ValidationEmails": [
                    "hostmaster@example.com",
                    "admin@example.com",
                    "postmaster@example.com",
                    "webmaster@example.com",
                    "administrator@example.com"
                ],
                "ValidationDomain": "example.com",
                "DomainName": "example.com"
            },
            {
                "ValidationEmails": [
                    "hostmaster@example.com",
                    "admin@example.com",
                    "postmaster@example.com",
                    "webmaster@example.com",
                    "administrator@example.com"
                ],
                "ValidationDomain": "www.example.com",
                "DomainName": "www.example.com"
            }
        ],
        "Subject": "CN=example.com",
        "CertificateKeyPairOrigin": "AWS_MANAGED"
    }
}
```


Note

For certificates issued through ACME, `CertificateKeyPairOrigin` is `ACME`, and the output also includes `AcmeEndpointArn` and `AcmeAccountId`.

Delete certificates managed by AWS Certificate Manager

You can use the ACM console or the AWS CLI to delete a certificate. Deleting a ticket is eventually consistent. A certificate may appear in lists for a short time after it's deleted.

Important

- You cannot delete an ACM certificate that is being used by another AWS service. To delete a certificate that is in use, you must first remove the certificate association. This is done using the console or CLI *for the associated service*.
- You can delete only *expired* certificates issued through ACME (certificates with a `CertificateKeyPairOrigin` of `ACME`). An ACME certificate remains in your account until one year after it expires, when ACM automatically removes it.
- Deleting a certificate issued by a private certificate authority (CA) has no effect on the CA. You will continue to be charged for the CA until it is deleted. For more information, see Deleting Your Private CA in the *AWS Private Certificate Authority User Guide*.

To delete a certificate using the console

1. Open the ACM console at https://console.aws.amazon.com/acm/.
2. In the list of certificates, select the check box for an ACM certificate, then choose **Delete**.

Note

Depending on how you have ordered the list, a certificate you are looking for might not be immediately visible. You can click the black triangle at right to change the ordering. You can also navigate through multiple pages of certificates using the page numbers at upper-right.

To delete a certificate using the AWS CLI

Use the delete-certificate command to delete a certificate, as shown in the following command:

```
$ aws acm delete-certificate --certificate-arn
 arn:aws:acm:Region:444455556666:certificate/certificate_ID
```

Tag AWS Certificate Manager resources

A *tag* is a label that you can assign to an ACM certificate. Each tag consists of a *key* and a *value*. You can use the AWS Certificate Manager console, AWS Command Line Interface (AWS CLI), or ACM API to add, view, or remove tags for ACM certificates. You can choose which tags to display in the ACM console.

You can create custom tags that suit your needs. For example, you could tag multiple ACM certificates with an `Environment = Prod` or `Environment = Beta` tag to identify which environment each ACM certificate is intended for. The following list includes a few additional examples of other custom tags:

- `Admin = Alice`
- `Purpose = Website`
- `Protocol = TLS`
- `Registrar = Route53`

Other AWS resources also support tagging. You can, therefore, assign the same tag to different resources to indicate whether those resources are related. For example, you can assign a tag such as `Website = example.com` to the ACM certificate, the load balancer, and other resources used for your example.com website.

Topics

- Tag restrictions
- Managing tags

Tag restrictions

The following basic restrictions apply to ACM certificate tags:

- The maximum number of tags per ACM certificate is 50.
- The maximum length of a tag key is 127 characters.
- The maximum length of a tag value is 255 characters.
- Tag keys and values are case sensitive.

- The `aws:` prefix is reserved for AWS use; you cannot add, edit, or delete tags whose key begins with `aws:`. Tags that begin with `aws:` do not count against your tags-per-resource quota.
- If you plan to use your tagging schema across multiple services and resources, remember that other services may have other restrictions for allowed characters. Refer to the documentation for that service.
- ACM certificate tags are not available for use in the AWS Management Console's Resource Groups and Tag Editor.

For general information about AWS tagging conventions, see Tagging AWS Resources.

Managing tags

You can add, edit, and delete tags by using the AWS Management Console, the AWS Command Line Interface, or the AWS Certificate Manager API.

Managing tags (console)

You can use the AWS Management Console to add, delete, or edit tags. You can also display tags in columns.

Adding a tag

Use the following procedure to add tags by using the ACM console.

To add a tag to a certificate (console)

1. Sign into the AWS Management Console and open the AWS Certificate Manager console at https://console.aws.amazon.com/acm/home.
2. Choose the arrow next to the certificate that you want to tag.
3. In the details pane, scroll down to **Tags**.
4. Choose **Edit** and **Add Tag**.
5. Type a key and a value for the tag.
6. Choose **Save**.

Deleting a tag

Use the following procedure to delete tags by using the ACM console.

To delete a tag (console)

1. Sign into the AWS Management Console and open the AWS Certificate Manager console at https://console.aws.amazon.com/acm/home.
2. Choose the arrow next to the certificate with a tag that you want to delete.
3. In the details pane, scroll down to **Tags**.
4. Choose **Edit**.
5. Choose the **X** next to the tag you want to delete.
6. Choose **Save**.

Editing a tag

Use the following procedure to edit tags by using the ACM console.

To edit a tag (console)

1. Sign into the AWS Management Console and open the AWS Certificate Manager console at https://console.aws.amazon.com/acm/home.
2. Choose the arrow next to certificate you want to edit.
3. In the details pane, scroll down to **Tags**.
4. Choose **Edit**.
5. Modify the key or value of the tag you want to change.
6. Choose **Save**.

Showing tags in columns

Use the following procedure to show tags in columns in the ACM console.

To display tags in columns (console)

1. Sign into the AWS Management Console and open the AWS Certificate Manager console at https://console.aws.amazon.com/acm/home.
2. Choose the tags that you want to display as columns by choosing the gear icon

   in the upper right corner of the console.

3. Select the check box beside the tag that you want to display in a column.

Managing tags (CLI)

Refer to the following topics to learn how to add, list, and delete tags by using the AWS CLI.

- add-tags-to-certificate
- list-tags-for-certificate
- remove-tags-from-certificate

Managing tags (ACM API)

Refer to the following topics to learn how to add, list, and delete tags by using the API.

- AddTagsToCertificate
- ListTagsForCertificate
- RemoveTagsFromCertificate

> **Note**
>
> The certificate-specific tagging APIs above (`AddTagsToCertificate`, `ListTagsForCertificate`, `RemoveTagsFromCertificate`) apply only to `certificate` resource types. For all other ACM resource types (such as ACME endpoints, ACME external account bindings, and ACME domain validations), use the following APIs:

- TagResource
- ListTagsForResource
- UntagResource

Note

Calling `TagResource`, `UntagResource`, or `ListTagsForResource` with a `certificate`-typed ARN returns a `ValidationException`. Use the certificate-specific APIs for certificate resources.

Security in AWS Certificate Manager

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS Certificate Manager, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS Certificate Manager (ACM). The following topics show you how to configure ACM to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your ACM resources.

Topics

- Data protection in AWS Certificate Manager
- Identity and Access Management for AWS Certificate Manager
- Resilience in AWS Certificate Manager
- Infrastructure security in AWS Certificate Manager

Data protection in AWS Certificate Manager

The AWS shared responsibility model applies to data protection in AWS Certificate Manager. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for

the AWS services that you use. For more information about data privacy, see Data Privacy FAQ. For information about data protection in Europe, see the General Data Protection Regulation (GDPR) Center.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with ACM or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Security for certificate private keys

When you request a public certificate, AWS Certificate Manager (ACM) generates a public/private key pair. For imported certificates, you generate the key pair. The public key becomes part of the certificate. ACM stores the certificate and its corresponding private key, and uses AWS Key Management Service (AWS KMS) to help protect the private key. The process works like this:

1. The first time you request or import a certificate in an AWS Region, ACM creates a managed AWS KMS key with the alias **aws/acm**. This KMS key is unique in each AWS account and each AWS Region.
2. ACM uses this KMS key to encrypt the certificate's private key. ACM stores only an encrypted version of the private key; ACM does not store the private key in plaintext form. ACM uses the same KMS key to encrypt the private keys for all certificates in a specific AWS account and a specific AWS Region.
3. When you associate the certificate with a service that is integrated with AWS Certificate Manager, ACM sends the certificate and the encrypted private key to the service. A grant is also created in AWS KMS that allows the service to use the KMS key to decrypt the certificate's private key. For more information about grants, see Using Grants in the *AWS Key Management Service Developer Guide*. For more information about services supported by ACM, see Managed automation with integrated services.

   > **Note**
   >
   > You have control over the automatically created AWS KMS grant. If you delete this grant for any reason, you lose ACM functionality for the integrated service.

4. Integrated services use the KMS key to decrypt the private key. Then the service uses the certificate and the decrypted (plaintext) private key to establish secure communication channels (SSL/TLS sessions) with its clients.
5. When the certificate is disassociated from an integrated service, the grant created in step 3 is retired. This means the service can no longer use the KMS key to decrypt the certificate's private key.

Identity and Access Management for AWS Certificate Manager

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use ACM resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities

- Managing access using policies
- How AWS Certificate Manager works with IAM
- Identity-based policy examples for AWS Certificate Manager
- ACM API permissions: Actions and resources reference
- AWS managed policies for AWS Certificate Manager
- Use condition keys with ACM
- Condition keys for ACME certificate requests
- Use a service-linked role (SLR) with ACM
- IAM for ACME certificate automation
- Troubleshooting AWS Certificate Manager identity and access

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting AWS Certificate Manager identity and access)
- **Service administrator** - determine user access and submit permission requests (see How AWS Certificate Manager works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for AWS Certificate Manager)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS Certificate Manager works with IAM

Before you use IAM to manage access to ACM, learn what IAM features are available to use with ACM.

IAM features you can use with AWS Certificate Manager

IAM feature	ACM support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No

IAM feature	ACM support
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level view of how ACM and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for ACM

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for ACM

To view examples of ACM identity-based policies, see Identity-based policy examples for AWS Certificate Manager.

Resource-based policies within ACM

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific

resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for ACM

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of ACM actions, see Actions defined by AWS Certificate Manager in the *Service Authorization Reference*.

Policy actions in ACM use the following prefix before the action:

```
acm
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "acm:action1",
      "acm:action2"
         ]
```

To view examples of ACM identity-based policies, see Identity-based policy examples for AWS Certificate Manager.

Policy resources for ACM

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of ACM resource types and their ARNs, see Resources defined by AWS Certificate Manager in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS Certificate Manager.

To view examples of ACM identity-based policies, see Identity-based policy examples for AWS Certificate Manager.

Policy condition keys for ACM

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of ACM condition keys, see Condition keys for AWS Certificate Manager in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Certificate Manager.

To view examples of ACM identity-based policies, see Identity-based policy examples for AWS Certificate Manager.

ACLs in ACM

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with ACM

Supports ABAC (tags in policies): Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with ACM

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Cross-service principal permissions for ACM

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for ACM

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break ACM functionality. Edit service roles only when ACM provides guidance to do so.

Service-linked roles for ACM

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for AWS Certificate Manager

By default, users and roles don't have permission to create or modify ACM resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by ACM, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Certificate Manager in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the ACM console
- Allow users to view their own permissions
- Listing certificates
- Request a certificate
- Retrieving a certificate
- Importing a certificate
- Deleting a certificate

Policy best practices

Identity-based policies determine whether someone can create, access, or delete ACM resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides

more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the ACM console

To access the AWS Certificate Manager console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the ACM resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the ACM console, also attach the ACM `AWSCertificateManagerReadOnly` AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

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

```
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Listing certificates

The following policy allows a user to list all of the ACM certificates in the user's account.

JSON

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

> **Note**
>
> This permission is required for ACM certificates to appear in the Elastic Load Balancing and CloudFront consoles.

Request a certificate

The following policy denies a user from requesting ACM exportable public certificates.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyACMCertificateRequest",
      "Effect": "Deny",
      "Action": [
        "acm:RequestCertificate"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "acm:Export": "ENABLED"
        }
      }
    }
  ]
}
```

Retrieving a certificate

The following policy allows a user to retrieve a specific ACM certificate.

JSON

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

Importing a certificate

The following policy allows a user to import a certificate.

JSON

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

Deleting a certificate

The following policy allows a user to delete a specific ACM certificate.

JSON

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

```
        "Action": "acm:DeleteCertificate",
        "Resource": "arn:aws:acm:us-
east-1:123456789012:certificate/certificate_ID"
    }
}
```

ACM API permissions: Actions and resources reference

When you set up access control and write permissions policies that you can attach to an IAM user or role, you can use the following table as a reference. The first column in the table lists each AWS Certificate Manager API operation. You specify actions in a policy's `Action` element. The remaining columns provide the additional information:

You can use the IAM policy elements in your ACM policies to express conditions. For a complete list, see Available Keys in the *IAM User Guide*.

> **Note**
>
> To specify an action, use the `acm:` prefix followed by the API operation name (for example, `acm:RequestCertificate`).

ACM API operations and permissions

ACM API Operations	Required Permissions (API Operations)	Resources
AddTagsToCertificate	acm:AddTagsToCertificate	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*
DeleteCertificate	acm:DeleteCertificate	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*
DescribeCertificate	acm:DescribeCertificate	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*

ACM API Operations	Required Permissions (API Operations)	Resources
ExportCertificate	acm:ExportCertificate	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*
GetAccountConfiguration	acm:GetAccountConfiguration	*
GetCertificate	acm:GetCertificate	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*
ImportCertificate	acm:ImportCertificate	arn:aws:acm: *region*:*account*:certificate/* or *
ListCertificates	acm:ListCertificates	*
ListTagsForCertificate	acm:ListTagsForCertificate	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*
ListTagsForResource	acm:ListTagsForResource	arn:aws:acm: *region*:*account*:*resource-type* /*resource_ID*
PutAccountConfiguration	acm:PutAccountConfiguration	*
RemoveTagsFromCertificate	acm:RemoveTagsFromCertificate	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*

ACM API Operations	Required Permissions (API Operations)	Resources
RequestCertificate	acm:RequestCertificate	arn:aws:acm: *region*:*account*:certificate/* or *
ResendValidationEmail	acm:ResendValidationEmail	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*
SearchCertificates	acm:SearchCertificates	*
TagResource	acm:TagResource	arn:aws:acm: *region*:*account*:*resource-type* /*resource_ID*
UntagResource	acm:UntagResource	arn:aws:acm: *region*:*account*:*resource-type* /*resource_ID*
UpdateCertificateOptions	acm:UpdateCertificateOptions	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*

AWS managed policies for AWS Certificate Manager

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWSCertificateManagerReadOnly

This policy provides read–only access to ACM certificates; it allows users to describe, list, search, and retrieve ACM certificates. The policy also includes describe and list permissions for ACME certificate automation resources.

To view this AWS managed policy in the console, go to https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AWSCertificateManagerReadOnly.

For a JSON listing of the policy details, see AWSCertificateManagerReadOnly.

AWSCertificateManagerFullAccess

This policy provides full access to all ACM actions and resources.

To view this AWS managed policy in the console, go to https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AWSCertificateManagerFullAccess.

For a JSON listing of the policy details, see AWSCertificateManagerFullAccess.

ACM updates to AWS managed policies

View details about updates to AWS managed policies for ACM since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the ACM Document history page.

Change	Description	Date
Added ACME read action support to the AWSCertificateManagerReadOnly policy.	The AWSCertificateManagerReadOnly policy now includes permission to call the DescribeAcmeAccount, DescribeAcmeDomainValidation, DescribeAcmeEndpoint, DescribeAcmeExternalAccountBinding, ListAcmeAccounts, ListAcmeDomainValidations, ListAcmeEndpoints, ListAcmeExternalAccountBindings, and ListTagsForResource API actions.	June 30, 2026
Added SearchCertificates support to the AWSCertificateManagerReadOnly policy.	The AWSCertificateManagerReadOnly policy now includes permission to call the SearchCertificates API action.	March 31, 2026
Added GetAccountConfiguration support to the AWSCertificateManagerReadOnly policy.	The AWSCertificateManagerReadOnly policy now includes permission to call the GetAccountConfiguration API action.	March 3, 2021
ACM starts tracking changes	ACM starts tracking changes for AWS managed policies.	March 3, 2021

Use condition keys with ACM

AWS Certificate Manager uses AWS Identity and Access Management (IAM) condition keys to limit access to certificate requests. With condition keys from IAM policies or Service Control Policies (SCP) you can create certificate requests that conform to your organization's guidelines.

> **Note**
>
> Combine ACM condition keys with AWS global condition keys such as `aws:PrincipalArn` to further restrict actions to specific users or roles.

Certificates requested through an ACME endpoint are authorized as standard ACM actions, so these condition keys apply to them as well. The keys available on each action and the values ACM supplies differ from a direct API call. For those values, see Condition keys for ACME certificate requests.

Supported conditions for ACM

ACM API operations and supported conditions

Condition Key	Supported ACM API Operations	Type	Description
`acm:ValidationMethod`	RequestCertificate	String (`DNS`, `EMAIL`, `HTTP`)	Filter requests based on ACM validation method
`acm:DomainNames`	RequestCertificate	ArrayOfString	Filter based on domain names in the ACM request
`acm:KeyAlgorithm`	RequestCertificate	String	Filter requests based on ACM key algorithm and size
`acm:CertificateTra`	RequestCertificate	String (`ENABLED`, `DISABLED`)	Deprecated. This condition will continue to be

Condition Key	Supported ACM API Operations	Type	Description
`nsparency Logging`			enforced but certificate transparency logging is always enabled for public certificates and cannot be disabled.
`acm:CertificateAuthority`	RequestCertificate	ARN	Filter requests based on certificate authorities in the ACM request
`acm:Export`	RequestCertificate	String (`ENABLED`, `DISABLED`)	Filter requests based on the export option in the request. Use this condition to restrict the creation of exportable public certificates.

Condition Key	Supported ACM API Operations	Type	Description
acm:CertificateKeyPairOrigin	RequestCertificate, AddTagsToCertificate, RevokeCertificate	String (AWS_MANAGED , ACME, CUSTOMER_PROVIDED)	Filter requests based on the certificate's key pair origin. For RequestCertificate , the value is always AWS_MANAGED . When requesting certificates through an ACME client, the value is ACME. For AddTagsToCertificate and RevokeCertificate , the value is read from the existing certificate.

Example 1: Restricting validation method

The following policy denies new certificate requests using the Email Validation method except for a request made using the arn:aws:iam::123456789012:role/AllowedEmailValidation role.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":{
        "Effect":"Deny",
        "Action":"acm:RequestCertificate",
        "Resource":"*",
        "Condition":{
            "StringLike" : {
```

```
                "acm:ValidationMethod":"EMAIL"
            },
            "ArnNotLike": {
                "aws:PrincipalArn": [ "arn:aws:iam::123456789012:role/
AllowedEmailValidation"]
            }
        }
    }
}
```

Example 2: Preventing wildcard domains

The following policy denies any new ACM certificate request that uses wildcard domains.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":{
        "Effect":"Deny",
        "Action":"acm:RequestCertificate",
        "Resource":"*",
        "Condition": {
            "ForAnyValue:StringLike": {
                "acm:DomainNames": [
                    "${*}.*"
                ]
            }
        }
    }
}
```

Example 3: Restricting certificate domains

The following policy denies any new ACM certificate request for domains that don't end with `*.amazonaws.com`

JSON

```
{
    "Version":"2012-10-17",
    "Statement":{
        "Effect":"Deny",
        "Action":"acm:RequestCertificate",
        "Resource":"*",
        "Condition": {
            "ForAnyValue:StringNotLike": {
                "acm:DomainNames": ["*.amazonaws.com"]
            }
        }
    }
}
```

The policy could be further restricted to specific subdomains. This policy would only allow requests where every domain matches at least one of the conditional domain names.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":{
        "Effect":"Deny",
        "Action":"acm:RequestCertificate",
        "Resource":"*",
        "Condition": {
            "ForAllValues:StringNotLike": {
                "acm:DomainNames": ["support.amazonaws.com",
 "developer.amazonaws.com"]
            }
        }
    }
}
```

Example 4: Restricting key algorithm

The following policy uses the condition key `StringNotLike` to allow only certificates requested with the ECDSA 384 bit (`EC_secp384r1`) key algorithm.

JSON

```
{
    "Version":"2012-10-17",
        "Statement":{
        "Effect":"Deny",
        "Action":"acm:RequestCertificate",
        "Resource":"*",
        "Condition":{
            "StringNotLike" : {
                "acm:KeyAlgorithm":"EC_secp384r1"
            }
        }
    }
}
```

The following policy uses the condition key `StringLike` and wildcard * matching to prevent requests for new certificates in ACM with any RSA key algorithm.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":{
        "Effect":"Deny",
        "Action":"acm:RequestCertificate",
        "Resource":"*",
        "Condition":{
            "StringLike" : {
                "acm:KeyAlgorithm":"RSA*"
            }
        }
```

```
    }
}
```

Example 5: Restricting certificate authority

The following policy would only allow requests for private certificates using the provided Private Certificate Authority (PCA) ARN.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":{
        "Effect":"Deny",
        "Action":"acm:RequestCertificate",
        "Resource":"*",
        "Condition":{
            "StringNotLike": {
                "acm:CertificateAuthority":" arn:aws:acm-
pca:region:account:certificate-authority/CA_ID"
            }
        }
    }
}
```

This policy uses the `acm:CertificateAuthority` condition to allow only requests for publicly trusted certificates issued by Amazon Trust Services. Setting the Certificate Authority ARN to `false` prevents requests for private certificates from PCA.

JSON

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

```
        "Action":"acm:RequestCertificate",
        "Resource":"*",
        "Condition":{
            "Null" : {
                "acm:CertificateAuthority":"false"
            }
        }
    }
}
```

Example 6: Restricting actions by certificate key pair origin

The following policy allows tagging only on ACME certificates. This is useful when you want to restrict tagging operations to certificates issued through the ACME protocol.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "acm:AddTagsToCertificate",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "acm:CertificateKeyPairOrigin": "ACME"
        }
      }
    }
  ]
}
```

Condition keys for ACME certificate requests

Certificate issuance and revocation through an ACME endpoint are authorized as standard ACM actions. This authorization uses the IAM role attached to the ACME client's external account binding. The ACM condition keys therefore apply to ACME requests as well, and you can use them in the role's permissions policy, in an IAM policy, in a Service Control Policy (SCP), or in a VPC endpoint policy.

For the full list of keys and the operations that support them, see Use condition keys with ACM. For more information about the role, see IAM for ACME certificate automation.

What differs is the set of keys available on each action, and the values ACM supplies. An ACME client sends a certificate signing request (CSR) rather than an ACM API request. As a result, ACM derives some values from the CSR, sets others to a fixed value, and leaves some keys unpopulated. Write conditions against the keys listed for the action you are restricting: a condition on a key that an action does not carry denies that action outright.

Condition keys for ACME issuance

When an ACME client finalizes an order, ACM authorizes the `acm:RequestCertificate` action against the certificates in your account and populates the following condition keys.

Condition key values for ACME issuance

Condition key	Value
`acm:CertificateKeyPairOrigin`	Always `ACME`. Use this key to write policies that apply only to ACME issuance, or only to issuance that does not use ACME.
`acm:DomainNames`	The domain names the certificate is requested for, taken from the subject alternative name (SAN) entries of the CSR.
`acm:KeyAlgorithm`	The key algorithm of the CSR's public key: `RSA_2048`, `EC_prime256v1` , or `EC_secp384r1` .
`acm:ValidationMethod`	Always `DNS`. ACME issuance uses the domain validations configured on the endpoint, which are DNS validations.
`acm:CertificateTransparencyLogging`	Always `ENABLED`. Certificate transparency logging cannot be disabled for public certificates.
`acm:Export`	Always `ENABLED`. The ACME client generates and holds the private key, so an ACME certifica

Condition key	Value
	te is exported by definition. A policy that denies issuance when `acm:Export` is `ENABLED` therefore denies all ACME issuance.

> **Note**
>
> ACM does not set `acm:CertificateAuthority` for ACME issuance. To restrict which certificate authority an ACME endpoint issues from, configure the endpoint itself. For more information, see Endpoint configuration.

Condition keys for ACME revocation

Revocation through the ACME endpoint's revoke-cert URL is authorized as the `acm:RevokeCertificate` action. This action is scoped to the ARN of the certificate being revoked.

> **Important**
>
> `acm:CertificateKeyPairOrigin`, set to `ACME`, is the only condition key ACM populates for ACME revocation. In particular, `acm:KeyAlgorithm` is not available. This condition key is read from a certificate request, but an ACME client identifies the certificate to revoke by its bytes, not by a request that carries a key algorithm. A statement that requires `acm:KeyAlgorithm` with a `StringEquals` condition never matches a revocation request, and so denies all ACME revocation.

Condition keys for tagging an ACME certificate

If the ACME endpoint is configured with certificate tags, ACM applies those tags to each certificate it issues from the endpoint. ACM authorizes that as a separate `acm:AddTagsToCertificate` action. This action must also be allowed for issuance to succeed, and it populates the following condition keys.

Condition key values for tagging an ACME certificate

Condition key	Type	Value
`aws:RequestTag/` *tag-key*	String	The value of each certificate tag configured on the endpoint.
`aws:TagKeys`	ArrayOfString	The keys of all certificate tags configured on the endpoint.
`aws:ResourceTag/` *tag-key*	String	The value of each tag being applied to the new certificate.
`acm:CertificateKeyPairOrigin`	String	Always ACME.

Important

If your external account binding role allows only `acm:RequestCertificate`, it cannot issue certificates from an endpoint that carries certificate tags. Grant `acm:AddTagsToCertificate` as well, or remove the tags from the endpoint.

How a denial appears to the ACME client

When a policy condition denies an ACME certificate request, the client receives the ACME error type `urn:ietf:params:acme:error:unauthorized`. The error detail names the denied ACM action. Certbot and other clients report this at the finalize step of the order.

A request that the endpoint's own configuration rejects looks different. For example, suppose the CSR uses a key algorithm that the endpoint's allowed key algorithms do not permit. In this case, the client receives `urn:ietf:params:acme:error:badCSR`, which names the permitted algorithms. That is a configuration result rather than an authorization result, so check the endpoint configuration rather than your policies when you see it.

Example: Restricting ACME issuance and revocation

The following permissions policy for an external account binding role allows ACME issuance only for subdomains of `example.com` and only with an ECDSA P-384 key. It also allows revocation of any ACME certificate in the account.

Issuance and revocation are separate statements because the two actions do not carry the same condition keys. Adding the `acm:KeyAlgorithm` or `acm:DomainNames` condition to the revocation statement would prevent the client from revoking anything.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAcmeIssuance",
      "Effect": "Allow",
      "Action": "acm:RequestCertificate",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "acm:CertificateKeyPairOrigin": "ACME",
          "acm:KeyAlgorithm": "EC_secp384r1"
        },
        "ForAllValues:StringLike": {
          "acm:DomainNames": "*.example.com"
        }
      }
    },
    {
      "Sid": "AllowAcmeRevocation",
      "Effect": "Allow",
      "Action": "acm:RevokeCertificate",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "acm:CertificateKeyPairOrigin": "ACME"
        }
      }
    }
  ]
}
```

If the ACME endpoint is configured with certificate tags, add a statement allowing `acm:AddTagsToCertificate` as well.

Use a service-linked role (SLR) with ACM

AWS Certificate Manager uses an AWS Identity and Access Management (IAM) service-linked role to enable enable automatic renewals of private certificates issued from a private CA for another account shared by AWS Resource Access Manager. A service-linked role (SLR) is an IAM role that is linked directly to the ACM service. SLRs are predefined by ACM and include all the permissions that the service requires to call other AWS services on your behalf.

The SLR makes setting up ACM easier because you don't have to manually add the necessary permissions for unattended certificate signing. ACM defines the permissions of its SLR, and unless defined otherwise, only ACM can assume the role. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

For information about other services that support SLRs, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the SLR documentation for that service.

SLR permissions for ACM

ACM uses an SLR named Amazon Certificate Manager Service Role Policy.

The AWSServiceRoleForCertificateManager SLR trusts the following services to assume the role:

- `acm.amazonaws.com`

The role permissions policy allows ACM to complete the following actions on the specified resources:

- Actions: `acm-pca:IssueCertificate`, `acm-pca:GetCertificate` on "*"

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete an SLR. For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

Important

ACM might alert you that it cannot determine whether an SLR exists on your account. If the required `iam:GetRole` permission has already been granted to the ACM SLR for your account, then the alert will not recur after the SLR is created. If it does recur, then you or your account administrator might need to grant the `iam:GetRole` permission to ACM, or associate your account with the ACM-managed policy `AWSCertificateManagerFullAccess`.

Creating the SLR for ACM

You don't need to manually create the SLR that ACM uses. When you issue an ACM certificate using the AWS Management Console, the AWS CLI, or the AWS API, ACM creates the SLR for you the first time that you a private CA for another account shared by AWS RAM to sign your certificate.

If you encounter messages stating that ACM cannot determine whether an SLR exists on your account, it may mean that your account has not granted a read permission that AWS Private CA requires. This will not prevent the SLR from being installed, and you can still issue certificates, but ACM will be unable to renew the certificates automatically until you resolve the problem. For more information, see Problems with the ACM service-linked role (SLR).

Important

This SLR can appear in your account if you completed an action in another service that uses the features supported by this role. Also, if you were using the ACM service before January 1, 2017, when it began supporting SLRs, then ACM created the AWSServiceRoleForCertificateManager role in your account. To learn more, see A New Role Appeared in My IAM Account.

If you delete this SLR, and then need to create it again, you can use either of these methods:

- In the IAM console, choose **Role**, **Create role**, **Certificate Manager** to create a new role with the **CertificateManagerServiceRolePolicy** use case.
- Using the IAM API CreateServiceLinkedRole or the corresponding AWS CLI command create-service-linked-role, create an SLR with the `acm.amazonaws.com` service name.

For more information, see Creating a Service-Linked Role in the *IAM User Guide*.

Editing the SLR for ACM

ACM does not allow you to edit the AWSServiceRoleForCertificateManager service-linked role. After you create an SLR, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the *IAM User Guide*.

Deleting the SLR for ACM

You typically don't need to delete the AWSServiceRoleForCertificateManager SLR. However, you can delete the role manually using the IAM console, the AWS CLI or the AWS API. For more information, see Deleting a Service-Linked Role in the *IAM User Guide*.

Supported Regions for ACM SLRs

ACM supports using SLRs in all of the regions where both ACM and AWS Private CA are available. For more information, see AWS Regions and Endpoints.

Region name	Region identity	Support in ACM
US East (N. Virginia)	us-east-1	Yes
US East (Ohio)	us-east-2	Yes
US West (N. California)	us-west-1	Yes
US West (Oregon)	us-west-2	Yes
Asia Pacific (Mumbai)	ap-south-1	Yes
Asia Pacific (Osaka)	ap-northeast-3	Yes
Asia Pacific (Seoul)	ap-northeast-2	Yes
Asia Pacific (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes
Asia Pacific (Tokyo)	ap-northeast-1	Yes

Region name	Region identity	Support in ACM
Canada (Central)	ca-central-1	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Zurich)	eu-central-2	Yes
Europe (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes
South America (São Paulo)	sa-east-1	Yes
AWS GovCloud (US-West)	us-gov-west-1	Yes
AWS GovCloud (US-East) East	us-gov-east-1	Yes

IAM for ACME certificate automation

ACME certificate automation uses IAM roles to authorize certificate issuance and revocation. This section describes the permissions model for ACME.

PKI administrator permissions

PKI administrators who create and manage ACME resources need the following permissions:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "acm:CreateAcmeEndpoint",
                "acm:DescribeAcmeEndpoint",
                "acm:ListAcmeEndpoints",
                "acm:UpdateAcmeEndpoint",
                "acm:DeleteAcmeEndpoint",
                "acm:CreateAcmeExternalAccountBinding",
```

```
                "acm:DescribeAcmeExternalAccountBinding",
                "acm:ListAcmeExternalAccountBindings",
                "acm:GetAcmeExternalAccountBindingCredentials",
                "acm:RevokeAcmeExternalAccountBinding",
                "acm:DeleteAcmeExternalAccountBinding",
                "acm:CreateAcmeDomainValidation",
                "acm:DescribeAcmeDomainValidation",
                "acm:ListAcmeDomainValidations",
                "acm:UpdateAcmeDomainValidation",
                "acm:DeleteAcmeDomainValidation"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/AcmeIssuanceRole",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "acm-acme.amazonaws.com"
                }
            }
        }
    ]
}
```

EAB role requirements

Each external account binding is associated with an IAM role. ACM uses this role to authorize certificate issuance and revocation for ACME clients that authenticate with the binding's credentials.

Trust policy

The role must trust the ACME service principal, granting `sts:AssumeRole`, `sts:TagSession`, and `sts:SetSourceIdentity`. The following trust policy also uses a condition on `sts:SourceIdentity` to allow only sessions that ACM establishes for ACME (source identities that begin with `acm-acme-`):

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

```
            "Effect": "Allow",
            "Principal": {
                "Service": "acm-acme.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringLikeIfExists": {
                    "sts:SourceIdentity": "acm-acme-*"
                }
            }
        }
    ]
}
```

Permissions policy

The role needs permissions for the certificate operations you want to allow. The same ACM actions and condition keys that apply to direct API calls apply here:

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

You can restrict issuance using the same condition keys supported by `acm:RequestCertificate`, such as `acm:DomainNames` or `acm:KeyAlgorithm`. For more information, see Use condition keys with ACM.

For the values ACM sets for each of those condition keys during ACME issuance, and for the additional `acm:AddTagsToCertificate` permission that an endpoint configured with certificate tags requires, see Condition keys for ACME certificate requests.

Role session name and source identity

When ACM assumes the role, it sets a role session name and a source identity that appear in CloudTrail logs and that you can reference with the `sts:RoleSessionName` and `sts:SourceIdentity` condition keys:

- **At certificate issuance and revocation** – the role session name is `acme-request-`*`request-id`* and the source identity is `acm-acme-`*`acme-account-id`*.
- **When validating the role at external account binding creation** – the role session name is `acme-verification` and the source identity is `acm-acme-verification`.

Both source identities begin with `acm-acme-`, so the `sts:SourceIdentity` condition in the trust policy allows both. ACM also attaches session tags on the assumed-role session, including `acme-endpoint-arn`, `acme-account-url`, and `acme-operation`.

SCP compatibility

Because the ACME service makes standard ACM API calls using the assumed role, AWS Organizations Service Control Policies (SCPs) are enforced at certificate issuance time. If an SCP denies `acm:RequestCertificate` for the account, ACME certificate issuance also fails. This provides the same governance controls for ACME-issued certificates as for certificates issued directly through the ACM API.

Troubleshooting AWS Certificate Manager identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with ACM and IAM.

Topics

- I am not authorized to perform an action in ACM
- I am not authorized to request a certificate in ACM
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my ACM resources

I am not authorized to perform an action in ACM

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `acm:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 acm:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `acm:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to request a certificate in ACM

If you receive this error, your ACM or PKI administrator has set rules that prevent you from requesting the certificate in its current state.

The following example error occurs when an IAM user tries to use the console to request a certificate using options that are configured with a DENY by the organization administrator.

```
User: arn:aws:sts::account::ID: is not authorized to perform: acm:RequestCertificate
on resource: arn:aws:acm:region:account:certificate/*
with an explicit deny in a service control policy
```

In this case the request should be made again in a way that is inline with the policies set by your administrator. Or the policy needs to be updated to allow requesting the certificate.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to ACM.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in ACM. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my ACM resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether ACM supports these features, see How AWS Certificate Manager works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Resilience in AWS Certificate Manager

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones

without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure security in AWS Certificate Manager

As a managed service, AWS Certificate Manager is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access ACM through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Granting programmative access to ACM

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development

Which user needs programmatic access?	To	By
		in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

Monitor and log AWS Certificate Manager

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Certificate Manager and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs.

The following topics describe AWS cloud-monitoring tools available for use with ACM.

Topics

- Using Amazon EventBridge
- Using CloudTrail with AWS Certificate Manager
- Supported CloudWatch metrics

Using Amazon EventBridge

You can use Amazon EventBridge (formerly CloudWatch Events) to automate your AWS services and respond automatically to system events such as application availability issues or resource changes. Events from AWS services, including ACM, are delivered to Amazon EventBridge in near-real time. You can use events to trigger targets including AWS Lambda functions, AWS Batch jobs, Amazon SNS topics, and many others. For more information, see What Is Amazon EventBridge?

Topics

- Amazon EventBridge support for ACM
- Initiating actions with Amazon EventBridge in ACM

Amazon EventBridge support for ACM

This topic lists and describes the ACM related events supported by Amazon EventBridge.

ACM Certificate Approaching Expiration event

ACM sends daily expiration events for all active certificates (public, private and imported) starting 45 days prior to expiration for private/imported certificates and 30 days prior to expiration for public certificates. This timing can be changed using the PutAccountConfiguration action of the ACM API.

ACM automatically initiates renewal of eligible certificates that it issued, but imported certificates need to be reissued and reimported prior to expiration to avoid outages. For more information, see Reimporting a certificate. You can use expiration events to set up automation to reimport certificates into ACM. For an example of automation using AWS Lambda, see Initiating actions with Amazon EventBridge in ACM.

ACM Certificate Approaching Expiration events have the following structure.

```
{
  "version": "0",
  "id": "id",
  "detail-type": "ACM Certificate Approaching Expiration",
  "source": "aws.acm",
  "account": "account",
  "time": "2020-09-30T06:51:08Z",
  "region": "region",
  "resources": [
    "arn:aws:acm:region:account:certificate/certificate_ID"
  ],
  "detail": {
    "DaysToExpiry": 31,
    "CommonName": "example.com",
    "FirstSubjectAlternativeName": "example.com",
    "ValidityInDays": 90,
    "CertificateKeyPairOrigin": "AWS_MANAGED" | "ACME" | "CUSTOMER_PROVIDED",
    "AcmeAccountId": "acme_account_id",
    "AcmeEndpointArn": "arn:aws:acm:region:account:acme-endpoint/endpoint_ID"
  }
}
```

ACM Certificate Expired event

> **Note**
>
> Certificate Expired events aren't available for imported certificates.

Customers can listen on this event to alert them if an ACM issued public or private certificate in their account expires.

ACM Certificate Expired events have the following structure.

```
{
    "version": "0",
    "id": "id",
    "detail-type": "ACM Certificate Expired",
    "source": "aws.acm",
    "account": "account",
    "time": "2019-12-22T18:43:48Z",
    "region": "region",
    "resources": [
        "arn:aws:acm:region:account:certificate/certificate_ID"
     ],
     "detail": {
        "CertificateType" : "AMAZON_ISSUED" | "PRIVATE",
        "RootHierarchyType" : "N2_AMAZON_MANAGED" | "N2_SFG2_AMAZON_MANAGED",
        "CertificateUsage" : "CLIENT_AND_SERVER_AUTH" | "CLIENT_AUTH_ONLY",
        "CommonName": "example.com",
        "FirstSubjectAlternativeName": "example.com",
        "ValidityInDays": 90,
        "CertificateKeyPairOrigin": "AWS_MANAGED" | "ACME" | "CUSTOMER_PROVIDED",
        "AcmeAccountId": "acme_account_id",
        "AcmeEndpointArn": "arn:aws:acm:region:account:acme-endpoint/endpoint_ID",
        "DomainValidationMethod" : "EMAIL" | "DNS",
        "CertificateCreatedDate" : "2018-12-22T18:43:48Z",
        "CertificateExpirationDate" : "2019-12-22T18:43:48Z",
        "InUse" : TRUE | FALSE,
        "Exported" : TRUE | FALSE
    }
 }
```

`RootHierarchyType` and `CertificateUsage` are present only for Amazon N2 certificates.

ACM Certificate Available event

Customers can listen on this event to be notified when a managed public or private certificate is ready for use. The event is published on issuance, renewal, and import. For a private certificate, once it becomes available, customer action is still required to deploy it to hosts.

ACM Certificate Available events have the following structure.

```
{
    "version": "0",
    "id": "id",
    "detail-type": "ACM Certificate Available",
```

```
    "source": "aws.acm",
    "account": "account",
    "time": "2019-12-22T18:43:48Z",
    "region": "region",
    "resources": [
        "arn:aws:acm:region:account:certificate/certificate_ID"
    ],
    "detail": {
       “Action” : "ISSUANCE" | "RENEWAL" | "IMPORT" | "REIMPORT",
       "CertificateType" : "AMAZON_ISSUED" | "PRIVATE" | "IMPORTED",
       "RootHierarchyType" : "N2_AMAZON_MANAGED" | "N2_SFG2_AMAZON_MANAGED",
       "CertificateUsage" : "CLIENT_AND_SERVER_AUTH" | "CLIENT_AUTH_ONLY",
       "CommonName": "example.com",
       "FirstSubjectAlternativeName": "example.com",
       "ValidityInDays": 90,
       "CertificateKeyPairOrigin": "AWS_MANAGED" | "ACME" | "CUSTOMER_PROVIDED",
       "AcmeAccountId": "acme_account_id",
       "AcmeEndpointArn": "arn:aws:acm:region:account:acme-endpoint/endpoint_ID",
       "DomainValidationMethod" : "EMAIL" | "DNS",
       "CertificateCreatedDate" : "2019-12-22T18:43:48Z",
       "CertificateExpirationDate" : "2019-12-22T18:43:48Z",
       "DaysToExpiry" : 198,
       "InUse" : TRUE | FALSE,
       "Exported" : TRUE | FALSE
     }
}
```

`RootHierarchyType` and `CertificateUsage` are present only for Amazon N2 certificates.

ACM Certificate Renewal Action Required event

> **Note**
>
> Certificate Renewal Action Required events aren't available for imported certificates.

Customers can listen on this event to be alerted when a customer action must be taken before a certificate can be renewed. For instance, if a customer adds CAA records that prevent ACM from renewing a certificate, ACM publishes this event when automatic renewal fails at 45 days before expiration for private certificates and 30 days before expiration for public certificates. If no customer action is taken, ACM makes further renewal attempts at 30 days (for private only), 15

days, 3 days, and 1 day, or until customer action is taken, the certificate expires, or the certificate is no longer eligible for renewal. An event is published for each of these renewal attempts.

ACM Certificate Renewal Action Required events have the following structure.

```
{
   "version": "0",
   "id": "id",
   "detail-type": "ACM Certificate Renewal Action Required",
   "source": "aws.acm",
   "account": "account",
   "time": "2019-12-22T18:43:48Z",
   "region": "region",
   "resources": [
       "arn:aws:acm:region:account:certificate/certificate_ID"
    ],
    "detail": {
       "CertificateType" : "AMAZON_ISSUED" | "PRIVATE",
       "CommonName": "example.com",
       "FirstSubjectAlternativeName": "example.com",
       "ValidityInDays": 90,
       "CertificateKeyPairOrigin": "AWS_MANAGED",
       "DomainValidationMethod" : "EMAIL" | "DNS",
       "RenewalStatusReason" : "CAA_ERROR" | "PENDING_DOMAIN_VALIDATION" |
 "NO_AVAILABLE_CONTACTS" | "ADDITIONAL_VERIFICATION_REQUIRED" | "DOMAIN_NOT_ALLOWED"
 | "INVALID_PUBLIC_DOMAIN" | "DOMAIN_VALIDATION_DENIED" | "PCA_LIMIT_EXCEEDED"
 | "PCA_INVALID_ARN" | "PCA_INVALID_STATE" | "PCA_REQUEST_FAILED" |
 "PCA_NAME_CONSTRAINTS_VALIDATION" | "PCA_RESOURCE_NOT_FOUND" | "PCA_INVALID_ARGS" |
 "PCA_INVALID_DURATION" | "PCA_ACCESS_DENIED" | "SLR_NOT_FOUND" | "OTHER",
       "DaysToExpiry": 30,
       "CertificateExpirationDate" : "2019-12-22T18:43:48Z",
       "InUse" : TRUE | FALSE,
       "Exported" : TRUE | FALSE
   }
}
```

ACM Certificate Revoked event

Customers can listen on this event to alert them if an ACM issued public or private certificate in their account is revoked.

> **Note**
>
> Only exported certificates can be revoked. Imported certificates cannot be revoked via revoke-certificate.

ACM Certificate Revoked events have the following structure.

```
{
  "version": "0",
  "id": "id",
  "detail-type": "ACM Certificate Revoked",
  "source": "aws.acm",
  "account": "account",
  "time": "2019-12-22T18:43:48Z",
  "region": "region",
 "resources": [
        "arn:aws:acm:region:account:certificate/certificate_ID"
     ],
  "detail": {
    "CertificateType" : "AMAZON_ISSUED" | "PRIVATE",
    "CommonName": "example.com",
    "FirstSubjectAlternativeName": "example.com",
    "ValidityInDays": 90,
    "CertificateKeyPairOrigin": "AWS_MANAGED" | "ACME" | "CUSTOMER_PROVIDED",
    "AcmeAccountId": "acme_account_id",
    "AcmeEndpointArn": "arn:aws:acm:region:account:acme-endpoint/endpoint_ID",
    "CertificateExpirationDate" : "2019-12-22T18:43:48Z",
    "Exportable": TRUE | FALSE
  }
}
```

AWS health events

AWS health events are generated for ACM certificates that are eligible for renewal. For information about renewal eligibility, see Managed certificate renewal in AWS Certificate Manager.

Health events are generated in two scenarios:

- On successful renewal of a public or private certificate.

- When a customer must take action for a renewal to occur. This may mean choosing a link in an email message (for email-validated certificates), or resolving an error. One of the following event codes is included with each event. The codes are exposed as variables that you can use for filtering.
  - `AWS_ACM_RENEWAL_STATE_CHANGE` (the certificate has been renewed, has expired, or is due to expire)
  - `CAA_CHECK_FAILURE` (CAA check failed)
  - `AWS_ACM_RENEWAL_FAILURE` (for certificates signed by a private CA)

Health events have the following structure. In this example, an `AWS_ACM_RENEWAL_STATE_CHANGE` event has been generated.

```
{
   "source":[
      "aws.health"
   ],
   "detail-type":[
      "AWS Health Event"
   ],
   "detail":{
      "service":[
         "ACM"
      ],
      "eventTypeCategory":[
         "scheduledChange"
      ],
      "eventTypeCode":[
         "AWS_ACM_RENEWAL_STATE_CHANGE"
      ]
   }
}
```

Initiating actions with Amazon EventBridge in ACM

You can create Amazon EventBridge rules based on these events and use the Amazon EventBridge console to configure actions that take place when the events are detected. This section provides sample procedures for configuring Amazon EventBridge rules and resulting actions.

Topics

- Responding to an event with Amazon SNS
- Responding to an event with a Lambda function

Responding to an event with Amazon SNS

This section shows how to configure Amazon SNS to send a text notification whenever ACM generates a health event.

Complete the following procedure to configure a response.

To create a Amazon EventBridge rule and trigger an action

1. Create an Amazon EventBridge rule. For more information, see Creating Amazon EventBridge rules that react to events.

   a. In the Amazon EventBridge console at https://console.aws.amazon.com/events/, navigate to the **Events** > **Rules** page and choose **Create rule**.

   b. On the **Create rule** page, select **Event Pattern**.

   c. For **Service Name**, choose **Health** from the menu.

   d. For **Event Type**, choose **Specific Health events**.

   e. Select **Specific service(s)** and choose **ACM** from the menu.

   f. Select **Specific event type category(s)** and choose **accountNotification**.

   g. Choose **Any event type code**.

   h. Choose **Any resource**.

   i. In the **Event pattern preview** editor, paste the JSON pattern emitted by the event. This example uses the pattern from the AWS health events section.

```
{
   "source":[
      "aws.health"
   ],
   "detail-type":[
      "AWS Health Event"
   ],
   "detail":{
      "service":[
         "ACM"
```

```
      ],
      "eventTypeCategory":[
         "scheduledChange"
      ],
      "eventTypeCode":[
         "AWS_ACM_RENEWAL_STATE_CHANGE"
      ]
   }
}
```

2. Configure an action.

   In the **Targets** section, you can choose from among many services that can immediately consume your event, such as Amazon Simple Notification Service (SNS), or you can choose **Lambda function** to pass the event to customized executable code. For an example of an AWS Lambda implementation, see Responding to an event with a Lambda function.

Responding to an event with a Lambda function

This procedure demonstrates how to use AWS Lambda to listen on Amazon EventBridge, create notifications with Amazon Simple Notification Service (SNS), and publish findings to AWS Security Hub CSPM, providing visibility to administrators and security teams.

To set up a Lambda function and IAM role

1. First configure an AWS Identity and Access Management (IAM) role and define the permissions needed by the Lambda function. This security best practice gives you flexibility in designating who has authorization to call the function, and in limiting the permissions granted to that person. It is not recommended to run most AWS operations directly under a user account and especially not under an administrator account.

   Open the IAM console at https://console.aws.amazon.com/iam/.

2. Use the JSON policy editor to create the policy defined in the template below. Provide your own Region and AWS account details. For more information, see Creating policies on the JSON tab.

   JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "LambdaCertificateExpiryPolicy1",
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:us-east-1:123456789012:*"
        },
        {
            "Sid": "LambdaCertificateExpiryPolicy2",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:/aws/lambda/
handle-expiring-certificates:*"
            ]
        },
        {
            "Sid": "LambdaCertificateExpiryPolicy3",
            "Effect": "Allow",
            "Action": [
                "acm:DescribeCertificate",
                "acm:GetCertificate",
                "acm:ListCertificates",
                "acm:ListTagsForCertificate"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LambdaCertificateExpiryPolicy4",
            "Effect": "Allow",
            "Action": "SNS:Publish",
            "Resource": "*"
        },
        {
            "Sid": "LambdaCertificateExpiryPolicy5",
            "Effect": "Allow",
            "Action": [
                "SecurityHub:BatchImportFindings",
                "SecurityHub:BatchUpdateFindings",
                "SecurityHub:DescribeHub"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "LambdaCertificateExpiryPolicy6",
            "Effect": "Allow",
            "Action": "cloudwatch:ListMetrics",
            "Resource": "*"
        }
    ]
}
```

3. Create an IAM role and attach the new policy to it. For information about creating an IAM role and attaching a policy, see Creating a role for an AWS service (console).
4. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
5. Create the Lambda function. For more information, see Create a Lambda function with the console. Complete the following steps:

   a. On the **Create function** page, choose the **Author from scratch** option to create the function.
   b. Specify a name such as "handle-expiring-certificates" in the **Function name** field.
   c. Choose Python 3.8 from the **Runtime** list.
   d. Expand **Change default execution role** and choose **Use an existing role**.
   e. Choose the role you previously created from the **Existing role** list.
   f. Choose **Create function**.
   g. Under **Function code**, insert the following code:

```
# Copyright 2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
```

```
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

import json
import boto3
import os
from datetime import datetime, timedelta, timezone
# ----------------------------------------
# setup global data
# ----------------------------------------
utc = timezone.utc
# make today timezone aware
today = datetime.now().replace(tzinfo=utc)
# set up time window for alert - default to 45 if its missing
if os.environ.get('EXPIRY_DAYS') is None:
    expiry_days = 45
else:
    expiry_days = int(os.environ['EXPIRY_DAYS'])
expiry_window = today + timedelta(days = expiry_days)
def lambda_handler(event, context):
    # if this is coming from the ACM event, its for a single certificate
    if (event['detail-type'] == "ACM Certificate Approaching Expiration"):
        response = handle_single_cert(event, context.invoked_function_arn)
    return {
        'statusCode': 200,
        'body': response
    }
def handle_single_cert(event, context_arn):
    cert_client = boto3.client('acm')
    cert_details =
 cert_client.describe_certificate(CertificateArn=event['resources'][0])
    result = 'The following certificate is expiring within ' + str(expiry_days)
 + ' days: ' + cert_details['Certificate']['DomainName']
    # check the expiry window before logging to Security Hub and sending an SNS
    if cert_details['Certificate']['NotAfter'] < expiry_window:
        # This call is the text going into the SNS notification
```

```
        result = result + ' (' + cert_details['Certificate']['CertificateArn']
 + ') '
        # this call is publishing to SH
        result = result + ' - ' + log_finding_to_sh(event, cert_details,
 context_arn)
        # if there's an SNS topic, publish a notification to it
        if os.environ.get('SNS_TOPIC_ARN') is None:
            response = result
        else:
            sns_client = boto3.client('sns')
            response = sns_client.publish(TopicArn=os.environ['SNS_TOPIC_ARN'],
 Message=result, Subject='Certificate Expiration Notification')
    return result
def log_finding_to_sh(event, cert_details, context_arn):
    # setup for security hub
    sh_region = get_sh_region(event['region'])
    sh_hub_arn = "arn:aws:securityhub:{0}:{1}:hub/default".format(sh_region,
 event['account'])
    sh_product_arn = "arn:aws:securityhub:{0}:{1}:product/{1}/
default".format(sh_region, event['account'])
    # check if security hub is enabled, and if the hub arn exists
    sh_client = boto3.client('securityhub', region_name = sh_region)
    try:
        sh_enabled = sh_client.describe_hub(HubArn = sh_hub_arn)
    # the previous command throws an error indicating the hub doesn't exist or
 lambda doesn't have rights to it so we'll stop attempting to use it
    except Exception as error:
        sh_enabled = None
        print ('Default Security Hub product doesn\'t exist')
        response = 'Security Hub disabled'
    # This is used to generate the URL to the cert in the Security Hub Findings
 to link directly to it
    cert_id = right(cert_details['Certificate']['CertificateArn'], 36)
    if sh_enabled:
        # set up a new findings list
        new_findings = []
            # add expiring certificate to the new findings list
        new_findings.append({
            "SchemaVersion": "2018-10-08",
            "Id": cert_id,
            "ProductArn": sh_product_arn,
            "GeneratorId": context_arn,
            "AwsAccountId": event['account'],
            "Types": [
```

```
                "Software and Configuration Checks/AWS Config Analysis"
            ],
            "CreatedAt": event['time'],
            "UpdatedAt": event['time'],
            "Severity": {
                "Original": '89.0',
                "Label": 'HIGH'
            },
            "Title": 'Certificate expiration',
            "Description": 'cert expiry',
            'Remediation': {
                'Recommendation': {
                    'Text': 'A new certificate for ' +
 cert_details['Certificate']['DomainName'] + ' should be imported to replace
 the existing imported certificate before expiration',
                    'Url': "https://console.aws.amazon.com/acm/home?region=" +
 event['region'] + "#/?id=" + cert_id
                }
            },
            'Resources': [
                {
                    'Id': event['id'],
                    'Type': 'ACM Certificate',
                    'Partition': 'aws',
                    'Region': event['region']
                }
            ],
            'Compliance': {'Status': 'WARNING'}
        })
        # push any new findings to security hub
        if new_findings:
            try:
                response =
 sh_client.batch_import_findings(Findings=new_findings)
                if response['FailedCount'] > 0:
                    print("Failed to import {}
 findings".format(response['FailedCount']))
            except Exception as error:
                print("Error: ", error)
                raise
    return json.dumps(response)
# function to setup the sh region
def get_sh_region(event_region):
```

```
    # security hub findings may need to go to a different region so set that
 here
    if os.environ.get('SECURITY_HUB_REGION') is None:
        sh_region_local = event_region
    else:
        sh_region_local = os.environ['SECURITY_HUB_REGION']
    return sh_region_local
# quick function to trim off right side of a string
def right(value, count):
    # To get right part of string, use negative first index in slice.
    return value[-count:]
```

h. Under **Environment variables**, choose **Edit** and optionally add the following variables.

- (Optional) EXPIRY_DAYS

  Specifies how much lead time, in days, before the certificate expiration notice is sent. The function defaults to 45 days, but you can specify custom values.

- (Optional) SNS_TOPIC_ARN

  Specifies an ARN for an Amazon SNS. Provide the full ARN in the format of arn:aws:sns:*<region>*:*<account-number>*:*<topic-name>*.

- (Optional) SECURITY_HUB_REGION

  Specifies an AWS Security Hub CSPM in a different Region. If this is not specified, the Region of the running Lambda function is used. If the function is run in multiple Regions, it may be desirable to have all certificate messages go to Security Hub CSPM in a single Region.

i. Under **Basic settings**, set **Timeout** to 30 seconds.

j. At the top of the page, choose **Deploy**.

Complete the tasks in the following procedure to begin using this solution.

To automate an email notice of expiration

In this example, we provide a single email for each expiring certificate at the moment the event is raised through Amazon EventBridge. By default, ACM raises an event each day for a certificate that is 45 days or less from expiration. (This period can be customized using the PutAccountConfiguration operation of the ACM API.) Each of these events triggers the following cascade of automated actions:

```
ACM raises Amazon EventBridge event #
>>>>>>> events

          Event matches Amazon EventBridge rule #

                    Rule calls Lambda function #

                              Function sends SNS email and logs a Finding in Security
 Hub CSPM
```

1. Create the Lambda function and configure permissions. (Already completed – see To set up a Lambda function and IAM role).
2. Create a *standard* SNS topic for the Lambda function to use to send out notifications. For more information, see Creating an Amazon SNS topic.
3. Subscribe any interested parties to the new SNS topic. For more information, see Subscribing to an Amazon SNS topic.
4. Create an Amazon EventBridge rule to trigger the Lambda function. For more information, see Creating Amazon EventBridge rules that react to events.

   In the Amazon EventBridge console at https://console.aws.amazon.com/events/, navigate to the **Events** > **Rules** page and choose **Create rule**. Specify **Service Name**, **Event Type**, and **Lambda function**. In the **Event Pattern preview** editor, paste the following code:

   ```
   {
     "source": [
       "aws.acm"
     ],
     "detail-type": [
       "ACM Certificate Approaching Expiration"
     ]
   }
   ```

   An event such as Lambda receives is displayed under **Show sample event(s)**:

   ```
   {
     "version": "0",
     "id": "9c95e8e4-96a4-ef3f-b739-b6aa5b193afb",
     "detail-type": "ACM Certificate Approaching Expiration",
     "source": "aws.acm",
   ```

```
  "account": "123456789012",
  "time": "2020-09-30T06:51:08Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:acm:us-east-1:123456789012:certificate/61f50cd4-45b9-4259-b049-
d0a53682fa4b"
  ],
  "detail": {
    "DaysToExpiry": 31,
    "CommonName": "My Awesome Service"
  }
}
```

To clean up

Once you no longer need the example configuration, or any configuration, it is a best practice to remove all traces of it to avoid security problems and unexpected future charges:

- IAM policy and role
- Lambda function
- CloudWatch Events rule
- CloudWatch Logs associated with Lambda
- SNS Topic

Using CloudTrail with AWS Certificate Manager

AWS Certificate Manager is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in ACM. CloudTrail is enabled by default on your AWS account. CloudTrail captures API calls for ACM as events, including calls from the ACM console and code calls to the ACM API operations. If you configure a *trail*, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for ACM. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**.

Using the information collected by CloudTrail, you can determine the request that was made to ACM, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Viewing Events with CloudTrail Event History. When supported event activity occurs in ACM, that activity is recorded in a CloudTrail event along

with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account.

Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs.

For more information about CloudTrail, consult the following documentation:

- AWS CloudTrail User Guide.
- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

Topics

- ACM API actions supported in CloudTrail logging
- Logging API calls for integrated services

ACM API actions supported in CloudTrail logging

ACM supports logging the following actions as events in CloudTrail log files:

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with AWS account root user or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service

For more information, see the CloudTrail userIdentity Element.

ACM records management events and data events in CloudTrail.

Management events

- Adding tags to a certificate (AddTagsToCertificate)
- Changing an ACME account key (ChangeAccountKey)
- Creating an ACME domain validation (CreateAcmeDomainValidation)
- Creating an ACME endpoint (CreateAcmeEndpoint)
- Creating an ACME external account binding (CreateAcmeExternalAccountBinding)
- Deleting an ACME domain validation (DeleteAcmeDomainValidation)
- Deleting an ACME endpoint (DeleteAcmeEndpoint)
- Deleting an ACME external account binding (DeleteAcmeExternalAccountBinding)
- Deleting a certificate (DeleteCertificate)
- Describing an ACME account (DescribeAcmeAccount)
- Describing an ACME domain validation (DescribeAcmeDomainValidation)
- Describing an ACME endpoint (DescribeAcmeEndpoint)
- Describing an ACME external account binding (DescribeAcmeExternalAccountBinding)
- Describing a certificate (DescribeCertificate)
- Exporting a certificate (ExportCertificate)
- Retrieving external account binding credentials (GetAcmeExternalAccountBindingCredentials)
- Retrieving a certificate (GetCertificate)
- Import a certificate (ImportCertificate)
- Issuing a certificate from an ACME endpoint (IssueCertificate)
- Listing ACME accounts (ListAcmeAccounts)
- Listing ACME domain validations (ListAcmeDomainValidations)
- Listing ACME endpoints (ListAcmeEndpoints)
- Listing ACME external account bindings (ListAcmeExternalAccountBindings)
- Listing certificates (ListCertificates)
- Listing tags for a certificate (ListTagsForCertificate)
- Listing tags for a resource (ListTagsForResource)
- Managing an ACME account (ManageAccount)
- Registering an ACME account (NewAccount)
- Removing tags from a certificate (RemoveTagsFromCertificate)
- Requesting a certificate (RequestCertificate)

- Resending validation email (ResendValidationEmail)
- Revoking an ACME account (RevokeAcmeAccount)
- Revoking an ACME external account binding (RevokeAcmeExternalAccountBinding)
- Revoke a certificate (RevokeCertificate)
- Searching certificates (SearchCertificates)
- Tagging a resource (TagResource)
- Removing tags from a resource (UntagResource)
- Updating an ACME domain validation (UpdateAcmeDomainValidation)
- Updating an ACME endpoint (UpdateAcmeEndpoint)

Data events

- Finalizing an order (FinalizeOrder)
- Downloading a certificate (GetCertificate)
- Retrieving the directory (GetDirectory)
- Retrieving an order (GetOrder)
- Listing orders (ListOrders)
- Managing an authorization (ManageAuthorization)
- Requesting a nonce (NewNonce)
- Creating an order (NewOrder)
- Revoking a certificate (RevokeCertificate)

Management events

ACM logs the following operations as CloudTrail management events. Management events are logged by default.

Adding tags to a certificate (AddTagsToCertificate)

The following CloudTrail example shows the results of a call to the AddTagsToCertificate API.

```
{

   "Records":[
      {
```

```
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2016-04-06T13:53:53Z",
         "eventSource":"acm.amazonaws.com",
         "eventName":"AddTagsToCertificate",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/1.10.16",
         "requestParameters":{
            "tags":[
               {
                  "value":"Alice",
                  "key":"Admin"
               }
            ],
            "certificateArn":"arn:aws:acm:us-east-1:123456789012:certificate/
fedcba98-7654-3210-fedc-ba9876543210"
         },
         "responseElements":null,
         "requestID":"fedcba98-7654-3210-fedc-ba9876543210",
         "eventID":"fedcba98-7654-3210-fedc-ba9876543210",
         "eventType":"AwsApiCall",
         "recipientAccountId":"123456789012"
      }
   ]
}
```


Changing an ACME account key (ChangeAccountKey)

The following CloudTrail example shows a log entry for the ChangeAccountKey operation.


```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "anonymous",
    "arn": "anonymous",
```

```
    "accountId": "123456789012",
    "userName": "anonymous"
  },
  "eventTime": "2026-06-10T20:29:57Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "ChangeAccountKey",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "innerJws": "EXAMPLE"
  },
  "responseElements": {
    "location": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acct/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "status": "valid",
    "orders": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acct/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222/orders"
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme-enroll.us-east-1.api.aws"
  }
}
```

Creating an ACME domain validation (CreateAcmeDomainValidation)

The following CloudTrail example shows a log entry for the `CreateAcmeDomainValidation` operation.


```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:45Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "CreateAcmeDomainValidation",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "idempotencyToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "domainName": "example.com",
    "prevalidationOptions": {
      "dnsPrevalidation": {
        "domainScope": {
          "exactDomain": "ENABLED"
        },
        "hostedZoneId": "Z00443972VKAL6HT44MI"
```

```
      }
    }
  },
  "responseElements": {
    "acmeDomainValidationArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/acme-domain-validation/a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222"
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeDomainValidation",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acme-domain-validation/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```


Creating an ACME endpoint (CreateAcmeEndpoint)

The following CloudTrail example shows a log entry for the `CreateAcmeEndpoint` operation.


```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
```

```
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:45Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "CreateAcmeEndpoint",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "idempotencyToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "authorizationBehavior": "PRE_APPROVED",
    "certificateAuthority": {
      "publicCertificateAuthority": {
      }
    }
  },
  "responseElements": {
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
```

```
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Creating an ACME external account binding (CreateAcmeExternalAccountBinding)

The following CloudTrail example shows a log entry for the `CreateAcmeExternalAccountBinding` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:45Z",
```

```
  "eventSource": "acm.amazonaws.com",
  "eventName": "CreateAcmeExternalAccountBinding",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "idempotencyToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "roleArn": "arn:aws:iam::123456789012:role/example-role",
    "expiration": {
      "value": 1,
      "type": "DAYS"
    }
  },
  "responseElements": {
    "externalAccountBinding": {
      "acmeExternalAccountBindingArn": "arn:aws:acm:us-east-1:123456789012:acme-
endpoint/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/acme-external-account-binding/
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "roleArn": "arn:aws:iam::123456789012:role/example-role",
      "expiresAt": "2026-06-11T20:28:45Z"
    }
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeExternalAccountBinding",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acme-external-account-binding/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
  ],
  "eventType": "AwsApiCall",
```

```
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Deleting an ACME domain validation (DeleteAcmeDomainValidation)

The following CloudTrail example shows a log entry for the `DeleteAcmeDomainValidation` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:45Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "DeleteAcmeDomainValidation",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
```

```
  "requestParameters": {
    "acmeDomainValidationArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/acme-domain-validation/a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeDomainValidation",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acme-domain-validation/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```


Deleting an ACME endpoint (DeleteAcmeEndpoint)

The following CloudTrail example shows a log entry for the `DeleteAcmeEndpoint` operation.


```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
```

```
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:45Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "DeleteAcmeEndpoint",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Deleting an ACME external account binding (DeleteAcmeExternalAccountBinding)

The following CloudTrail example shows a log entry for the `DeleteAcmeExternalAccountBinding` operation.


```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:45Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "DeleteAcmeExternalAccountBinding",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeExternalAccountBindingArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/acme-external-account-binding/a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
```

```
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeExternalAccountBinding",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acme-external-account-binding/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Deleting a certificate (DeleteCertificate)

The following CloudTrail example shows the results of a call to the DeleteCertificate API.

```
{

   "Records":[
      {
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2016-03-18T00:00:26Z",
         "eventSource":"acm.amazonaws.com",
         "eventName":"DeleteCertificate",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/1.9.15",
         "requestParameters":{
            "certificateArn":"arn:aws:acm:us-east-1:123456789012:certificate/
fedcba98-7654-3210-fedc-ba9876543210"
         },
```

```
         "responseElements":null,
         "requestID":"01234567-89ab-cdef-0123-456789abcdef",
         "eventID":"01234567-89ab-cdef-0123-456789abcdef",
         "eventType":"AwsApiCall",
         "recipientAccountId":"123456789012"
      }
   ]
}
```

Describing an ACME account (DescribeAcmeAccount)

The following CloudTrail example shows a log entry for the `DescribeAcmeAccount` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:46Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "DescribeAcmeAccount",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
```

```
    "accountUrl": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acct/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Describing an ACME domain validation (DescribeAcmeDomainValidation)

The following CloudTrail example shows a log entry for the `DescribeAcmeDomainValidation` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
```

```
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:29:37Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "DescribeAcmeDomainValidation",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeDomainValidationArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/acme-domain-validation/a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeDomainValidation",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acme-domain-validation/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Describing an ACME endpoint (DescribeAcmeEndpoint)

The following CloudTrail example shows a log entry for the `DescribeAcmeEndpoint` operation.


```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:45Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "DescribeAcmeEndpoint",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
```

```
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Describing an ACME external account binding (DescribeAcmeExternalAccountBinding)

The following CloudTrail example shows a log entry for the `DescribeAcmeExternalAccountBinding` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:29:06Z",
```

```
  "eventSource": "acm.amazonaws.com",
  "eventName": "DescribeAcmeExternalAccountBinding",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeExternalAccountBindingArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/acme-external-account-binding/a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeExternalAccountBinding",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acme-external-account-binding/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```


Describing a certificate (DescribeCertificate)

The following CloudTrail example shows the results of a call to the DescribeCertificate API.

> **Note**
>
> The CloudTrail log for the `DescribeCertificate` operation does not display information about the ACM certificate you specify. You can view information about the certificate by using the console, the AWS Command Line Interface, or the DescribeCertificate API.

```
{
   "Records":[
      {
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2016-03-18T00:00:42Z",
         "eventSource":"acm.amazonaws.com",
         "eventName":"DescribeCertificate",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/1.9.15",
         "requestParameters":{
            "certificateArn":"arn:aws:acm:us-east-1:123456789012:certificate/
fedcba98-7654-3210-fedc-ba9876543210"
         },
         "responseElements":null,
         "requestID":"fedcba98-7654-3210-fedc-ba9876543210",
         "eventID":"fedcba98-7654-3210-fedc-ba9876543210",
         "eventType":"AwsApiCall",
         "recipientAccountId":"123456789012"
      }
   ]
}
```


Exporting a certificate (ExportCertificate)

The following CloudTrail example shows the results of a call to the ExportCertificate API.


```
{
   "Records":[
      {
         "version":"0",
         "id":"01234567-89ab-cdef-0123-456789abcdef",
         "detail-type":"AWS API Call via CloudTrail",
         "source":"aws.acm",
         "account":"123456789012",
```

```
         "time":"2018-05-24T15:28:11Z",
         "region":"us-east-1",
         "resources":[

         ],
         "detail":{
            "eventVersion":"1.04",
            "userIdentity":{
               "type":"Root",
               "principalId":"123456789012",
               "arn":"arn:aws:iam::123456789012:user/Alice",
               "accountId":"123456789012",
               "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
               "userName":"Alice"
            },
            "eventTime":"2018-05-24T15:28:11Z",
            "eventSource":"acm.amazonaws.com",
            "eventName":"ExportCertificate",
            "awsRegion":"us-east-1",
            "sourceIPAddress":"192.0.2.0",
            "userAgent":"aws-cli/1.15.4 Python/2.7.9 Windows/8 botocore/1.10.4",
            "requestParameters":{
              "certificateArn":"arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012",
               "passphrase": "HIDDEN_DUE_TO_SECURITY_REASONS"
            },
            "responseElements":{
                "certificateChain":
                "-----BEGIN CERTIFICATE-----
                base64 certificate
                -----END CERTIFICATE-----
                -----BEGIN CERTIFICATE-----
                base64 certificate
                -----END CERTIFICATE-----",
                "privateKey":"**********",
                "certificate":
                "-----BEGIN CERTIFICATE-----
                base64 certificate
                -----END CERTIFICATE-----",
                "privateKey": "HIDDEN_DUE_TO_SECURITY_REASONS"
            },
            "requestID":"01234567-89ab-cdef-0123-456789abcdef",
            "eventID":"fedcba98-7654-3210-fedc-ba9876543210",
            "readOnly": false,
```

```
            "eventType":"AwsApiCall"
                "managementEvent": true,
                "recipientAccountId": "123456789012",
                "eventCategory": "Management",
                "tlsDetails": {
                     "tlsVersion": "TLSv1.3",
                     "cipherSuite": "TLS_AES_128_GCM_SHA256",
                     "clientProvidedHostHeader": "acm.us-east-1.amazonaws.com"
                 },
                 "sessionCredentialFromConsole": "true"
}
```

Retrieving external account binding credentials (GetAcmeExternalAccountBindingCredentials)

The following CloudTrail example shows a log entry for the `GetAcmeExternalAccountBindingCredentials` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:29:26Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "GetAcmeExternalAccountBindingCredentials",
```

```
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeExternalAccountBindingArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/acme-external-account-binding/a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeExternalAccountBinding",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acme-external-account-binding/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```


Retrieving a certificate (GetCertificate)

The following CloudTrail example shows the results of a call to the GetCertificate API.


```
{

   "Records":[
      {
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
```

```
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2016-03-18T00:00:41Z",
         "eventSource":"acm.amazonaws.com",
         "eventName":"GetCertificate",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/1.9.15",
         "requestParameters":{
            "certificateArn":"arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
         },
         "responseElements":{
            "certificateChain":

            "-----BEGIN CERTIFICATE-----
            Base64-encoded certificate chain
            -----END CERTIFICATE-----",
            "certificate":
            "-----BEGIN CERTIFICATE-----
            Base64-encoded certificate
            -----END CERTIFICATE-----"

         },
         "requestID":"744dd891-ec9c-11e5-ac34-d1e4dfe1a11b",
         "eventID":"7aa4f909-00dd-478a-9a00-b2709bcad2bb",
         "eventType":"AwsApiCall",
         "recipientAccountId":"123456789012"
      }
   ]
}
```

Import a certificate (ImportCertificate)

The following example shows the CloudTrail log entry that records a call to the ACM ImportCertificate API operation.

```
{
   "eventVersion":"1.04",
   "userIdentity":{
      "type":"IAMUser",
```

```
      "principalId":"AIDACKCEVSQ6C2EXAMPLE",
      "arn":"arn:aws:iam::111122223333:user/Alice",
      "accountId":"111122223333",
      "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
      "userName":"Alice"
   },
   "eventTime":"2016-10-04T16:01:30Z",
   "eventSource":"acm.amazonaws.com",
   "eventName":"ImportCertificate",
   "awsRegion":"ap-southeast-2",
   "sourceIPAddress":"54.240.193.129",
   "userAgent":"Coral/Netty",
   "requestParameters":{
      "privateKey":{
         "hb":[
            "byte",
            "byte",
            "byte",
            "..."
         ],
         "offset":0,
         "isReadOnly":false,
         "bigEndian":true,
         "nativeByteOrder":false,
         "mark":-1,
         "position":0,
         "limit":1674,
         "capacity":1674,
         "address":0
      },
      "certificateChain":{
         "hb":[
            "byte",
            "byte",
            "byte",
            "..."
         ],
         "offset":0,
         "isReadOnly":false,
         "bigEndian":true,
         "nativeByteOrder":false,
         "mark":-1,
         "position":0,
         "limit":2105,
```

```
         "capacity":2105,
         "address":0
      },
      "certificate":{
         "hb":[
            "byte",
            "byte",
            "byte",
            "..."
         ],
         "offset":0,
         "isReadOnly":false,
         "bigEndian":true,
         "nativeByteOrder":false,
         "mark":-1,
         "position":0,
         "limit":2503,
         "capacity":2503,
         "address":0
      }
   },
   "responseElements":{
      "certificateArn":"arn:aws:acm:ap-
southeast-2:111122223333:certificate/01234567-89ab-cdef-0123-456789abcdef"
   },
   "requestID":"01234567-89ab-cdef-0123-456789abcdef",
   "eventID":"01234567-89ab-cdef-0123-456789abcdef",
   "eventType":"AwsApiCall",
   "recipientAccountId":"111122223333"
}
```


Issuing a certificate from an ACME endpoint (IssueCertificate)

ACM records an `IssueCertificate` management event when it finishes processing a finalized order. The following CloudTrail example shows a log entry for an issued certificate.


```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "accountId": "123456789012",
    "invokedBy": "acm-acme.amazonaws.com"
  },
  "eventTime": "2026-10-15T20:11:22Z",
```

```
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "IssueCertificate",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "acm-acme.amazonaws.com",
  "userAgent": "acm-acme.amazonaws.com",
  "requestParameters": null,
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::Certificate",
      "ARN": "arn:aws:acm:us-east-1:123456789012:certificate/a1b2c3d4-5678-90ab-cdef-
EXAMPLE55555"
    },
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "serviceEventDetails": {
    "certificateKeyPairOrigin": "ACME",
    "acmeOrderId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "status": "ISSUED",
    "acmeAccountId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "certificateArn": "arn:aws:acm:us-east-1:123456789012:certificate/
a1b2c3d4-5678-90ab-cdef-EXAMPLE55555"
  },
  "eventCategory": "Management"
}
```


Listing ACME accounts (ListAcmeAccounts)

The following CloudTrail example shows a log entry for the `ListAcmeAccounts` operation.


```
{
  "eventVersion": "1.11",
```

```
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:29:37Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "ListAcmeAccounts",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
```

```
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Listing ACME domain validations (ListAcmeDomainValidations)

The following CloudTrail example shows a log entry for the `ListAcmeDomainValidations` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:29:57Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "ListAcmeDomainValidations",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
```

```
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Listing ACME endpoints (ListAcmeEndpoints)

The following CloudTrail example shows a log entry for the `ListAcmeEndpoints` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
```

```
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:56Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "ListAcmeEndpoints",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": null,
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Listing ACME external account bindings (ListAcmeExternalAccountBindings)

The following CloudTrail example shows a log entry for the `ListAcmeExternalAccountBindings` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
```

```
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:29:15Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "ListAcmeExternalAccountBindings",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
```

```
  }
}
```

Listing certificates (ListCertificates)

The following CloudTrail example shows the results of a call to the ListCertificates API.

> **Note**
>
> The CloudTrail log for the `ListCertificates` operation does not display your ACM certificates. You can view the certificate list by using the console, the AWS Command Line Interface, or the ListCertificates API.


```
{
   "Records":[
      {
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2016-03-18T00:00:43Z",
         "eventSource":"acm.amazonaws.com",
         "eventName":"ListCertificates",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/1.9.15",
         "requestParameters":{
            "maxItems":1000,
            "certificateStatuses":[
               "ISSUED"
            ]
         },
         "responseElements":null,
         "requestID":"74c99844-ec9c-11e5-ac34-d1e4dfe1a11b",
         "eventID":"cdfe1051-88aa-4aa3-8c33-a325270bff21",
         "eventType":"AwsApiCall",
```

```
         "recipientAccountId":"123456789012"
      }
   ]
}
```

Listing tags for a certificate (ListTagsForCertificate)

The following CloudTrail example shows the results of a call to the ListTagsForCertificate API.

> **Note**
>
> The CloudTrail log for the `ListTagsForCertificate` operation does not display your tags. You can view the tag list by using the console, the AWS Command Line Interface, or the ListTagsForCertificate API.

```
{
   "Records":[
      {
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2016-04-06T13:30:11Z",
         "eventSource":"acm.amazonaws.com",
         "eventName":"ListTagsForCertificate",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/1.10.16",
         "requestParameters":{
            "certificateArn":"arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
         },
         "responseElements":null,
         "requestID":"b010767f-fbfb-11e5-b596-79e9a97a2544",
         "eventID":"32181be6-a4a0-48d3-8014-c0d972b5163b",
         "eventType":"AwsApiCall",
```

```
         "recipientAccountId":"123456789012"
      }
   ]
}
```

Listing tags for a resource (ListTagsForResource)

The following example shows a CloudTrail log entry for the ListTagsForResource API.

The CloudTrail log for the `ListTagsForResource` operation does not display tags in the response elements.

```
{
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2026-01-15T20:43:00Z",
         "eventSource":"acm.amazonaws.com",
         "eventName":"ListTagsForResource",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/2.0",
         "requestParameters":{
            "resourceArn":"arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-abc123"
         },
         "responseElements":null,
         "requestID":"fedcba98-7654-3210-fedc-ba9876543212",
         "eventID":"12345678-1234-1234-1234-123456789014",
         "eventType":"AwsApiCall",
         "recipientAccountId":"123456789012"
}
```

Managing an ACME account (ManageAccount)

The following CloudTrail example shows a log entry for the `ManageAccount` operation.

```
{
```

```
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "anonymous",
    "arn": "anonymous",
    "accountId": "123456789012",
    "userName": "anonymous"
  },
  "eventTime": "2026-06-10T20:30:40Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "ManageAccount",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "accountId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
  },
  "responseElements": {
    "location": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acct/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "status": "valid",
    "contact": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "orders": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acct/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222/orders"
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
```

```
    "clientProvidedHostHeader": "acm-acme-enroll.us-east-1.api.aws"
  }
}
```

Registering an ACME account (NewAccount)

The following CloudTrail example shows a log entry for the NewAccount operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "anonymous",
    "arn": "anonymous",
    "accountId": "123456789012",
    "userName": "anonymous"
  },
  "eventTime": "2026-06-10T20:30:39Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "NewAccount",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "contact": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "externalAccountBinding": {
      "jwsProtected": "EXAMPLE",
      "payload": "EXAMPLE",
      "signature": "EXAMPLE"
    }
  },
  "responseElements": {
    "location": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acct/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "status": "valid",
    "contact": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "orders": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acct/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222/orders",
    "externalAccountBinding": {
      "jwsProtected": "EXAMPLE",
      "payload": "EXAMPLE",
      "signature": "EXAMPLE"
    },
```

```
    "statusCode": 201
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme-enroll.us-east-1.api.aws"
  }
}
```


Removing tags from a certificate (RemoveTagsFromCertificate)

The following CloudTrail example shows the results of a call to the RemoveTagsFromCertificate API.


```
{
   "Records":[
      {
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2016-04-06T14:10:01Z",
         "eventSource":"acm.amazonaws.com",
```

```
         "eventName":"RemoveTagsFromCertificate",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/1.10.16",
         "requestParameters":{
            "certificateArn":"arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012",
            "tags":[
               {
                  "value":"Bob",
                  "key":"Admin"
               }
            ]
         },
         "responseElements":null,
         "requestID":"40ded461-fc01-11e5-a747-85804766d6c9",
         "eventID":"0cfa142e-ef74-4b21-9515-47197780c424",
         "eventType":"AwsApiCall",
         "recipientAccountId":"123456789012"
      }
   ]
}
```


Requesting a certificate (RequestCertificate)

The following CloudTrail example shows the results of a call to the RequestCertificate API.


```
{
   "Records":[
      {
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2016-03-18T00:00:49Z",
         "eventSource":"acm.amazonaws.com",
         "eventName":"RequestCertificate",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
```

```
         "userAgent":"aws-cli/1.9.15",
         "requestParameters":{
            "domainName":"example.com",
            "validationMethod": "DNS",
            "idempotencyToken":"8186023d89681c3ad5",
            "options": {
            "export": "ENABLED"
        },
        "keyAlgorithm": "RSA_2048"
         },
         "responseElements":{
            "certificateArn":"arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
         },
         "requestID":"77dacef3-ec9c-11e5-ac34-d1e4dfe1a11b",
         "eventID":"a4954cdb-8f38-44c7-8927-a38ad4be3ac8",
         "eventType":"AwsApiCall",
         "tlsDetails": {
           "tlsVersion": "TLSv1.3",
           "cipherSuite": "TLS_AES_128_GCM_SHA256",
           "clientProvidedHostHeader": "acm.us-east-1.amazonaws.com"
          },
         "recipientAccountId":"123456789012"
      }
   ]
}
```


Resending validation email (ResendValidationEmail)

The following CloudTrail example shows the results of a call to the ResendValidationEmail API.


```
{
   "Records":[
      {
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2016-03-17T23:58:25Z",
```

```
         "eventSource":"acm.amazonaws.com",
         "eventName":"ResendValidationEmail",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/1.9.15",
         "requestParameters":{
            "domain":"example.com",
            "certificateArn":"arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012",
            "validationDomain":"example.com"
         },
         "responseElements":null,
         "requestID":"23760b88-ec9c-11e5-b6f4-cb861a6f0a28",
         "eventID":"41c11b06-ca91-4c1c-8c61-af349ea8bab8",
         "eventType":"AwsApiCall",
         "recipientAccountId":"123456789012"
      }
   ]
}
```

Revoking an ACME account (RevokeAcmeAccount)

The following CloudTrail example shows a log entry for the RevokeAcmeAccount operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
```

```
    }
  },
  "eventTime": "2026-06-10T20:28:46Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "RevokeAcmeAccount",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "accountUrl": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acct/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```


Revoking an ACME external account binding (RevokeAcmeExternalAccountBinding)

The following CloudTrail example shows a log entry for the `RevokeAcmeExternalAccountBinding` operation.


```
{
```

```
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:56Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "RevokeAcmeExternalAccountBinding",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeExternalAccountBindingArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/acme-external-account-binding/a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeExternalAccountBinding",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acme-external-account-binding/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
  ],
```

```
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```


Revoke a certificate (RevokeCertificate)

The following CloudTrail example shows the results of a call to the RevokeCertificate API.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:Role-Session-Name",
        "arn": arn:aws:sts::111122223333:assumed-role/Role-Name/Role-Session-Name",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2016-01-01T19:35:52Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime":"2016-01-01T21:11:45Z",
    "eventSource": "acm.amazonaws.com",
    "eventName": "RevokeCertificate",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:128.0) Gecko/20100101
 Firefox/128.0",
```

```
    "requestParameters": {
        "certificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012",
        "revocationReason": "UNSPECIFIED"
    },
    "responseElements": {
        "certificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
    },
    "requestID": "01234567-89ab-cdef-0123-456789abcdef",
    "eventID": "01234567-89ab-cdef-0123-456789abcdef",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "acm.us-east-1.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```

Searching certificates (SearchCertificates)

The following CloudTrail example shows the results of a call to the SearchCertificates API.

```
{

   "Records":[
      {
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2016-04-06T13:53:53Z",
         "eventSource":"acm.amazonaws.com",
         "eventName":"SearchCertificates",
```

```
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/1.10.16",
         "readOnly":true,
         "requestParameters":{
            "maxResults":10,
            "sortBy":"CREATED_AT",
            "sortOrder":"DESCENDING"
         },
         "responseElements":null,
         "requestID":"01234567-89ab-cdef-0123-456789abcdef",
         "eventID":"01234567-89ab-cdef-0123-456789abcdef",
         "eventType":"AwsApiCall",
         "recipientAccountId":"123456789012"
      }
   ]
}
```

Tagging a resource (TagResource)

The following example shows a CloudTrail log entry for the TagResource API.

```
{
         "eventVersion":"1.04",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
         },
         "eventTime":"2026-01-15T20:41:00Z",
         "eventSource":"acm.amazonaws.com",
         "eventName":"TagResource",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"aws-cli/2.0",
         "requestParameters":{
            "resourceArn":"arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-abc123",
            "tags":[
               {
                  "key":"Environment",
                  "value":"Production"
```

```
                }
            ]
        },
        "responseElements":null,
        "requestID":"fedcba98-7654-3210-fedc-ba9876543210",
        "eventID":"12345678-1234-1234-1234-123456789012",
        "eventType":"AwsApiCall",
        "recipientAccountId":"123456789012"
}
```

Removing tags from a resource (UntagResource)

The following example shows a CloudTrail log entry for the UntagResource API.

```
{
        "eventVersion":"1.04",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:user/Alice",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"Alice"
        },
        "eventTime":"2026-01-15T20:42:00Z",
        "eventSource":"acm.amazonaws.com",
        "eventName":"UntagResource",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"192.0.2.0",
        "userAgent":"aws-cli/2.0",
        "requestParameters":{
            "resourceArn":"arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-abc123",
            "tagKeys":["Environment"]
        },
        "responseElements":null,
        "requestID":"fedcba98-7654-3210-fedc-ba9876543211",
        "eventID":"12345678-1234-1234-1234-123456789013",
        "eventType":"AwsApiCall",
        "recipientAccountId":"123456789012"
}
```

Updating an ACME domain validation (UpdateAcmeDomainValidation)

The following CloudTrail example shows a log entry for the `UpdateAcmeDomainValidation` operation.


```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:29:46Z",
  "eventSource": "acm.amazonaws.com",
  "eventName": "UpdateAcmeDomainValidation",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeDomainValidationArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/acme-domain-validation/a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
```

```
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeDomainValidation",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/acme-domain-validation/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```

Updating an ACME endpoint (UpdateAcmeEndpoint)

The following CloudTrail example shows a log entry for the `UpdateAcmeEndpoint` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:28:43Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2026-06-10T20:28:56Z",
```

```
  "eventSource": "acm.amazonaws.com",
  "eventName": "UpdateAcmeEndpoint",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "acmeEndpointArn": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme.us-east-1.api.aws"
  }
}
```


Data events

ACM logs the following ACME protocol operations as CloudTrail data events. Unlike management events, data events are not logged by default. To record them, configure data event logging in your CloudTrail trail or event data store.

Finalizing an order (FinalizeOrder)

The following CloudTrail example shows a data event log entry for the `FinalizeOrder` operation.


```
{
```

```
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:29:27Z",
        "mfaAuthenticated": "false"
      },
      "sourceIdentity": "acm-acme-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    },
    "invokedBy": "acm-acme.amazonaws.com"
  },
  "eventTime": "2026-06-10T20:29:27Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "FinalizeOrder",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "acm-acme.amazonaws.com",
  "userAgent": "acm-acme.amazonaws.com",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "orderId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "csr": "EXAMPLE"
  },
  "responseElements": {
    "status": "processing",
    "expires": "2026-06-17T20:29:27Z",
    "identifiers": [
      {
        "type": "dns",
        "value": "example.example.com"
      }
    ],
    "authorizations": [
```

```
      "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/
authz/a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
    ],
    "finalize": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/order/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222/finalize",
    "location": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/order/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "statusCode": 200
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE66666"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data"
}
```

Downloading a certificate (GetCertificate)

The following CloudTrail example shows a data event log entry for the `GetCertificate` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "anonymous",
    "arn": "anonymous",
    "accountId": "123456789012",
    "userName": "anonymous"
  },
  "eventTime": "2026-06-10T20:29:56Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "GetCertificate",
```

```
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "certId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE77777"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE66666"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme-enroll.us-east-1.api.aws"
  }
}
```


Retrieving the directory (GetDirectory)

The following CloudTrail example shows a data event log entry for the `GetDirectory` operation.


```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "anonymous",
    "arn": "anonymous",
    "accountId": "123456789012",
    "userName": "anonymous"
  },
  "eventTime": "2026-06-10T20:30:39Z",
```

```
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "GetDirectory",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE66666"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme-enroll.us-east-1.api.aws"
  }
}
```

Retrieving an order (GetOrder)

The following CloudTrail example shows a data event log entry for the `GetOrder` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "anonymous",
    "arn": "anonymous",
    "accountId": "123456789012",
    "userName": "anonymous"
  },
```

```
  "eventTime": "2026-06-10T20:30:38Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "GetOrder",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "orderId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE66666"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme-enroll.us-east-1.api.aws"
  }
}
```

Listing orders (ListOrders)

The following CloudTrail example shows a data event log entry for the `ListOrders` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "anonymous",
    "arn": "anonymous",
    "accountId": "123456789012",
```

```
    "userName": "anonymous"
  },
  "eventTime": "2026-06-10T20:29:57Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "ListOrders",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "accountId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE88888"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE66666"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme-enroll.us-east-1.api.aws"
  }
}
```

Managing an authorization (ManageAuthorization)

The following CloudTrail example shows a data event log entry for the `ManageAuthorization` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
```

```
    "type": "Unknown",
    "principalId": "anonymous",
    "arn": "anonymous",
    "accountId": "123456789012",
    "userName": "anonymous"
  },
  "eventTime": "2026-06-10T20:30:40Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "ManageAuthorization",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "authorizationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE99999"
  },
  "responseElements": {
    "location": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/authz/a1b2c3d4-5678-90ab-cdef-EXAMPLE99999",
    "status": "valid",
    "expires": "2026-06-17T20:30:40Z",
    "identifier": {
      "type": "dns",
      "value": "example.example.com"
    },
    "challenges": []
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE66666"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
```

```
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme-enroll.us-east-1.api.aws"
  }
}
```

Requesting a nonce (NewNonce)

The following CloudTrail example shows a data event log entry for the NewNonce operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "anonymous",
    "arn": "anonymous",
    "accountId": "123456789012",
    "userName": "anonymous"
  },
  "eventTime": "2026-06-10T20:30:39Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "NewNonce",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE66666"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
```

```
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme-enroll.us-east-1.api.aws"
  }
}
```

Creating an order (NewOrder)

The following CloudTrail example shows a data event log entry for the `NewOrder` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "anonymous",
    "arn": "anonymous",
    "accountId": "123456789012",
    "userName": "anonymous"
  },
  "eventTime": "2026-06-10T20:29:26Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "NewOrder",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/2.0",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "identifiers": [
      {
        "type": "dns",
        "value": "example.example.com"
      }
    ]
  },
  "responseElements": {
    "status": "ready",
    "expires": "2026-06-17T20:29:26Z",
    "identifiers": [
      {
        "type": "dns",
        "value": "example.example.com"
      }
    ],
    "authorizations": [
```

```
      "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/
authz/a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
    ],
    "finalize": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/order/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222/finalize",
    "location": "https://acm-acme-enroll.us-east-1.api.aws/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111/order/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "statusCode": 201
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE66666"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "acm-acme-enroll.us-east-1.api.aws"
  }
}
```

Revoking a certificate (RevokeCertificate)

The following CloudTrail example shows a data event log entry for the `RevokeCertificate` operation.

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAEXAMPLEID:example-session",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/example-session",
    "accountId": "123456789012",
```

```
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      },
      "attributes": {
        "creationDate": "2026-06-10T20:30:37Z",
        "mfaAuthenticated": "false"
      },
      "sourceIdentity": "acm-acme-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    },
    "invokedBy": "acm-acme.amazonaws.com"
  },
  "eventTime": "2026-06-10T20:30:37Z",
  "eventSource": "acm-acme.amazonaws.com",
  "eventName": "RevokeCertificate",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "acm-acme.amazonaws.com",
  "userAgent": "acm-acme.amazonaws.com",
  "requestParameters": {
    "serverUuid": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "certificate": "EXAMPLE"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::CertificateManager::AcmeEndpoint",
      "ARN": "arn:aws:acm:us-east-1:123456789012:acme-endpoint/a1b2c3d4-5678-90ab-cdef-
EXAMPLE66666"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data"
```

```
}
```

Logging API calls for integrated services

You can use CloudTrail to audit API calls made by services that are integrated with ACM. For more information about using CloudTrail, see the AWS CloudTrail User Guide. The following examples show the types of logs that can be generated depending on the AWS resources on which you provision the ACM certificate.

Topics

- Creating a load balancer

Creating a load balancer

You can use CloudTrail to audit API calls made by services that are integrated with ACM. For more information about using CloudTrail, see the AWS CloudTrail User Guide. The following examples show the types of logs that can be generated depending on the AWS resources on which you provision the ACM certificate.

Topics

- Creating a Load Balancer
- Registering an Amazon EC2 Instance with a Load Balancer
- Encrypting a Private Key
- Decrypting a Private Key

Creating a Load Balancer

The following example shows a call to the `CreateLoadBalancer` function by an IAM user named Alice. The name of the load balancer is `TestLinuxDefault`, and the listener is created using an ACM certificate.

```
{

   "eventVersion":"1.03",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"AIDACKCEVSQ6C2EXAMPLE",
      "arn":"arn:aws:iam::111122223333:user/Alice",
```

```
      "accountId":"111122223333",
      "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
      "userName":"Alice"
   },
   "eventTime":"2016-01-01T21:10:36Z",
   "eventSource":"elasticloadbalancing.amazonaws.com",
   "eventName":"CreateLoadBalancer",
   "awsRegion":"us-east-1",
   "sourceIPAddress":"192.0.2.0/24",
   "userAgent":"aws-cli/1.9.15",
   "requestParameters":{
      "availabilityZones":[
         "us-east-1b"
      ],
      "loadBalancerName":"LinuxTest",
      "listeners":[
         {
            "sSLCertificateId":"arn:aws:acm:us-
east-1:111122223333:certificate/12345678-1234-1234-1234-123456789012",
            "protocol":"HTTPS",
            "loadBalancerPort":443,
            "instanceProtocol":"HTTP",
            "instancePort":80
         }
      ]
   },
   "responseElements":{
      "dNSName":"LinuxTest-1234567890.us-east-1.elb.amazonaws.com"
   },
   "requestID":"19669c3b-b0cc-11e5-85b2-57397210a2e5",
   "eventID":"5d6c00c9-a9b8-46ef-9f3b-4589f5be63f7",
   "eventType":"AwsApiCall",
   "recipientAccountId":"111122223333"
}
```


Registering an Amazon EC2 Instance with a Load Balancer

When you provision your website or application on an Amazon Elastic Compute Cloud (Amazon EC2) instance, the load balancer must be made aware of that instance. This can be accomplished through the Elastic Load Balancing console or the AWS Command Line Interface. The following example shows a call to `RegisterInstancesWithLoadBalancer` for a load balancer named LinuxTest on AWS account 123456789012.

```
{
   "eventVersion":"1.03",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"AIDACKCEVSQ6C2EXAMPLE",
      "arn":"arn:aws:iam::123456789012:user/ALice",
      "accountId":"123456789012",
      "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
      "userName":"Alice",
      "sessionContext":{
         "attributes":{
            "mfaAuthenticated":"false",
            "creationDate":"2016-01-01T19:35:52Z"
         }
      },
      "invokedBy":"signin.amazonaws.com"
   },
   "eventTime":"2016-01-01T21:11:45Z",
   "eventSource":"elasticloadbalancing.amazonaws.com",
   "eventName":"RegisterInstancesWithLoadBalancer",
   "awsRegion":"us-east-1",
   "sourceIPAddress":"192.0.2.0/24",
   "userAgent":"signin.amazonaws.com",
   "requestParameters":{
      "loadBalancerName":"LinuxTest",
      "instances":[
         {
            "instanceId":"i-c67f4e78"
         }
      ]
   },
   "responseElements":{
      "instances":[
         {
            "instanceId":"i-c67f4e78"
         }
      ]
   },
   "requestID":"438b07dc-b0cc-11e5-8afb-cda7ba020551",
   "eventID":"9f284ca6-cbe5-42a1-8251-4f0e6b5739d6",
   "eventType":"AwsApiCall",
   "recipientAccountId":"123456789012"
}
```

Encrypting a Private Key

The following example shows an `Encrypt` call that encrypts the private key associated with an ACM certificate. Encryption is performed within AWS.


```
{
   "Records":[
      {
         "eventVersion":"1.03",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/acm",
            "accountId":"111122223333",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "userName":"acm"
         },
         "eventTime":"2016-01-05T18:36:29Z",
         "eventSource":"kms.amazonaws.com",
         "eventName":"Encrypt",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"AWS Internal",
         "userAgent":"aws-internal",
         "requestParameters":{
            "keyId":"arn:aws:kms:us-east-1:123456789012:alias/aws/acm",
            "encryptionContext":{
               "aws:acm:arn":"arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
            }
         },
         "responseElements":null,
         "requestID":"3c417351-b3db-11e5-9a24-7d9457362fcc",
         "eventID":"1794fe70-796a-45f5-811b-6584948f24ac",
         "readOnly":true,
         "resources":[
            {
               "ARN":"arn:aws:kms:us-
east-1:123456789012:key/87654321-4321-4321-4321-210987654321",
               "accountId":"123456789012"
            }
         ],
         "eventType":"AwsServiceEvent",
         "recipientAccountId":"123456789012"
      }
```

```
   ]
}
```

Decrypting a Private Key

The following example shows a `Decrypt` call that decrypts the private key associated with an ACM certificate. Decryption is performed within AWS, and the decrypted key never leaves AWS.

```
{
   "eventVersion":"1.03",
   "userIdentity":{
      "type":"AssumedRole",
      "principalId":"AIDACKCEVSQ6C2EXAMPLE:1aba0dc8b3a728d6998c234a99178eff",
      "arn":"arn:aws:sts::111122223333:assumed-role/
DecryptACMCertificate/1aba0dc8b3a728d6998c234a99178eff",
      "accountId":"111122223333",
      "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
      "sessionContext":{
         "attributes":{
            "mfaAuthenticated":"false",
            "creationDate":"2016-01-01T21:13:28Z"
         },
         "sessionIssuer":{
            "type":"Role",
            "principalId":"APKAEIBAERJR2EXAMPLE",
            "arn":"arn:aws:iam::111122223333:role/DecryptACMCertificate",
            "accountId":"111122223333",
            "userName":"DecryptACMCertificate"
         }
      }
   },
   "eventTime":"2016-01-01T21:13:28Z",
   "eventSource":"kms.amazonaws.com",
   "eventName":"Decrypt",
   "awsRegion":"us-east-1",
   "sourceIPAddress":"AWS Internal",
   "userAgent":"aws-internal/3",
   "requestParameters":{
      "encryptionContext":{
         "aws:elasticloadbalancing:arn":"arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/LinuxTest",
         "aws:acm:arn":"arn:aws:acm:us-
east-1:123456789012:certificate/87654321-4321-4321-4321-210987654321"
```

```
      }
   },
   "responseElements":null,
   "requestID":"809a70ff-b0cc-11e5-8f42-c7fdf1cb6e6a",
   "eventID":"7f89f7a7-baff-4802-8a88-851488607fb9",
   "readOnly":true,
   "resources":[
      {
         "ARN":"arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012",
         "accountId":"123456789012"
      }
   ],
   "eventType":"AwsServiceEvent",
   "recipientAccountId":"123456789012"
}
```

Supported CloudWatch metrics

Amazon CloudWatch is a monitoring service for AWS resources. You can use CloudWatch to collect and track metrics, set alarms, and automatically react to changes in your AWS resources.

The `AWS/CertificateManager` namespace includes the following metrics.

Metric	Description	Unit	Dimensions
`DaysToExpiry`	Number of days until a certificate expires. ACM publishes this metric twice per day for every certificate until expiration, and stops publishing it after a certificate expires.	Integer	CertificateArn • Value: ARN of the certificate
`CertificateIssuanceSuccess`	The number of certificates successfully issued through	Count	AcmeEndpointArn • Value: ARN of the ACME endpoint

Metric	Description	Unit	Dimensions
	an ACME endpoint. ACM publishes a value of 1 for each successful issuance and 0 otherwise.		
`CertificateIssuanceFailed`	The number of certificate issuance attempts that failed for an ACME endpoint. ACM publishes a value of 1 for each failed issuance and 0 otherwise.	Count	AcmeEndpointArn • Value: ARN of the ACME endpoint

For more information about CloudWatch metrics, see the following topics:

- Using Amazon CloudWatch Metrics
- Creating Amazon CloudWatch Alarms

Use AWS Certificate Manager with the SDK for Java

You can use the AWS Certificate Manager API to interact with the service programmatically by sending HTTP requests. For more information, see the AWS Certificate Manager API Reference.

In addition to the web API (or HTTP API), you can use the AWS SDKs and command line tools to interact with ACM and other services. For more information, see Tools for Amazon Web Services.

The following topics show you how to use one of the AWS SDKs, the AWS SDK for Java, to perform some of the available operations in the AWS Certificate Manager API.

Topics

- Adding tags to a certificate
- Creating an ACME domain validation
- Creating an ACME endpoint
- Creating an ACME external account binding
- Deleting an ACME domain validation
- Deleting an ACME endpoint
- Deleting an ACME external account binding
- Deleting a certificate
- Describing an ACME account
- Describing an ACME domain validation
- Describing an ACME endpoint
- Describing an ACME external account binding
- Describing a certificate
- Exporting a certificate
- Getting ACME external account binding credentials
- Retrieve a certificate and certificate chain
- Importing a certificate
- Listing ACME accounts
- Listing ACME domain validations
- Listing ACME endpoints
- Listing ACME external account bindings

- Listing certificates
- Listing certificate tags
- Listing tags for a resource
- Removing tags from a certificate
- Renewing a certificate
- Requesting a certificate
- Resending validation email
- Revoking an ACME account
- Revoking an ACME external account binding
- Searching certificates
- Updating a certificate's validation method
- Tagging a resource
- Listing domain validation status
- Removing tags from a resource
- Updating an ACME domain validation
- Updating an ACME endpoint

Adding tags to a certificate

The following example shows how to use the AddTagsToCertificate function.

```
package com.amazonaws.samples;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;

import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.model.ImportCertificateRequest;
import com.amazonaws.services.certificatemanager.model.ImportCertificateResult;
```

```
/**
 * This sample demonstrates how to use the ImportCertificate function in the AWS
 Certificate Manager
 * service.
 *
 * Input parameters:
 *   Accesskey - AWS access key
 *   SecretKey - AWS secret key
 *   CertificateArn - Use to reimport a certificate (not included in this example).
 *   region - AWS region
 *   Certificate - PEM file that contains the certificate to import. Ex: /data/certs/
servercert.pem
 *   CertificateChain - The certificate chain, not including the end-entity
 certificate.
 *   PrivateKey - The private key that matches the public key in the certificate.
 *
 * Output parameter:
 *   CertificcateArn - The ARN of the imported certificate.
 *
 */
public class AWSCertificateManagerSample {

    public static void main(String[] args) throws IOException {
     String accessKey = "";
     String secretKey = "";
     String certificateArn = null;
     Regions region = Regions.DEFAULT_REGION;
     String serverCertFilePath = "";
     String privateKeyFilePath = "";
     String caCertFilePath = "";

     ImportCertificateRequest req = new ImportCertificateRequest()
       .withCertificate(getCertContent(serverCertFilePath))
       .withPrivateKey(getCertContent(privateKeyFilePath))

 .withCertificateChain(getCertContent(caCertFilePath)).withCertificateArn(certificateArn);

     AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.standard().withRegion(region)
       .withCredentials(new AWSStaticCredentialsProvider(new
 BasicAWSCredentials(accessKey, secretKey)))
       .build();
     ImportCertificateResult result = client.importCertificate(req);
```

```
     System.out.println(result.getCertificateArn());

     List<Tag> expectedTags =
 ImmutableList.of(Tag.builder().withKey("key").withValue("value").build());

     AddTagsToCertificateRequest addTagsToCertificateRequest =
 AddTagsToCertificateRequest.builder()
             .withCertificateArn(result.getCertificateArn())
             .withTags(tags)
             .build();

     client.addTagsToCertificate(addTagsToCertificateRequest);
    }

    private static ByteBuffer getCertContent(String filePath) throws IOException {
     String fileContent = new String(Files.readAllBytes(Paths.get(filePath)));
     return StandardCharsets.UTF_8.encode(fileContent);
    }
}
```

Creating an ACME domain validation

The following example shows how to use the CreateAcmeDomainValidation function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.CreateAcmeDomainValidationRequest;
import
 com.amazonaws.services.certificatemanager.model.CreateAcmeDomainValidationResult;
import com.amazonaws.services.certificatemanager.model.PrevalidationOptions;
import com.amazonaws.services.certificatemanager.model.DnsPrevalidationOptions;
import com.amazonaws.services.certificatemanager.model.DomainScope;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();
```

```
        // Configure domain scope.
        DomainScope domainScope = new DomainScope()
            .withExactDomain("ENABLED")
            .withSubdomains("ENABLED")
            .withWildcards("ENABLED");

        // Configure DNS prevalidation options.
        DnsPrevalidationOptions dnsOptions = new DnsPrevalidationOptions()
            .withDomainScope(domainScope)
            .withHostedZoneId("Z1234567890");

        PrevalidationOptions prevalidationOptions = new PrevalidationOptions()
            .withDnsPrevalidation(dnsOptions);

        // Create the request.
        CreateAcmeDomainValidationRequest req = new CreateAcmeDomainValidationRequest()
            .withAcmeEndpointArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
example")
            .withDomainName("example.com")
            .withPrevalidationOptions(prevalidationOptions);

        // Create the domain validation.
        CreateAcmeDomainValidationResult result =
 client.createAcmeDomainValidation(req);
        System.out.println(result.getAcmeDomainValidationArn());
    }
}
```

Creating an ACME endpoint

The following example shows how to use the CreateAcmeEndpoint function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.CreateAcmeEndpointRequest;
import com.amazonaws.services.certificatemanager.model.CreateAcmeEndpointResult;
import com.amazonaws.services.certificatemanager.model.CertificateAuthority;
import com.amazonaws.services.certificatemanager.model.PublicCertificateAuthority;
import com.amazonaws.services.certificatemanager.model.Tag;

import java.util.ArrayList;
```

```
import java.util.Arrays;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Configure the certificate authority.
        PublicCertificateAuthority publicCA = new PublicCertificateAuthority()
            .withAllowedKeyAlgorithms(Arrays.asList("RSA_2048", "EC_prime256v1"));

        CertificateAuthority ca = new CertificateAuthority()
            .withPublicCertificateAuthority(publicCA);

        // Specify tags for the endpoint.
        ArrayList<Tag> tags = new ArrayList<>();
        tags.add(new Tag().withKey("Environment").withValue("Production"));

        // Specify tags to apply to certificates issued by this endpoint.
        ArrayList<Tag> certTags = new ArrayList<>();
        certTags.add(new Tag().withKey("IssuedBy").withValue("ACME"));

        // Create the request.
        CreateAcmeEndpointRequest req = new CreateAcmeEndpointRequest()
            .withAuthorizationBehavior("PRE_APPROVED")
            .withContact("REQUIRED")
            .withCertificateAuthority(ca)
            .withTags(tags)
            .withCertificateTags(certTags);

        // Create the ACME endpoint.
        CreateAcmeEndpointResult result = client.createAcmeEndpoint(req);
        System.out.println(result.getAcmeEndpointArn());
    }
}
```

Creating an ACME external account binding

The following example shows how to use the CreateAcmeExternalAccountBinding function.

```
package com.amazonaws.samples;
```

```
import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.CreateAcmeExternalAccountBindingRequest;
import
 com.amazonaws.services.certificatemanager.model.CreateAcmeExternalAccountBindingResult;
import com.amazonaws.services.certificatemanager.model.Expiration;
import com.amazonaws.services.certificatemanager.model.Tag;

import java.util.ArrayList;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Configure expiration.
        Expiration expiration = new Expiration()
            .withValue(7L)
            .withType("DAYS");

        // Specify tags.
        ArrayList<Tag> tags = new ArrayList<>();
        tags.add(new Tag().withKey("Environment").withValue("Production"));

        // Create the request.
        CreateAcmeExternalAccountBindingRequest req = new
 CreateAcmeExternalAccountBindingRequest()
            .withAcmeEndpointArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
example")
            .withRoleArn("arn:aws:iam::123456789012:role/AcmeClientRole")
            .withExpiration(expiration)
            .withTags(tags);

        // Create the external account binding.
        CreateAcmeExternalAccountBindingResult result =
 client.createAcmeExternalAccountBinding(req);
        System.out.println(result);
    }
}
```

Deleting an ACME domain validation

The following example shows how to use the DeleteAcmeDomainValidation function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.DeleteAcmeDomainValidationRequest;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        DeleteAcmeDomainValidationRequest req = new DeleteAcmeDomainValidationRequest()
            .withAcmeDomainValidationArn("arn:aws:acm:us-east-1:123456789012:acme-
endpoint/ep-example/acme-domain-validation/dv-example");

        // Delete the domain validation.
        client.deleteAcmeDomainValidation(req);
    }
}
```

Deleting an ACME endpoint

The following example shows how to use the DeleteAcmeEndpoint function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.DeleteAcmeEndpointRequest;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {
```

```
        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        DeleteAcmeEndpointRequest req = new DeleteAcmeEndpointRequest()
            .withAcmeEndpointArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
example");

        // Delete the ACME endpoint.
        client.deleteAcmeEndpoint(req);
    }
}
```

Deleting an ACME external account binding

The following example shows how to use the DeleteAcmeExternalAccountBinding function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.DeleteAcmeExternalAccountBindingRequest;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        DeleteAcmeExternalAccountBindingRequest req = new
 DeleteAcmeExternalAccountBindingRequest()
            .withAcmeExternalAccountBindingArn("arn:aws:acm:us-
east-1:123456789012:acme-endpoint/ep-example/acme-external-account-binding/eab-
example");

        // Delete the external account binding.
        client.deleteAcmeExternalAccountBinding(req);
    }
}
```

Deleting a certificate

The following example shows how to use the DeleteCertificate function. If successful, the function returns an empty set {}.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.DeleteCertificateRequest;
import com.amazonaws.services.certificatemanager.model.DeleteCertificateResult;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;

import com.amazonaws.services.certificatemanager.model.InvalidArnException;
import com.amazonaws.services.certificatemanager.model.ResourceInUseException;
import com.amazonaws.services.certificatemanager.model.ResourceNotFoundException;
import com.amazonaws.AmazonClientException;

/**
 * This sample demonstrates how to use the DeleteCertificate function in the AWS
 Certificate
 * Manager service.
 *
 * Input parameter:
 *   CertificateArn - The ARN of the certificate to delete.
 *
 */

public class AWSCertificateManagerExample {

   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().getCredentials();
      }
```

```
      catch (Exception ex) {
          throw new AmazonClientException("Cannot load the credentials from file.",
 ex);
      }

      // Create a client.
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.US_EAST_1)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
              .build();

      // Create a request object and specify the ARN of the certificate to delete.
      DeleteCertificateRequest req = new DeleteCertificateRequest();

 req.setCertificateArn("arn:aws:acm:region:account:certificate/
12345678-1234-1234-1234-123456789012");

      // Delete the specified certificate.
      DeleteCertificateResult result = null;
      try {
         result = client.deleteCertificate(req);
      }
      catch (InvalidArnException ex)
      {
         throw ex;
      }
      catch (ResourceInUseException ex)
      {
         throw ex;
      }
      catch (ResourceNotFoundException ex)
      {
         throw ex;
      }

      // Display the result.
      System.out.println(result);

   }
}
```

Describing an ACME account

The following example shows how to use the DescribeAcmeAccount function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.DescribeAcmeAccountRequest;
import com.amazonaws.services.certificatemanager.model.DescribeAcmeAccountResult;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        DescribeAcmeAccountRequest req = new DescribeAcmeAccountRequest()
            .withAcmeEndpointArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
example")
            .withAccountUrl("https://acme.example.com/acme/acct/12345");

        // Describe the ACME account.
        DescribeAcmeAccountResult result = client.describeAcmeAccount(req);
        System.out.println(result);
    }
}
```

Describing an ACME domain validation

The following example shows how to use the DescribeAcmeDomainValidation function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.DescribeAcmeDomainValidationRequest;
import
 com.amazonaws.services.certificatemanager.model.DescribeAcmeDomainValidationResult;
```

```
public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        DescribeAcmeDomainValidationRequest req = new
 DescribeAcmeDomainValidationRequest()
            .withAcmeDomainValidationArn("arn:aws:acm:us-east-1:123456789012:acme-
endpoint/ep-example/acme-domain-validation/dv-example");

        // Describe the domain validation.
        DescribeAcmeDomainValidationResult result =
 client.describeAcmeDomainValidation(req);
        System.out.println(result);
    }
}
```

Describing an ACME endpoint

The following example shows how to use the DescribeAcmeEndpoint function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.DescribeAcmeEndpointRequest;
import com.amazonaws.services.certificatemanager.model.DescribeAcmeEndpointResult;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        DescribeAcmeEndpointRequest req = new DescribeAcmeEndpointRequest()
            .withAcmeEndpointArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
example");
```

```
        // Describe the ACME endpoint.
        DescribeAcmeEndpointResult result = client.describeAcmeEndpoint(req);
        System.out.println(result);
    }
}
```

Describing an ACME external account binding

The following example shows how to use the DescribeAcmeExternalAccountBinding function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.DescribeAcmeExternalAccountBindingRequest;
import
 com.amazonaws.services.certificatemanager.model.DescribeAcmeExternalAccountBindingResult;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        DescribeAcmeExternalAccountBindingRequest req = new
 DescribeAcmeExternalAccountBindingRequest()
            .withAcmeExternalAccountBindingArn("arn:aws:acm:us-
east-1:123456789012:acme-endpoint/ep-example/acme-external-account-binding/eab-
example");

        // Describe the external account binding.
        DescribeAcmeExternalAccountBindingResult result =
 client.describeAcmeExternalAccountBinding(req);
        System.out.println(result);
    }
}
```

Describing a certificate

The following example shows how to use the DescribeCertificate function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.DescribeCertificateRequest;
import com.amazonaws.services.certificatemanager.model.DescribeCertificateResult;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;

import com.amazonaws.services.certificatemanager.model.InvalidArnException;
import com.amazonaws.services.certificatemanager.model.ResourceNotFoundException;
import com.amazonaws.AmazonClientException;

/**
 * This sample demonstrates how to use the DescribeCertificate function in the AWS
 Certificate
 * Manager service.
 *
 * Input parameter:
 *   CertificateArn - The ARN of the certificate to be described.
 *
 * Output parameter:
 *   Certificate information
 *
 */

public class AWSCertificateManagerExample {

   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().getCredentials();
```

```
      }
      catch (Exception ex) {
          throw new AmazonClientException("Cannot load the credentials from file.",
 ex);
      }

      // Create a client.
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.US_EAST_1)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
              .build();

      // Create a request object and set the ARN of the certificate to be described.
      DescribeCertificateRequest req = new DescribeCertificateRequest();

 req.setCertificateArn("arn:aws:acm:region:account:certificate/
12345678-1234-1234-1234-123456789012");

      DescribeCertificateResult result = null;
      try{
         result = client.describeCertificate(req);
      }
      catch (InvalidArnException ex)
      {
         throw ex;
      }
      catch (ResourceNotFoundException ex)
      {
         throw ex;
      }

      // Display the certificate information.
      System.out.println(result);

   }
}
```

If successful, the preceding example displays information similar to the following.

```
{
    Certificate: {
        CertificateArn:
 arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012,
```

```
        DomainName: www.example.com,
        SubjectAlternativeNames: [www.example.com],
        DomainValidationOptions: [{
            DomainName: www.example.com,
        }],
        Serial: 10: 0a,
        Subject: C=US,
        ST=WA,
        L=Seattle,
        O=ExampleCompany,
        OU=sales,
        CN=www.example.com,
        Issuer: ExampleCompany,
        ImportedAt: FriOct0608: 17: 39PDT2017,
        Status: ISSUED,
        NotBefore: ThuOct0510: 14: 32PDT2017,
        NotAfter: SunOct0310: 14: 32PDT2027,
        KeyAlgorithm: RSA-2048,
        SignatureAlgorithm: SHA256WITHRSA,
        InUseBy: [],
        Type: IMPORTED,
    }
}
```

Exporting a certificate

The following example shows how to use the ExportCertificate function. The function exports a certificate in the PKCS #8 format. It also exports the certificate chain and private key. In the example, the passphrase for the key is stored in a local file.

> **Note**
>
> If you want to export public certificates issued through ACM, see ACM exportable public certificates.

```
package com.amazonaws.samples;

import com.amazonaws.AmazonClientException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
```

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;


import com.amazonaws.services.certificatemanager.model.ExportCertificateRequest;
import com.amazonaws.services.certificatemanager.model.ExportCertificateResult;

import com.amazonaws.services.certificatemanager.model.InvalidArnException;
import com.amazonaws.services.certificatemanager.model.InvalidTagException;
import com.amazonaws.services.certificatemanager.model.ResourceNotFoundException;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

public class ExportCertificate {

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

      // Create a client.
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.your_region)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
              .build();

      // Initialize a file descriptor for the passphrase file.
```

```
      RandomAccessFile file_passphrase = null;

      // Initialize a buffer for the passphrase.
      ByteBuffer buf_passphrase = null;

      // Create a file stream for reading the private key passphrase.
      try {
         file_passphrase = new RandomAccessFile("C:\\Temp\\password.txt", "r");
      }
      catch (IllegalArgumentException ex) {
         throw ex;
      }
      catch (SecurityException ex) {
         throw ex;
      }
      catch (FileNotFoundException ex) {
         throw ex;
      }

      // Create a channel to map the file.
      FileChannel channel_passphrase = file_passphrase.getChannel();

      // Map the file to the buffer.
      try {
         buf_passphrase = channel_passphrase.map(FileChannel.MapMode.READ_ONLY, 0,
channel_passphrase.size());

         // Clean up after the file is mapped.
         channel_passphrase.close();
         file_passphrase.close();
      }
      catch (IOException ex)
      {
         throw ex;
      }

      // Create a request object.
      ExportCertificateRequest req = new ExportCertificateRequest();

      // Set the certificate ARN.
      req.withCertificateArn("arn:aws:acm:region:account:"
            +"certificate/M12345678-1234-1234-1234-123456789012");

      // Set the passphrase.
```

```
      req.withPassphrase(buf_passphrase);

      // Export the certificate.
      ExportCertificateResult result = null;

      try {
         result = client.exportCertificate(req);
      }
      catch(InvalidArnException ex)
      {
         throw ex;
      }
      catch (InvalidTagException ex)
      {
         throw ex;
      }
      catch (ResourceNotFoundException ex)
      {
         throw ex;
      }

      // Clear the buffer.
      buf_passphrase.clear();

      // Display the certificate and certificate chain.
      String certificate = result.getCertificate();
      System.out.println(certificate);

      String certificate_chain = result.getCertificateChain();
      System.out.println(certificate_chain);

      // This example retrieves but does not display the private key.
      String private_key = result.getPrivateKey();
   }
}
```

Getting ACME external account binding credentials

The following example shows how to use the GetAcmeExternalAccountBindingCredentials function.

```
package com.amazonaws.samples;
```

```
import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.GetAcmeExternalAccountBindingCredentialsReques
import
 com.amazonaws.services.certificatemanager.model.GetAcmeExternalAccountBindingCredentialsResult

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        GetAcmeExternalAccountBindingCredentialsRequest req = new
 GetAcmeExternalAccountBindingCredentialsRequest()
            .withAcmeExternalAccountBindingArn("arn:aws:acm:us-
east-1:123456789012:acme-endpoint/ep-example/acme-external-account-binding/eab-
example");

        // Get the credentials.
        GetAcmeExternalAccountBindingCredentialsResult result =
 client.getAcmeExternalAccountBindingCredentials(req);

        // Retrieve the key ID and MAC key from the response.
        System.out.println("KeyId: " + result.getKeyId());
        System.out.println("MacKey: " + result.getMacKey());
    }
}
```

Retrieve a certificate and certificate chain

The following example shows how to use the GetCertificate function.

```
package com.amazonaws.samples;

import com.amazonaws.regions.Regions;
import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.GetCertificateRequest;
import com.amazonaws.services.certificatemanager.model.GetCertificateResult;
```

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;

import com.amazonaws.services.certificatemanager.model.InvalidArnException;
import com.amazonaws.services.certificatemanager.model.ResourceNotFoundException;
import com.amazonaws.services.certificatemanager.model.RequestInProgressException;
import com.amazonaws.AmazonClientException;

/**
 * This sample demonstrates how to use the GetCertificate function in the AWS
 Certificate
 * Manager service.
 *
 * Input parameter:
 *   CertificateArn - The ARN of the certificate to retrieve.
 *
 * Output parameters:
 *   Certificate - A base64-encoded certificate in PEM format.
 *   CertificateChain - The base64-encoded certificate chain in PEM format.
 *
 */

public class AWSCertificateManagerExample {

   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().getCredentials();
      }
      catch (Exception ex) {
          throw new AmazonClientException("Cannot load the credentials from the
 credential profiles file.", ex);
      }

      // Create a client.
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.US_EAST_1)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
```

```
                .build();

      // Create a request object and set the ARN of the certificate to be described.
      GetCertificateRequest req = new GetCertificateRequest();

 req.setCertificateArn("arn:aws:acm:region:account:certificate/
12345678-1234-1234-1234-123456789012");

      // Retrieve the certificate and certificate chain.
      // If you recently requested the certificate, loop until it has been created.
      GetCertificateResult result = null;
      long totalTimeout = 120000l;
      long timeSlept = 0l;
      long sleepInterval = 10000l;
      while (result == null && timeSlept < totalTimeout) {
         try {
            result = client.getCertificate(req);
         }
         catch (RequestInProgressException ex) {
            Thread.sleep(sleepInterval);
         }
         catch (ResourceNotFoundException ex)
         {
            throw ex;
         }
         catch (InvalidArnException ex)
         {
            throw ex;
         }

         timeSlept += sleepInterval;
      }

      // Display the certificate information.
      System.out.println(result);
   }
}
```

The preceding example creates output similar to the following.

```
{Certificate: -----BEGIN CERTIFICATE-----
    base64-encoded certificate
-----END CERTIFICATE-----,
```

```
CertificateChain: -----BEGIN CERTIFICATE-----
    base64-encoded certificate chain
-----END CERTIFICATE-----
}
```

Importing a certificate

The following example shows how to use the ImportCertificate function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;

import com.amazonaws.services.certificatemanager.model.ImportCertificateRequest;
import com.amazonaws.services.certificatemanager.model.ImportCertificateResult;
import com.amazonaws.services.certificatemanager.model.LimitExceededException;
import com.amazonaws.services.certificatemanager.model.ResourceNotFoundException;
import com.amazonaws.AmazonClientException;
import java.io.FileNotFoundException;
import java.io.IOException;

import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

/**
 * This sample demonstrates how to use the ImportCertificate function in the AWS
 Certificate Manager
 * service.
 *
 * Input parameters:
 *   Certificate - PEM file that contains the certificate to import.
 *   CertificateArn - Use to reimport a certificate (not included in this example).
 *   CertificateChain - The certificate chain, not including the end-entity
 certificate.
 *   PrivateKey - The private key that matches the public key in the certificate.
 *
```

```
 * Output parameter:
 *   CertificcateArn - The ARN of the imported certificate.
 *
 */
public class AWSCertificateManagerSample {

   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().getCredentials();
      }
      catch (Exception ex) {
          throw new AmazonClientException(
              "Cannot load the credentials from file.", ex);
      }

      // Create a client.
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.US_EAST_1)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
              .build();

      // Initialize the file descriptors.
      RandomAccessFile file_certificate = null;
      RandomAccessFile file_chain = null;
      RandomAccessFile file_key = null;

      // Initialize the buffers.
      ByteBuffer buf_certificate = null;
      ByteBuffer buf_chain = null;
      ByteBuffer buf_key = null;

      // Create the file streams for reading.
      try {
         file_certificate = new RandomAccessFile("C:\\Temp\\certificate.pem", "r");
         file_chain = new RandomAccessFile("C:\\Temp\\chain.pem", "r");
         file_key = new RandomAccessFile("C:\\Temp\\private_key.pem", "r");
      }
      catch (IllegalArgumentException ex) {
         throw ex;
```

```
      }
      catch (SecurityException ex) {
         throw ex;
      }
      catch (FileNotFoundException ex) {
         throw ex;
      }

      // Create channels for mapping the files.
      FileChannel channel_certificate = file_certificate.getChannel();
      FileChannel channel_chain = file_chain.getChannel();
      FileChannel channel_key = file_key.getChannel();

      // Map the files to buffers.
      try {
         buf_certificate = channel_certificate.map(FileChannel.MapMode.READ_ONLY, 0,
channel_certificate.size());
         buf_chain = channel_chain.map(FileChannel.MapMode.READ_ONLY, 0,
channel_chain.size());
         buf_key = channel_key.map(FileChannel.MapMode.READ_ONLY, 0,
channel_key.size());

         // The files have been mapped, so clean up.
         channel_certificate.close();
         channel_chain.close();
         channel_key.close();
         file_certificate.close();
         file_chain.close();
         file_key.close();
      }
      catch (IOException ex)
      {
         throw ex;
      }

      // Create a request object and set the parameters.
      ImportCertificateRequest req = new ImportCertificateRequest();
      req.setCertificate(buf_certificate);
      req.setCertificateChain(buf_chain);
      req.setPrivateKey(buf_key);

      // Import the certificate.
      ImportCertificateResult result = null;
      try {
```

```
         result = client.importCertificate(req);
      }
      catch(LimitExceededException ex)
      {
         throw ex;
      }
      catch (ResourceNotFoundException ex)
      {
         throw ex;
      }

      // Clear the buffers.
      buf_certificate.clear();
      buf_chain.clear();
      buf_key.clear();

      // Retrieve and display the certificate ARN.
      String arn = result.getCertificateArn();
      System.out.println(arn);
    }
}
```

Listing ACME accounts

The following example shows how to use the ListAcmeAccounts function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.ListAcmeAccountsRequest;
import com.amazonaws.services.certificatemanager.model.ListAcmeAccountsResult;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        ListAcmeAccountsRequest req = new ListAcmeAccountsRequest()
```

```
            .withAcmeEndpointArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
example")
            .withMaxResults(10);

        // List the ACME accounts.
        ListAcmeAccountsResult result = client.listAcmeAccounts(req);
        System.out.println(result);
    }
}
```

Listing ACME domain validations

The following example shows how to use the ListAcmeDomainValidations function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.ListAcmeDomainValidationsRequest;
import com.amazonaws.services.certificatemanager.model.ListAcmeDomainValidationsResult;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        ListAcmeDomainValidationsRequest req = new ListAcmeDomainValidationsRequest()
            .withAcmeEndpointArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
example")
            .withMaxResults(10);

        // List the domain validations.
        ListAcmeDomainValidationsResult result = client.listAcmeDomainValidations(req);
        System.out.println(result);
    }
}
```

Listing ACME endpoints

The following example shows how to use the ListAcmeEndpoints function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.ListAcmeEndpointsRequest;
import com.amazonaws.services.certificatemanager.model.ListAcmeEndpointsResult;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        ListAcmeEndpointsRequest req = new ListAcmeEndpointsRequest()
            .withMaxResults(10);

        // List the ACME endpoints.
        ListAcmeEndpointsResult result = client.listAcmeEndpoints(req);
        System.out.println(result);
    }
}
```

Listing ACME external account bindings

The following example shows how to use the ListAcmeExternalAccountBindings function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.ListAcmeExternalAccountBindingsRequest;
import
 com.amazonaws.services.certificatemanager.model.ListAcmeExternalAccountBindingsResult;

public class AWSCertificateManagerSample {
```

```
    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        ListAcmeExternalAccountBindingsRequest req = new
 ListAcmeExternalAccountBindingsRequest()
            .withAcmeEndpointArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
example")
            .withMaxResults(10);

        // List the external account bindings.
        ListAcmeExternalAccountBindingsResult result =
 client.listAcmeExternalAccountBindings(req);
        System.out.println(result);
    }
}
```

Listing certificates

The following example shows how to use the ListCertificates function.

> **Note**
>
> By default, `ListCertificates` does not return Amazon N2 certificates. To include them, set `RootHierarchyTypes` on the request. For more information, see Listing and finding.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.ListCertificatesRequest;
import com.amazonaws.services.certificatemanager.model.ListCertificatesResult;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.AmazonClientException;

import java.util.Arrays;
import java.util.List;

/**
 * This sample demonstrates how to use the ListCertificates function in the AWS
 Certificate
 * Manager service.
 *
 * Input parameters:
 *   CertificateStatuses - An array of strings that contains the statuses to use for
 filtering.
 *   MaxItems - The maximum number of certificates to return in the response.
 *   NextToken - Use when paginating results.
 *
 * Output parameters:
 *   CertificateSummaryList - A list of certificates.
 *   NextToken - Use to show additional results when paginating a truncated list.
 *
 */

public class AWSCertificateManagerExample {

   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().getCredentials();
      }
      catch (Exception ex) {
          throw new AmazonClientException("Cannot load the credentials from file.",
 ex);
      }

      // Create a client.
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.US_EAST_1)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
              .build();
```

```
      // Create a request object and set the parameters.
      ListCertificatesRequest req = new ListCertificatesRequest();
      List<String> Statuses = Arrays.asList("ISSUED", "EXPIRED", "PENDING_VALIDATION",
 "FAILED");
      req.setCertificateStatuses(Statuses);
      req.setMaxItems(10);

      // Retrieve the list of certificates.
      ListCertificatesResult result = null;
      try {
         result = client.listCertificates(req);
      }
      catch (Exception ex)
      {
         throw ex;
      }

      // Display the certificate list.
      System.out.println(result);
   }
}
```

The preceding sample creates output similar to the following.

```
{
    CertificateSummaryList: [{
        CertificateArn:
 arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012,
        DomainName: www.example1.com
    },
    {
        CertificateArn:
 arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012,
        DomainName: www.example2.com
    },
    {
        CertificateArn:
 arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012,
        DomainName: www.example3.com
    }]
}
```

Listing certificate tags

The following example shows how to use the ListTagsForCertificate function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.ListTagsForCertificateRequest;
import com.amazonaws.services.certificatemanager.model.ListTagsForCertificateResult;

import com.amazonaws.services.certificatemanager.model.InvalidArnException;
import com.amazonaws.services.certificatemanager.model.ResourceNotFoundException;
import com.amazonaws.AmazonClientException;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.regions.Regions;


/**
 * This sample demonstrates how to use the ListTagsForCertificate function in the AWS
 Certificate
 * Manager service.
 *
 * Input parameter:
 *   CertificateArn - The ARN of the certificate whose tags you want to list.
 *
*/

public class AWSCertificateManagerExample {

   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().getCredentials();
      }
      catch (Exception ex) {
```

```
            throw new AmazonClientException("Cannot load your credentials from file.",
 ex);
        }

        // Create a client.
        AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
                .withRegion(Regions.US_EAST_1)
                .withCredentials(new AWSStaticCredentialsProvider(credentials))
                .build();

        // Create a request object and specify the ARN of the certificate.
        ListTagsForCertificateRequest req = new ListTagsForCertificateRequest();

 req.setCertificateArn("arn:aws:acm:region:account:certificate/
12345678-1234-1234-1234-123456789012");

        // Create a result object.
        ListTagsForCertificateResult result = null;
        try {
           result = client.listTagsForCertificate(req);
        }
        catch(InvalidArnException ex) {
           throw ex;
        }
        catch(ResourceNotFoundException ex) {
           throw ex;
        }

        // Display the result.
        System.out.println(result);

    }
}
```

The preceding sample creates output similar to the following.

```
{Tags: [{Key: Purpose,Value: Test}, {Key: Short_Name,Value: My_Cert}]}
```

Listing tags for a resource

The following example shows how to use the ListTagsForResource function. This API supports all ACM resource types except the `certificate` resource type. For certificate resources, use ListTagsForCertificate.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.ListTagsForResourceRequest;
import com.amazonaws.services.certificatemanager.model.ListTagsForResourceResult;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create a request object.
        ListTagsForResourceRequest req = new ListTagsForResourceRequest()
            .withResourceArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
abc123");

        // List the tags.
        ListTagsForResourceResult result = client.listTagsForResource(req);
        System.out.println(result);
    }
}
```

Removing tags from a certificate

The following example shows how to use the RemoveTagsFromCertificate function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.RemoveTagsFromCertificateRequest;
```

```
import com.amazonaws.services.certificatemanager.model.RemoveTagsFromCertificateResult;
import com.amazonaws.services.certificatemanager.model.Tag;

import com.amazonaws.services.certificatemanager.model.InvalidArnException;
import com.amazonaws.services.certificatemanager.model.InvalidTagException;
import com.amazonaws.services.certificatemanager.model.ResourceNotFoundException;
import com.amazonaws.AmazonClientException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;

import java.util.ArrayList;

/**
 * This sample demonstrates how to use the RemoveTagsFromCertificate function in the
 AWS Certificate
 * Manager service.
 *
 * Input parameters:
 *   CertificateArn - The ARN of the certificate from which you want to remove one or
 more tags.
 *   Tags - A collection of key-value pairs that specify which tags to remove.
 *
*/

public class AWSCertificateManagerExample {

   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().getCredentials();
      }
      catch (Exception ex) {
          throw new AmazonClientException("Cannot load your credentials from file.",
 ex);
      }

      // Create a client.
```

```
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.US_EAST_1)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
              .build();

      // Specify the tags to remove.
      Tag tag1 = new Tag();
      tag1.setKey("Short_Name");
      tag1.setValue("My_Cert");

      Tag tag2 = new Tag()
            .withKey("Purpose")
            .withValue("Test");

      // Add the tags to a collection.
      ArrayList<Tag> tags = new ArrayList<Tag>();
      tags.add(tag1);
      tags.add(tag2);

      // Create a request object.
      RemoveTagsFromCertificateRequest req = new RemoveTagsFromCertificateRequest();

 req.setCertificateArn("arn:aws:acm:region:account:certificate/
12345678-1234-1234-1234-123456789012");
      req.setTags(tags);

      // Create a result object.
      RemoveTagsFromCertificateResult result = null;
      try {
         result = client.removeTagsFromCertificate(req);
      }
      catch(InvalidArnException ex)
      {
         throw ex;
      }
      catch(InvalidTagException ex)
      {
         throw ex;
      }
      catch(ResourceNotFoundException ex)
      {
         throw ex;
      }
```

```
        // Display the result.
        System.out.println(result);
    }
}
```

Renewing a certificate

The following example shows how to use the RenewCertificate function. The function renews a private certificate issued by a private certificate authority (CA) and exported with the ExportCertificate function. At this time, only exported private certificates can be renewed with this function. In order to renew your AWS Private CA certificates with ACM, you must first grant the ACM service principal permissions to do so. For more information, see Assigning Certificate Renewal Permissions to ACM.

```
package com.amazonaws.samples;

import com.amazonaws.AmazonClientException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;

import com.amazonaws.services.certificatemanager.model.RenewCertificateRequest;
import com.amazonaws.services.certificatemanager.model.RenewCertificateResult;


import com.amazonaws.services.certificatemanager.model.InvalidArnException;
import com.amazonaws.services.certificatemanager.model.ResourceNotFoundException;
import com.amazonaws.services.certificatemanager.model.ValidationException;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

public class RenewCertificate {
```

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

     // Create a client.
     AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
             .withRegion(Regions.your_region)
             .withCredentials(new AWSStaticCredentialsProvider(credentials))
             .build();


     // Create a request object and specify the ARN of the certificate to renew.
     RenewCertificateRequest req = new RenewCertificateRequest();
     req.withCertificateArn("arn:aws:acm:region:account:"
           +"certificate/M12345678-1234-1234-1234-123456789012");


     // Renew the certificate.
     RenewCertificateResult result = null;
     try {
        result = client.renewCertificate(req);
     }
     catch(InvalidArnException ex)
     {
        throw ex;
     }
     catch (ResourceNotFoundException ex)
     {
        throw ex;
     }
     catch (ValidationException ex)
     {
        throw ex;
     }
```

```
      // Display the result.
     System.out.println(result);
   }
}
```

Requesting a certificate

The following example shows how to use the RequestCertificate function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.RequestCertificateRequest;
import com.amazonaws.services.certificatemanager.model.RequestCertificateResult;

import
 com.amazonaws.services.certificatemanager.model.InvalidDomainValidationOptionsException;
import com.amazonaws.services.certificatemanager.model.LimitExceededException;
import com.amazonaws.AmazonClientException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;

import java.util.ArrayList;

/**
 * This sample demonstrates how to use the RequestCertificate function in the AWS
 Certificate
 * Manager service.
 *
 * Input parameters:
 *   DomainName - FQDN of your site.
 *   DomainValidationOptions - Domain name for email validation.
 *   IdempotencyToken - Distinguishes between calls to RequestCertificate.
 *   SubjectAlternativeNames - Additional FQDNs for the subject alternative names
 extension.
 *
 * Output parameter:
 *   Certificate ARN - The Amazon Resource Name (ARN) of the certificate you requested.
```

```
 *
*/

public class AWSCertificateManagerExample {

   public static void main(String[] args) {

      // 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().getCredentials();
      }
      catch (Exception ex) {
          throw new AmazonClientException("Cannot load your credentials from file.",
 ex);
      }

      // Create a client.
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.US_EAST_1)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
              .build();

      // Specify a SAN.
      ArrayList<String> san = new ArrayList<String>();
      san.add("www.example.com");

      // Create a request object and set the input parameters.
      RequestCertificateRequest req = new RequestCertificateRequest();
      req.setDomainName("example.com");
      req.setIdempotencyToken("1Aq25pTy");
      req.setSubjectAlternativeNames(san);

      // Create a result object and display the certificate ARN.
      RequestCertificateResult result = null;
      try {
         result = client.requestCertificate(req);
      }
      catch(InvalidDomainValidationOptionsException ex)
      {
         throw ex;
      }
```

```
        catch(LimitExceededException ex)
        {
           throw ex;
        }

        // Display the ARN.
        System.out.println(result);

    }

}
```

The preceding sample creates output similar to the following.

```
{CertificateArn:
 arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012}
```

Resending validation email

> **Important**
>
> Throughout 2027, AWS Certificate Manager will deprecate email validation for new public certificates and renewals. Use DNS validation instead. For more information, see AWS Certificate Manager will discontinue email validation to prove domain validation for certificates on the AWS Security Blog.

The following example shows you how to use the ResendValidationEmail function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.ResendValidationEmailRequest;
import com.amazonaws.services.certificatemanager.model.ResendValidationEmailResult;

import
 com.amazonaws.services.certificatemanager.model.InvalidDomainValidationOptionsException;
import com.amazonaws.services.certificatemanager.model.ResourceNotFoundException;
import com.amazonaws.services.certificatemanager.model.InvalidStateException;
import com.amazonaws.services.certificatemanager.model.InvalidArnException;
```

```
import com.amazonaws.AmazonClientException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;

/**
 * This sample demonstrates how to use the ResendValidationEmail function in the AWS
 Certificate
 * Manager service.
 *
 * Input parameters:
 *   CertificateArn - Amazon Resource Name (ARN) of the certificate request.
 *   Domain - FQDN in the certificate request.
 *   ValidationDomain - The base validation domain that is used to send email.
 *
*/

public class AWSCertificateManagerExample {

   public static void main(String[] args) {

      // 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().getCredentials();
      }
      catch (Exception ex) {
          throw new AmazonClientException("Cannot load your credentials from file.",
 ex);
      }

      // Create a client.
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.US_EAST_1)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
              .build();

      // Create a request object and set the input parameters.
      ResendValidationEmailRequest req = new ResendValidationEmailRequest();
```

```
 req.setCertificateArn("arn:aws:acm:region:account:certificate/
12345678-1234-1234-1234-123456789012");
      req.setDomain("gregpe.io");
      req.setValidationDomain("gregpe.io");

      // Create a result object.
      ResendValidationEmailResult result = null;
      try {
         result = client.resendValidationEmail(req);
      }
      catch(ResourceNotFoundException ex)
      {
         throw ex;
      }
      catch (InvalidStateException ex)
      {
         throw ex;
      }
      catch (InvalidArnException ex)
      {
         throw ex;
      }
      catch (InvalidDomainValidationOptionsException ex)
      {
         throw ex;
      }

      // Display the result.
      System.out.println(result.toString());

   }
}
```

The preceding sample resends your validation email and displays an empty set.

Revoking an ACME account

The following example shows how to use the RevokeAcmeAccount function.

```
package com.amazonaws.samples;
```

```
import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.RevokeAcmeAccountRequest;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        RevokeAcmeAccountRequest req = new RevokeAcmeAccountRequest()
            .withAcmeEndpointArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
example")
            .withAccountUrl("https://acme.example.com/acme/acct/12345");

        // Revoke the ACME account.
        client.revokeAcmeAccount(req);
    }
}
```

Revoking an ACME external account binding

The following example shows how to use the RevokeAcmeExternalAccountBinding function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.RevokeAcmeExternalAccountBindingRequest;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Create the request.
        RevokeAcmeExternalAccountBindingRequest req = new
 RevokeAcmeExternalAccountBindingRequest()
```

```
            .withAcmeExternalAccountBindingArn("arn:aws:acm:us-
east-1:123456789012:acme-endpoint/ep-example/acme-external-account-binding/eab-
example");

        // Revoke the external account binding.
        client.revokeAcmeExternalAccountBinding(req);
    }
}
```

Searching certificates

The following Java example shows how to use the SearchCertificates function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.SearchCertificatesRequest;
import com.amazonaws.services.certificatemanager.model.SearchCertificatesResponse;
import com.amazonaws.services.certificatemanager.model.CertificateFilterStatement;
import com.amazonaws.services.certificatemanager.model.CertificateFilter;
import com.amazonaws.services.certificatemanager.model.AcmCertificateMetadataFilter;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;

import com.amazonaws.AmazonClientException;

/**
 * This sample demonstrates how to use the SearchCertificates function in the AWS
 Certificate
 * Manager service.
 *
 * Input parameters:
 *   FilterStatement - Optional filter to narrow search results.
 *   MaxResults - The maximum number of certificates to return in the response.
 *   NextToken - Use when paginating results.
 *   SortBy - The field to sort results by (default: CREATED_AT).
 *   SortOrder - The sort order (default: ASCENDING).
 *
 * Output parameters:
```

```
 *   Results - A list of certificate search results.
 *   NextToken - Use to show additional results when paginating a truncated list.
 *
 */

public class AWSCertificateManagerExample {

   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().getCredentials();
      }
      catch (Exception ex) {
          throw new AmazonClientException("Cannot load the credentials from file.",
 ex);
      }

      // Create a client.
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.US_EAST_1)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
              .build();

      // Create a request object and set the parameters.
      SearchCertificatesRequest req = new SearchCertificatesRequest();
      req.setMaxResults(10);

      // Optional: Filter by certificate status
      CertificateFilterStatement filter = CertificateFilterStatement.builder()
              .withFilter(CertificateFilter.builder()

 .withAcmCertificateMetadataFilter(AcmCertificateMetadataFilter.builder()
                              .withStatus("ISSUED")
                              .build())
                      .build())
              .build();
      req.setFilterStatement(filter);

      // Search for certificates.
      SearchCertificatesResponse result = null;
```

```
      try {
         result = client.searchCertificates(req);
      }
      catch (Exception ex)
      {
         throw ex;
      }

      // Display the certificate list.
      System.out.println(result);
   }
}
```

The preceding sample creates output similar to the following.

```
{
    Results: [{
        CertificateArn:
 arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012,
        X509Attributes: {
            Issuer: {
                CommonName: Example CA,
                Country: US,
                Organization: Example Corp
            },
            Subject: {
                CommonName: www.example1.com
            },
            ExtendedKeyUsages: [TLS_WEB_SERVER_AUTHENTICATION,
 TLS_WEB_CLIENT_AUTHENTICATION],
            KeyAlgorithm: RSA_2048,
            KeyUsages: [DIGITAL_SIGNATURE, KEY_ENCIPHERMENT],
            SerialNumber: serial_number,
            NotAfter: 2025-02-14T23:59:59+00:00,
            NotBefore: 2024-01-15T00:00:00+00:00
        },
        CertificateMetadata: {
            AcmCertificateMetadata: {
                CreatedAt: 2024-01-15T12:00:00+00:00,
                IssuedAt: 2024-01-15T12:05:00+00:00,
                Exported: false,
                InUse: true,
                RenewalEligibility: ELIGIBLE,
```

```
                Status: ISSUED,
                Type: AMAZON_ISSUED,
                ValidationMethod: DNS
            }
        }
    },
    {
        CertificateArn:
 arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789013,
        X509Attributes: {
            Issuer: {
                CommonName: Example CA,
                Country: US,
                Organization: Example Corp
            },
            Subject: {
                CommonName: www.example2.com
            },
            ExtendedKeyUsages: [TLS_WEB_SERVER_AUTHENTICATION],
            KeyAlgorithm: EC_prime256v1,
            KeyUsages: [DIGITAL_SIGNATURE],
            SerialNumber: serial_number,
            NotAfter: 2026-06-30T23:59:59+00:00,
            NotBefore: 2025-01-01T00:00:00+00:00
        },
        CertificateMetadata: {
            AcmCertificateMetadata: {
                CreatedAt: 2025-01-01T10:00:00+00:00,
                ImportedAt: 2025-01-01T10:00:00+00:00,
                Exported: false,
                InUse: false,
                RenewalEligibility: INELIGIBLE,
                Status: ISSUED,
                Type: IMPORTED
            }
        }
    }]
}
```

For an Amazon N2 certificate, `AcmCertificateMetadata` also contains `RootHierarchyType` and `CertificateUsage`.

Updating a certificate's validation method

The following example shows how to use the UpdateCertificateOptions function with the `ValidationMethod` option to migrate an existing email-validated public certificate to DNS validation. The certificate ARN is preserved during migration. After you call this function, retrieve the CNAME records by calling the ListCertificateDomainValidations function and add the records to your DNS configuration within 72 hours.

```
package com.amazonaws.samples;

import com.amazonaws.AmazonClientException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.regions.Regions;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;

import com.amazonaws.services.certificatemanager.model.CertificateOptions;
import com.amazonaws.services.certificatemanager.model.UpdateCertificateOptionsRequest;
import com.amazonaws.services.certificatemanager.model.UpdateCertificateOptionsResult;

import com.amazonaws.services.certificatemanager.model.ConflictException;
import com.amazonaws.services.certificatemanager.model.InvalidArnException;
import com.amazonaws.services.certificatemanager.model.InvalidStateException;
import com.amazonaws.services.certificatemanager.model.LimitExceededException;
import com.amazonaws.services.certificatemanager.model.ResourceNotFoundException;
import com.amazonaws.services.certificatemanager.model.ValidationException;

public class UpdateCertificateOptions {

   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 in Linux.
      AWSCredentials credentials = null;
      try {
          credentials = new ProfileCredentialsProvider().getCredentials();
      }
```

```
      catch (Exception ex) {
          throw new AmazonClientException("Cannot load your credentials from file.",
ex);
      }

      // Create a client.
      AWSCertificateManager client = AWSCertificateManagerClientBuilder.standard()
              .withRegion(Regions.your_region)
              .withCredentials(new AWSStaticCredentialsProvider(credentials))
              .build();

      // Build a CertificateOptions with ValidationMethod set to DNS, then create the
      // request. The certificate must be an issued public certificate that uses email
      // validation.
      CertificateOptions options = new CertificateOptions()
              .withValidationMethod("DNS");

      UpdateCertificateOptionsRequest req = new UpdateCertificateOptionsRequest()
              .withCertificateArn("arn:aws:acm:region:account:"
                      + "certificate/12345678-1234-1234-1234-123456789012")
              .withOptions(options);

      // Initiate the migration.
      UpdateCertificateOptionsResult result = null;
      try {
         result = client.updateCertificateOptions(req);
      }
      catch (ValidationException ex) {
         // Thrown when the certificate is not an issued public certificate,
         // or when the requested migration path is not supported (for example,
         // DNS to email).
         throw ex;
      }
      catch (ConflictException ex) {
         // Thrown when a migration is already in progress for this certificate.
         throw ex;
      }
      catch (InvalidArnException ex) {
         throw ex;
      }
      catch (InvalidStateException ex) {
         throw ex;
      }
      catch (LimitExceededException ex) {
```

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

        // The response is empty on success. After this call returns, retrieve the CNAME
        // records by calling ListCertificateDomainValidations and add them to your DNS
        // configuration within 72 hours.
        System.out.println("Migration initiated. Retrieve CNAME records with "
                + "ListCertificateDomainValidations.");
    }
}
```

Tagging a resource

The following example shows how to use the TagResource function. This API supports all ACM resource types except the `certificate` resource type. For certificate resources, use AddTagsToCertificate.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.TagResourceRequest;
import com.amazonaws.services.certificatemanager.model.Tag;

import java.util.ArrayList;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Specify the tags to add.
        Tag tag1 = new Tag()
            .withKey("Environment")
            .withValue("Production");

        ArrayList<Tag> tags = new ArrayList<>();
```

```
        tags.add(tag1);

        // Create a request object.
        TagResourceRequest req = new TagResourceRequest()
            .withResourceArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
abc123")
            .withTags(tags);

        // Tag the resource.
        client.tagResource(req);
    }
}
```

Listing domain validation status

The following example shows how to use the ListCertificateDomainValidations function. The function returns a list of domain validation summaries—one for each domain on the certificate. Each summary contains the active validation configuration and (during a migration) the requested validation configuration with the CNAME record to add to your DNS configuration. The example shows how to handle pagination with `NextToken`.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;

import
 com.amazonaws.services.certificatemanager.model.ListCertificateDomainValidationsRequest;
import
 com.amazonaws.services.certificatemanager.model.ListCertificateDomainValidationsResult;
import com.amazonaws.services.certificatemanager.model.DomainValidationSummary;
import com.amazonaws.services.certificatemanager.model.ValidationConfiguration;
import com.amazonaws.services.certificatemanager.model.ValidationChallenge;
import com.amazonaws.services.certificatemanager.model.DnsValidationChallenge;
import com.amazonaws.services.certificatemanager.model.ResourceRecord;

import java.util.Optional;

public class ListCertificateDomainValidations {

   public static void main(String[] args) {
```

```
     // Create a client.
     AWSCertificateManager client =
AWSCertificateManagerClientBuilder.defaultClient();

     String certificateArn =
             "arn:aws:acm:region:account:"
             + "certificate/12345678-1234-1234-1234-123456789012";

     // Page through the results using NextToken.
     String nextToken = null;
     do {
        ListCertificateDomainValidationsRequest req = new
ListCertificateDomainValidationsRequest()
                .withCertificateArn(certificateArn)
                .withMaxItems(100)
                .withNextToken(nextToken);

        ListCertificateDomainValidationsResult result =
client.listCertificateDomainValidations(req);

        // Print the validation status for each domain.
        for (DomainValidationSummary summary :
result.getDomainValidationSummaryList()) {
           System.out.println("Domain: " + summary.getDomainName());

           Optional.ofNullable(summary.getActiveValidationConfiguration())
              .ifPresent(active -> System.out.println("  Active method: "
                     + active.getValidationMethod()
                     + " (status: " + active.getValidationStatus() + ")"));

           Optional.ofNullable(summary.getRequestedValidationConfiguration())
              .ifPresent(requested -> {
                 System.out.println("  Requested method: " +
requested.getValidationMethod()
                        + " (status: " + requested.getValidationStatus() + ")");

                 // For DNS migration, print the CNAME record to add to DNS.
                 Optional.ofNullable(requested.getValidationChallenge())
                    .map(ValidationChallenge::getDnsValidationChallenge)
                    .map(DnsValidationChallenge::getResourceRecord)
                    .ifPresent(record -> System.out.println("  Add CNAME: "
                           + record.getName() + " -> " + record.getValue()));
              });
        }
```

```
         nextToken = result.getNextToken();
      } while (nextToken != null);
   }
}
```

Removing tags from a resource

The following example shows how to use the UntagResource function. This API supports all ACM resource types except the `certificate` resource type. For certificate resources, use RemoveTagsFromCertificate.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.UntagResourceRequest;

import java.util.ArrayList;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Specify the tag keys to remove.
        ArrayList<String> tagKeys = new ArrayList<>();
        tagKeys.add("Environment");

        // Create a request object.
        UntagResourceRequest req = new UntagResourceRequest()
            .withResourceArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
abc123")
            .withTagKeys(tagKeys);

        // Remove the tags.
        client.untagResource(req);
    }
}
```

Updating an ACME domain validation

The following example shows how to use the UpdateAcmeDomainValidation function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import
 com.amazonaws.services.certificatemanager.model.UpdateAcmeDomainValidationRequest;
import com.amazonaws.services.certificatemanager.model.PrevalidationOptions;
import com.amazonaws.services.certificatemanager.model.DnsPrevalidationOptions;
import com.amazonaws.services.certificatemanager.model.DomainScope;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Configure updated domain scope.
        DomainScope domainScope = new DomainScope()
            .withExactDomain("ENABLED")
            .withSubdomains("DISABLED")
            .withWildcards("DISABLED");

        DnsPrevalidationOptions dnsOptions = new DnsPrevalidationOptions()
            .withDomainScope(domainScope);

        PrevalidationOptions prevalidationOptions = new PrevalidationOptions()
            .withDnsPrevalidation(dnsOptions);

        // Create the request.
        UpdateAcmeDomainValidationRequest req = new UpdateAcmeDomainValidationRequest()
            .withAcmeDomainValidationArn("arn:aws:acm:us-east-1:123456789012:acme-
endpoint/ep-example/acme-domain-validation/dv-example")
            .withPrevalidationOptions(prevalidationOptions);

        // Update the domain validation.
        client.updateAcmeDomainValidation(req);
    }
}
```

Updating an ACME endpoint

The following example shows how to use the UpdateAcmeEndpoint function.

```
package com.amazonaws.samples;

import com.amazonaws.services.certificatemanager.AWSCertificateManagerClientBuilder;
import com.amazonaws.services.certificatemanager.AWSCertificateManager;
import com.amazonaws.services.certificatemanager.model.UpdateAcmeEndpointRequest;
import com.amazonaws.services.certificatemanager.model.CertificateAuthority;
import com.amazonaws.services.certificatemanager.model.PublicCertificateAuthority;

import java.util.Arrays;

public class AWSCertificateManagerSample {

    public static void main(String[] args) {

        AWSCertificateManager client =
 AWSCertificateManagerClientBuilder.defaultClient();

        // Configure an updated certificate authority.
        PublicCertificateAuthority publicCA = new PublicCertificateAuthority()
            .withAllowedKeyAlgorithms(Arrays.asList("RSA_2048", "EC_prime256v1",
 "EC_secp384r1"));

        CertificateAuthority ca = new CertificateAuthority()
            .withPublicCertificateAuthority(publicCA);

        // Create the request.
        UpdateAcmeEndpointRequest req = new UpdateAcmeEndpointRequest()
            .withAcmeEndpointArn("arn:aws:acm:us-east-1:123456789012:acme-endpoint/ep-
example")
            .withContact("NOT_REQUIRED")
            .withCertificateAuthority(ca);

        // Update the ACME endpoint.
        client.updateAcmeEndpoint(req);
    }
}
```

Troubleshoot issues with AWS Certificate Manager

Consult the following topics if you encounter problems using AWS Certificate Manager.

> **Note**
>
> If you don't see your issue addressed in this section, we recommend visiting the AWS Knowledge Center.

Topics

- Troubleshoot certificate requests
- Troubleshoot certificate validation
- Troubleshoot managed certificate renewal
- Troubleshoot ACME certificate automation
- Handling exceptions
- Troubleshoot other problems

Troubleshoot certificate requests

Consult the following topics if you encounter problems when requesting an ACM certificate.

Topics

- Certificate request times out
- Certificate request fails

Certificate request times out

Requests for ACM certificates time out if they are not validated within 72 hours. To correct this condition, open the console, find the record for the certificate, click the checkbox for it, choose **Actions**, and choose **Delete**. Then choose **Actions** and **Request a certificate** to begin again. For more information, see AWS Certificate Manager DNS validation or AWS Certificate Manager email validation. We recommend that you use DNS validation if possible.

Certificate request fails

If your request fails ACM and you receive one of the following error messages, follow the suggested steps to fix the problem. You cannot resubmit a failed certificate request – after resolving the problem, submit a new request.

Topics

- Error message: No Available Contacts
- Error message: Additional Verification Required
- Error message: Invalid Public Domain
- Error message: Other

Error message: No Available Contacts

You chose email validation when requesting a certificate, but ACM could not find an email address to use for validating one or more of the domain names in the request. To correct this problem, you can do one of the following:

- Ensure your domain is configured to receive email. Your domain's name server must have a mail exchanger record (MX record) so ACM's email servers know where to send the domain validation email.

Accomplishing just one of the preceding tasks is enough to correct this problem; you don't need to do both. After you correct the problem, request a new certificate.

For more information about how to ensure that you receive domain validation emails from ACM, see AWS Certificate Manager email validation or Not receiving validation email. If you follow these steps and continue to get the **No Available Contacts** message, then report this to AWS so that we can investigate it.

Error message: Additional Verification Required

ACM requires additional information to process this certificate request. This happens as a fraud-protection measure if your domain ranks within the Alexa top 1000 websites. To provide the required information, use the Support Center to contact Support. If you don't have a support plan, post a new thread in the ACM Discussion Forum.

> **Note**
>
> You cannot request a certificate for Amazon-owned domain names such as those ending in amazonaws.com, cloudfront.net, or elasticbeanstalk.com.

Error message: Invalid Public Domain

One or more of the domain names in the certificate request is not valid. Typically, this is because a domain name in the request is not a valid top-level domain. Try again to request a certificate, correcting any spelling errors or typos that were in the failed request, and ensure that all domain names in the request are for valid top-level domains. For example, you cannot request an ACM certificate for `example.invalidpublicdomain` because "**invalidpublicdomain**" is not a valid top-level domain. If you continue to receive this failure reason, contact the Support Center. If you don't have a support plan, post a new thread in the ACM Discussion Forum.

Error message: Other

Typically, this failure occurs when there is a typographical error in one or more of the domain names in the certificate request. Try again to request a certificate, correcting any spelling errors or typos that were in the failed request. If you continue to receive this failure message, use the Support Center to contact Support. If you don't have a support plan, post a new thread in the ACM Discussion Forum.

Troubleshoot certificate validation

If the ACM certificate request status is **Pending validation**, the request is waiting for action from you. If you chose email validation when you made the request, you or an authorized representative must respond to the validation email messages. These messages were sent to the common email addresses for the requested domain. For more information, see AWS Certificate Manager email validation. If you chose DNS validation, you must write the CNAME record that ACM created for you to your DNS database. For more information, see AWS Certificate Manager DNS validation.

> **Important**
>
> You must validate that you own or control every domain name that you included in your certificate request. If you chose email validation, you will receive validation email messages

for each domain. If you do not, then see Not receiving validation email. If you chose DNS validation, you must create one CNAME record for each domain.

Note

To automate public certificate issuance and renewal outside integrated services such as on Amazon EC2 instances, use the ACME protocol. For more information, see ACME certificate automation. Alternatively, if you cannot use ACME, you can automate exportable certificates issued from ACM through AWS Workload Credentials Provider.

We recommend that you use DNS validation rather than email validation.

Consult the following topics if you experience validation problems.

Topics

- Troubleshoot DNS validation problems
- Troubleshoot email validation problems
- Troubleshooting HTTP validation problems

Troubleshoot DNS validation problems

Consult the following guidance if you are having trouble validating a certificate with DNS.

The first step in DNS troubleshooting is to check the current status of your domain with tools such as the following:

- **dig** — Linux, Windows
- **nslookup** — Linux, Windows

Topics

- Underscores prohibited by DNS provider
- Default trailing period added by DNS provider
- DNS validation on GoDaddy fails
- ACM Console does not display "Create records in Route 53" button

- Route 53 validation fails on private (untrusted) domains
- Validation is successful but issuance or renewal fails
- Validation fails for DNS server on a VPN

Underscores prohibited by DNS provider

If your DNS provider prohibits leading underscores in CNAME values, you can remove the underscore from the ACM-provided value and validate your domain without it. For example, the CNAME value `_x2.acm-validations.aws` can be changed to `x2.acm-validations.aws` for validation purposes. However, the CNAME name parameter must always begin with a leading underscore.

You can use either of the values on the right side of the table below to validate a domain.

Name	Type	Value
`_<random value>.example.com.`	CNAME	`_<random value>.acm-validations.aws.`
`_<random value>.example.com.`	CNAME	`<random value>.acm-validations.aws.`

Default trailing period added by DNS provider

Some DNS providers add by default a trailing period to the CNAME value that you provide. As a result, adding the period yourself causes an error. For example, "`<random_value>.acm-validations.aws.`" is rejected while "`<random_value>.acm-validations.aws`" is accepted.

DNS validation on GoDaddy fails

DNS validation for domains registered with Godaddy and other registries may fail unless you modify the CNAME values provided by ACM. Taking example.com as the domain name, the issued CNAME record has the following form:

```
NAME: _ho9hv39800vb3examplew3vnewoib3u.example.com. VALUE:
 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws.
```

You can create a CNAME record compatible with GoDaddy by truncating the apex domain (including the period) at the end of the NAME field, as follows:

```
NAME: _ho9hv39800vb3examplew3vnewoib3u VALUE:
 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws.
```

ACM Console does not display "Create records in Route 53" button

If you select Amazon Route 53 as your DNS provider, AWS Certificate Manager can interact directly with it to validate your domain ownership. Under some circumstances, the console's **Create records in Route 53** button may not be available when you expect it. If this happens, check for the following possible causes.

- You are not using Route 53 as your DNS provider.
- You are logged into ACM and Route 53 through different accounts.
- You lack IAM permissions to create records in a zone hosted by Route 53.
- You or someone else has already validated the domain.
- The domain is not publicly addressable.

Route 53 validation fails on private (untrusted) domains

During DNS validation, ACM searches for a CNAME in a publicly hosted zone. When it doesn't find one, it times out after 72 hours with a status of **Validation timed out**. You cannot use it to host DNS records for private domains, including resources in an Amazon VPC private hosted zone, untrusted domains in your private PKI, and self-signed certificates.

AWS does provide support for publicly untrusted domains through the AWS Private CA service.

Validation is successful but issuance or renewal fails

If certificate issuance fails with "Pending validation" even though DNS is correct, check that issuance is not being blocked by a Certification Authority Authorization (CAA) record. For more information, see (Optional) Configure a CAA record.

Validation fails for DNS server on a VPN

If you locate a DNS server on a VPN and ACM fails to validate a certificate against it, check if the server is publicly accessible. Public certificate issuance using ACM DNS validation requires that the domain records be resolvable over the public internet.

Troubleshoot email validation problems

> **Important**
>
> Throughout 2027, AWS Certificate Manager will deprecate email validation for new public certificates and renewals. Use DNS validation instead. For more information, see AWS Certificate Manager will discontinue email validation to prove domain validation for certificates on the AWS Security Blog.

Consult the following guidance if you are having trouble validating a certificate domain with email.

Topics

- Not receiving validation email
- Persistent initial timestamp for email validation
- How to switch to DNS validation
- My email-to-DNS migration timed out

Not receiving validation email

When you request a certificate from ACM and choose email validation, domain validation email is sent to the five common administrative addresses. For more information, see AWS Certificate Manager email validation. If you are experiencing problems receiving validation email, review the suggestions that follow.

Where to look for email

ACM sends validation email messages to your requested domain name. You can also specify a superdomain as a validation domain if you wish to receive these emails at that domain instead. Any subdomain up to the minimal website address is valid, and is used as the domain for the email address as the suffix after @. For example, you can receive an email to admin@example.com if you specify example.com as the validation domain for subdomain.example.com. Review the list of email addresses that are displayed in the ACM console (or returned from the CLI or API) to determine where you should be looking for validation email. To see the list, click the icon next to the domain name in the box labeled **Validation not complete**.

The email is marked as spam

Check your spam folder for the validation email.

GMail automatically sorts your email

If you are using GMail, the validation email may have been automatically sorted into the **Updates** or **Promotions** tabs.

Contact the Support Center

If, after reviewing the preceding guidance, you still don't receive the domain validation email, please visit the Support Center and create a case. If you don't have a support agreement, post a message to the ACM Discussion Forum.

Persistent initial timestamp for email validation

The timestamp of a certificate's first email-validation request persists through later requests for validation renewal. This is not evidence of an error in ACM operations.

How to switch to DNS validation

To switch an existing certificate from email validation to DNS validation, use the `UpdateCertificateOptions` API.

My email-to-DNS migration timed out

If you initiate an email-to-DNS migration, you must add the CNAME records to your DNS configuration within 72 hours. If you don't, the migration request expires. The certificate remains email-validated and continues to work normally.

To resolve a timed-out migration:

1. Make sure the CNAME records returned by `ListCertificateDomainValidations` are correctly configured at your DNS provider. Use a tool such as **dig** or **nslookup** to confirm that the records resolve as expected.
2. Reinitiate the migration by calling `UpdateCertificateOptions` again, or by repeating the procedure in the ACM console. A new 72-hour window begins.
3. Monitor progress with `ListCertificateDomainValidations`. Migration completes when the active validation configuration for each domain shows `DNS`.

For more information, see Migrating from email to DNS validation.

Troubleshooting HTTP validation problems

Consult the following guidance if you're having trouble validating a certificate with HTTP.

The first step in HTTP troubleshooting is to check the current status of your domain with tools such as the following:

- **curl** — Linux and Windows
- **wget** — Linux and Windows

Topics

- Content mismatch between RedirectFrom and RedirectTo locations
- Incorrect CloudFront configuration
- HTTP redirect issues
- Validation timeout

Content mismatch between RedirectFrom and RedirectTo locations

If the content at the `RedirectFrom` location doesn't match the content at the `RedirectTo` location, validation will fail. Ensure that the content is identical for each domain in the certificate.

Incorrect CloudFront configuration

Make sure your CloudFront distribution is correctly configured to serve the validation content. Check that the origin and behavior settings are correct and that the distribution is deployed.

HTTP redirect issues

If you're using a redirect instead of serving the content directly, follow these steps to verify your configuration.

To verify redirect configuration

1. Copy the `RedirectFrom` URL and paste it into your browser's address bar.
2. In a new browser tab, paste the `RedirectTo` URL.

3. Compare the content at both URLs to ensure they match exactly.
4. Verify that the redirect returns a 302 status code.

Validation timeout

HTTP validation may time out if the content isn't available within the expected time frame. To troubleshoot validation issues, follow these steps.

To troubleshoot validation timeout

1. Do one of the following to check which domains are pending validation:
   a. Open the ACM console and view the certificate details page. Look for domains marked as **Pending validation**.
   b. Call the `DescribeCertificate` API operation to view the validation status of each domain.
2. For each pending domain, verify that the validation content is accessible from the internet.

Troubleshoot managed certificate renewal

ACM tries to automatically renew your ACM certificates before they expire so that no action is required from you. Consult the following topics if you have trouble with Managed certificate renewal in AWS Certificate Manager.

Preparing for automatic domain validation

Before ACM can renew your certificates automatically, the following must be true:

- Your certificate must be associated with an AWS service that is integrated with ACM. For information about the resources that ACM supports, see Managed automation with integrated services.
- For email-validated certificates, ACM must be able to reach you at an administrator email address for each domain listed in your certificate. The email addresses that will be tried are listed in AWS Certificate Manager email validation.
- For DNS-validated certificates, make sure that your DNS configuration contains the correct CNAME records as described in AWS Certificate Manager DNS validation.

- For HTTP-validated certificates, make sure that your redirects are configured as described in AWS Certificate Manager HTTP validation.

Handling failures in managed certificate renewal

As the certificate nears expiration (45 days for DNS, 45 for EMAIL and 60 days for Private), ACM attempts to renew the certificate if it meets the eligibility criteria. You might have to take actions for the renewal to succeed. For more information, see Managed certificate renewal in AWS Certificate Manager.

Managed certificate renewal for email-validated certificates

ACM certificates are valid for 198 days. Renewing a certificate requires action by the domain owner. ACM begins sending renewal notices to the email addresses associated with the domain 45 days before expiration. The notifications contain a link that the domain owner can click for renewal. Once all listed domains are validated, ACM issues a renewed certificate with the same ARN.

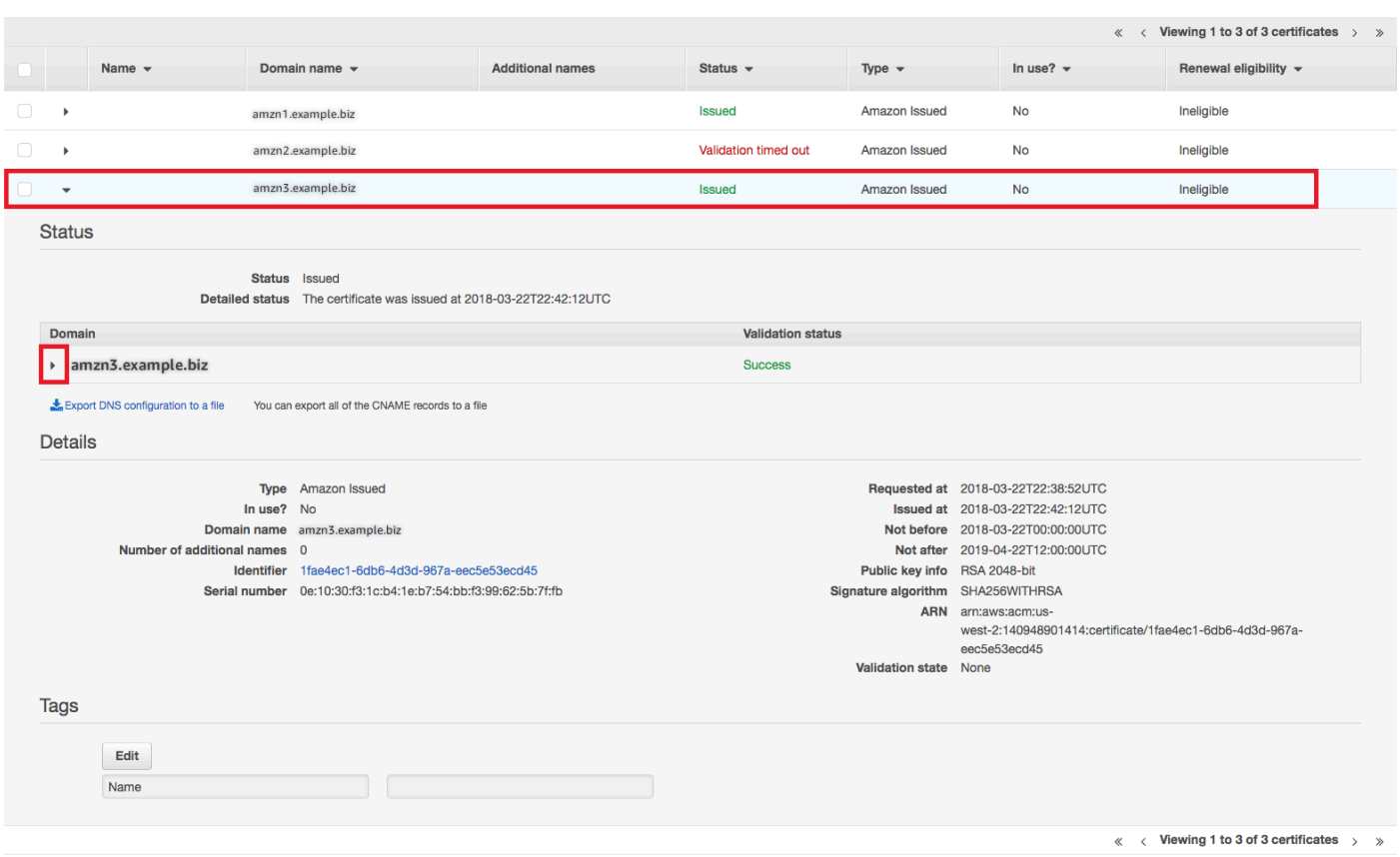
See Validate with Email for instructions on identifying which domains are in the `PENDING_VALIDATION` state and repeating the validation process for those domains.

Managed certificate renewal for DNS-validated certificates

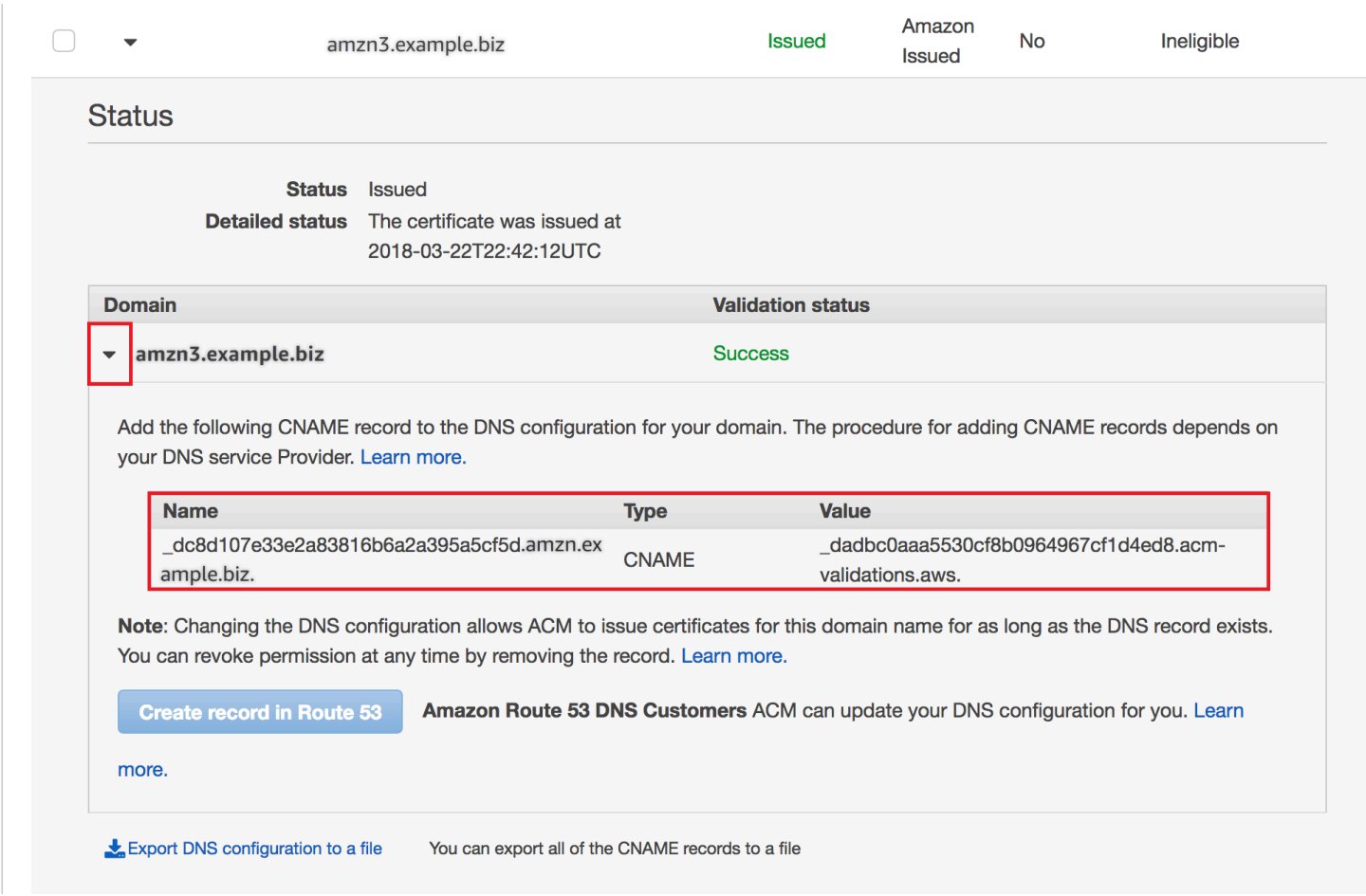
ACM does not attempt TLS validation for DNS-validated certificates. If ACM fails to renew a certificate you validated with DNS validation, it is most likely due to missing or inaccurate CNAME records in your DNS configuration. If this occurs, ACM notifies you that the certificate could not be renewed automatically.

> **Important**
>
> You must insert the correct CNAME records into your DNS database. Consult your domain registrar about how to do this.

You can find the CNAME records for your domains by expanding your certificate and its domain entries in the ACM console. Refer to the figures below for details. You can also retrieve CNAME records by using the DescribeCertificate operation in the ACM API or the describe-certificate command in the ACM CLI. For more information, see AWS Certificate Manager DNS validation.



Choose the target certificate from the console.



Expand the certificate window to find the certificate's CNAME information.

If the problem persists, contact the Support Center.

Managed certificate renewal for HTTP-validated certificates

ACM attempts to renew HTTP-validated certificates automatically. If renewal fails, it's likely due to issues with the HTTP validation records. In such cases, ACM notifies you that the certificate couldn't be renewed automatically.

> **Important**
>
> You must ensure that the content at the `RedirectFrom` location matches the content at the `RedirectTo` location for each domain in the certificate.

You can find the HTTP validation information for your domains by expanding your certificate and its domain entries in the ACM console. You can also retrieve this information using the

DescribeCertificate operation in the ACM API or the describe-certificate command in the ACM CLI. For more information, see AWS Certificate Manager HTTP validation.

If the problem persists, contact the Support Center.

Understanding renewal timing

Managed certificate renewal in AWS Certificate Manager is an asynchronous process. This means that the steps don't occur in immediate succession. After all domain names in an ACM certificate have been validated, there might be a delay before ACM obtains the new certificate. An additional delay can occur between the time when ACM obtains the renewed certificate and the time when that certificate is deployed to the AWS resources that use it. Therefore, changes to the certificate status can take up to several hours to appear in the console.

Troubleshoot ACME certificate automation

This section describes common issues with ACME certificate automation and how to resolve them.

Topics

- ACME failures not appearing in the console Monitoring tab
- Domain validation does not become valid
- Certificate issuance or revocation fails with access denied
- Certificate request is rejected
- Certificate issuance fails with a DNS CNAME error after the domain validation was previously valid
- Account registration fails
- ACME client times out waiting for certificate

ACME failures not appearing in the console Monitoring tab

The **Monitoring** tab on the ACME endpoint details page in the ACM console shows events for the final step of certificate issuance—when ACM creates the certificate (see Monitoring ACME endpoints). If your request failed before that step, it doesn't appear there.

Some failures happen earlier in the process—for example, invalid credentials, an unvalidated domain, or a domain that the endpoint doesn't allow. Your ACME client receives these failures directly.

To diagnose these failures, use one of the following options:

- **Check your ACME client logs**—Your ACME client logs the exact error the server returns. Consult your client's documentation for the location of its log files.
- **Enable CloudTrail data events**—To get a centralized view of all ACME activity in your AWS account for debugging or auditing, enable CloudTrail data event logging for ACM ACME endpoints. For more information about these events, see Data events.

Domain validation does not become valid

An ACME endpoint can issue certificates for a domain only after the domain's validation reaches the `VALID` status. If a domain validation stays in `VALIDATING` or becomes `INVALID`, check the following:

- Confirm that you provisioned the CNAME record exactly as shown in the domain validation details, including both the record name and value. To view the required record, use `DescribeAcmeDomainValidation` or the ACM console.
- If you provided a Route 53 hosted zone for automatic record management, confirm that the hosted zone ID is correct and that ACM has access to it.

`DescribeAcmeDomainValidation` reports a failure reason that indicates the cause:

- **`ACCESS_DENIED`:** ACM could not access the hosted zone to verify or create the record.
- **`DOMAIN_MISMATCH`:** The CNAME record does not match the expected value.
- **`HOSTED_ZONE_NOT_FOUND`:** The specified hosted zone could not be found.
- **`TIMED_OUT`:** The record was not detected within the allowed time. Verify that the record has propagated in DNS.
- **`INTERNAL_FAILURE`:** An internal error occurred. Try again, and if the problem persists, contact AWS Support.

For more information, see ACME domain validation.

Certificate issuance or revocation fails with access denied

ACM uses the IAM role associated with the client's external account binding (EAB) to authorize issuance and revocation. If these operations fail with an access denied error, check the following:

- The role's trust policy allows the ACME service principal (`acm-acme.amazonaws.com`) to perform `sts:AssumeRole`, `sts:TagSession`, and `sts:SetSourceIdentity`. If you added an `sts:SourceIdentity` or `sts:RoleSessionName` condition, confirm that it allows the values ACM uses.
- The role grants `acm:RequestCertificate` for issuance, or `acm:RevokeCertificate` for revocation.
- No AWS Organizations service control policy (SCP) denies the operation. SCPs are enforced at issuance time.

For more information, see IAM for ACME certificate automation.

Certificate request is rejected

If an ACME client's certificate request is rejected, check the following:

- The requested domain is covered by a domain validation in `VALID` status on the endpoint, and the validation's scope (exact domain, subdomains, or wildcards) permits the requested name. For more information, see Domain validation scope.
- The certificate's key algorithm is one of the endpoint's allowed key algorithms. For more information, see Endpoint configuration.

Certificate issuance fails with a DNS CNAME error after the domain validation was previously valid

An ACME domain validation requires that its CNAME record remain in DNS for as long as the validation is in use. If the CNAME is removed after the domain validation reaches `VALID` status, certificate orders for that domain can fail at issuance time, even though the domain validation resource itself was previously confirmed.

When this happens, the ACME client sees the order transition to `invalid` status and the order's `error` field carries a `type` of `urn:ietf:params:acme:error:dns` with a `detail` that names the CNAME records that could not be resolved. For example:

```
{
    "status": "invalid",
    "error": {
```

```
        "type": "urn:ietf:params:acme:error:dns",
        "detail": "DNS CNAME records not found or incorrect:
 [_a1b2c3d4e5f67890abcdef1234567890.example.com.]"
    },
    "identifiers": [{ "type": "dns", "value": "example.com" }],
    "authorizations": ["https://acm-acme-
enroll.region.api.aws/00000000-0000-0000-0000-000000000000/authz/a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111"],
    "finalize": "https://acm-acme-
enroll.region.api.aws/00000000-0000-0000-0000-000000000000/order/a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222/finalize",
    "expires": "2026-06-18T13:49:02Z"
}
```

ACME clients surface the order's error in their own output. For example, Certbot exits with the following error:

```
An unexpected error occurred:
DNS CNAME records not found or incorrect:
 [_a1b2c3d4e5f67890abcdef1234567890.example.com.]
```

The failure also appears in CloudTrail as an `IssueCertificate` management event with a `status` of `FAILED`. Its `serviceEventDetails` carry the ACME error type as `errorType` and its `detail` as `errorMessage`. Management events are logged by default. For more information about `IssueCertificate` events, see Issuing a certificate from an ACME endpoint (IssueCertificate).

To resolve the issue, restore the CNAME record listed in the error. To find the expected CNAME for a domain validation, use `DescribeAcmeDomainValidation` or view the domain validation in the ACM console.

Account registration fails

When an ACME client registers an account with an endpoint, check the following:

- The client provides external account binding (EAB) credentials (the key identifier and HMAC key) during registration. The endpoint requires these credentials.
- If the endpoint requires contact information, the client provides a contact email address during registration. For more information, see External account bindings.

ACME client times out waiting for certificate

Certificate issuance through the ACM ACME endpoint can take up to two minutes. If your ACME client times out before receiving the certificate, increase the client's issuance timeout to at least 120 seconds (2 minutes).

For Certbot, use the `--issuance-timeout` flag:

```
certbot certonly --issuance-timeout 120 ...
```

For other ACME clients, consult your client's documentation for the equivalent timeout configuration.

Handling exceptions

An AWS Certificate Manager command might fail for several reasons. For information about each exception, see the table below.

Private certificate exception handling

The following exceptions can occur when you attempt to renew a private PKI certificate issued by AWS Private CA.

> **Note**
>
> AWS Private CA is not supported in the China (Beijing) Region and the China (Ningxia) Region.

ACM failure code	Comment
`PCA_ACCESS_DENIED`	The private CA has not granted ACM permissions. This triggers a AWS Private CA `AccessDeniedException` failure code. To remedy the problem, grant the necessary permissions to the ACM service principal

ACM failure code	Comment
	using the AWS Private CA CreatePermission operation.
PCA_INVALID_DURATION	The validity period of the requested certificate exceeds the validity period of the issuing private CA. This triggers a AWS Private CA ValidationException failure code. To remedy the problem, install a new CA certificate with an appropriate validity period.
PCA_INVALID_STATE	The private CA being called is not in the correct state to perform the requested ACM operation. This triggers a AWS Private CA InvalidStateException failure code. Resolve the issue as follows: • If the CA has the status CREATING, wait for creation to finish and then install the CA certificate. • If the CA has status PENDING_CERTIFICATE, install the CA certificate. • If the CA has status DISABLED, update it to ACTIVE status. • If the CA has status DELETED, restore it. • If the CA has status EXPIRED, install a new certificate • If the CA has status FAILED, and you cannot resolve the issue, contact Support.

ACM failure code	Comment
PCA_LIMIT_EXCEEDED	The private CA has reached an issuance quota. This triggers a AWS Private CA `LimitExceededException` failure code. Try repeating your request before proceeding with this help. If the error persists, contact Support to request a quota increase.
PCA_REQUEST_FAILED	A network or system error occurred. This triggers a AWS Private CA `RequestFailedException` failure code. Try repeating your request before proceeding with this help. If the error persists, contact Support.
PCA_RESOURCE_NOT_FOUND	The private CA has been permanently deleted. This triggers a AWS Private CA `ResourceNotFoundException` failure code. Verify that you used the correct ARN. If that fails, you won't be able to use this CA. To remedy the problem, create a new CA.
SLR_NOT_FOUND	In order to renew a certificate signed by a private CA that resides in another account, ACM requires a Service Linked Role (SLR) on the account where the certificate resides. If you need to recreate a deleted SLR, see Creating the SLR for ACM.

Troubleshoot other problems

This section includes guidance for problems not related to issuing or validating ACM certificates.

Topics

- Certification Authority Authorization (CAA) problems

- Certificate import problems
- Certificate pinning problems
- API Gateway problems
- What to do when a working certificate fails unexpectedly
- Problems with the ACM service-linked role (SLR)

Certification Authority Authorization (CAA) problems

You can use CAA DNS records to specify that the Amazon certificate authority (CA) can issue ACM certificates for your domain or subdomain. If you receive an error during certificate issuance that says **One or more domain names have failed validation due to a Certification Authority Authorization (CAA) error**, check your CAA DNS records. If you receive this error after your ACM certificate request has been successfully validated, you must update your CAA records and request a certificate again. The **value** field in your CAA record must contain one of the following domain names:

- amazon.com
- amazontrust.com
- awstrust.com
- amazonaws.com

For more information about creating a CAA record, see (Optional) Configure a CAA record.

> **Note**
>
> You can choose to not configure a CAA record for your domain if you do not want to enable CAA checking.

Certificate import problems

You can import third-party certificates into ACM and associate them with integrated services. If you encounter problems, review the prerequisites and certificate format topics. In particular, note the following:

- You can import only X.509 version 3 SSL/TLS certificates.

- Your certificate can be self–signed or it can be signed by a certificate authority (CA).
- If your certificate is signed by a CA, you must include an intermediate certificate chain that provides a path to the root of authority.
- If your certificate is self-signed, you must include the private key in plaintext.
- Each certificate in the chain must directly certify the one preceding.
- Do not include your end-entity certificate in the intermediate certificate chain.
- Your certificate, certificate chain, and private key (if any) must be PEM–encoded. In general, PEM encoding consists of blocks of Base64-encoded ASCII text that begin and end with plaintext header and footer lines. You must not add lines or spaces or make any other changes to a PEM file while copying or uploading it. You can verify certificate chains using the OpenSSL verify utility on the OpenSSL website.
- Your private key (if any) must not be encrypted. (Tip: if it has a passphrase, it's encrypted.)
- Services integrated with ACM must use ACM-supported algorithms and key sizes. See the AWS Certificate Manager User Guide and the documentation for each service to make sure that your certificate will work.
- Certificate support by integrated services might differ depending on whether the certificate is imported into IAM or into ACM.
- The certificate must be valid when it is imported.
- Detail information for all of your certificates is displayed in the console. By default, however, if you call the ListCertificates API or the list-certificates AWS CLI command without specifying the `keyTypes` filter, only `RSA_1024` or `RSA_2048` certificates are displayed. Likewise, Amazon N2 certificates are displayed only when you specify the `RootHierarchyTypes` filter. For more information, see Listing and finding.

Certificate pinning problems

To renew a certificate, ACM generates a new public-private key pair. If your application uses Certificate pinning, sometimes known as SSL pinning, to pin an ACM certificate, the application might not be able to connect to your domain after AWS renews the certificate. For this reason, we recommend that you don't pin an ACM certificate. If your application must pin a certificate, you can do the following:

- Import your own certificate into ACM and then pin your application to the imported certificate. ACM doesn't provide managed renewal for imported certificates.

- If you're using a public certificate or an Amazon N2 certificate, pin your application to the public keys of all of the Amazon root certificates in the root hierarchy that your certificate chains to. If you're using a private certificate, pin your application to the CA's root certificate.

API Gateway problems

When you deploy an *edge-optimized* API endpoint, API Gateway sets up a CloudFront distribution for you. The CloudFront distribution is owned by API Gateway, not by your account. The distribution is bound to the ACM certificate that you used when deploying your API. To remove the binding and allow ACM to delete your certificate, you must remove the API Gateway custom domain that is associated with the certificate.

When you deploy a *regional* API endpoint, API Gateway creates an application load balancer (ALB) on your behalf. The load balancer is owned by API Gateway and is not visible to you. The ALB is bound to the ACM certificate that you used when deploying your API. To remove the binding and allow ACM to delete your certificate, you must remove the API Gateway custom domain that is associated with the certificate.

What to do when a working certificate fails unexpectedly

If you have successfully associated an ACM certificate with an integrated service, but the certificate stops working and the integrated service begins returning errors, the cause may be a change in the permissions that the service needs in order to use an ACM certificate.

For example, Elastic Load Balancing (ELB) requires permission to decrypt an AWS KMS key that, in turn, decrypts the certificate's private key. This permission is granted by a resource-based policy that ACM applies when you associate a certificate with ELB. If ELB loses the grant for that permission, it will fail the next time it attempts to decrypt the certificate key.

To investigate the problem, check the status of your grants in the AWS KMS console. Then take one of the following actions:

- If you believe that permissions granted to an integrated service have been revoked, visit the integrated service's console, disassociate the certificate from the service, then re-associate it. This will reapply the resource-based policy and put a new grant in place.
- If you believe that permissions granted to ACM have been revoked, contact Support at https://console.aws.amazon.com/support/home#/.

Problems with the ACM service-linked role (SLR)

When you issue a certificate signed by a private CA that has been shared with you by another account, ACM attempts on first use to set up a service-linked role (SLR) to interact as a principal with an AWS Private CA resource-based access policy. If you issue a private certificate from a shared CA and the SLR is not in place, ACM will be unable to automatically renew that certificate for you.

ACM might alert you that it cannot determine whether an SLR exists on your account. If the required `iam:GetRole` permission has already been granted to the ACM SLR for your account, then the alert will not recur after the SLR is created. If it does recur, then you or your account administrator might need to grant the `iam:GetRole` permission to ACM, or associate your account with the ACM-managed policy `AWSCertificateManagerFullAccess`.

For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

Quotas

The following AWS Certificate Manager (ACM) service quotas apply to each AWS region per each AWS account.

To see what quotas can be adjusted, see the ACM quotas table in the *AWS General Reference Guide*. To request quota increases, create a case at the Support Center.

General quotas

Item	Default quota
Number of ACM certificates Expired and revoked certificates continue to count toward this total. Certificates signed by a CA from AWS Private CA do not count toward this total. Certificates issued through ACME do not count toward this total. This total includes Amazon N2 certificates issued by ACM.	2500
Number of ACM certificates per year (last 365 days) You can request up to twice your quota of ACM certificates per year, region, and account. For example, if your quota is 2,500, you can request up to 5,000 ACM certificates per year in a given region and account. You can only have 2,500 certificates at any given time. To request 5,000 certificates in a year, you must delete 2,500 during the year to stay within the quota. If you need more than 2,500 certifica	5,000

Item	Default quota
tes at any given time, you must contact the **Support Center**. Certificates signed by a CA from AWS Private CA do not count toward this total. Certificates issued through ACME do not count toward this total. This total includes Amazon N2 certificates issued by ACM.	
Number of imported certificates	2,500
Number of imported certificates per year (last 365 days)	5,000

Item	Default quota
Number of domain names per ACM certificate The default quota is 10 domain names for each ACM certificate. Your quota may be greater. The first domain name that you submit is included as the subject common name (CN) of the certificate. All names are included in the Subject Alternative Name extension. You can request up to 100 domain names. To request an increase to your quota, create a request in the Service Quotas console for the ACM service. Before creating a case, however, make sure you understand how adding more domain names can create more administrative work for you if you use email validation. For more information, see Domain validation. The quota for the number of domain names per ACM certificate applies only to certificates that are provided by ACM. This quota does not apply to certificates that you import into ACM. The following sections apply only to ACM certificates.	10
Number of domain names per ACME-issued certificate Certificates issued through ACME can include up to 100 domain names. This quota is fixed and cannot be increased.	100

Item	Default quota
Number of Private CAs ACM is integrated with AWS Private Certificate Authority (AWS Private CA). You can use the ACM console, AWS CLI, or ACM API to request private certificates from an existing private certificate authority (CA) hosted by AWS Private CA. These certificates are managed within the ACM environment and have the same restrictions as public certificates issued by ACM. For more information, see Request a private certificate in AWS Certificate Manager. You can also issue private certificates by using the standalone AWS Private CA service. For more information, see Issue a Private End-Entity Certificate. A private CA that has been deleted will count towards your quota until the end of its restoration period. For more information, see Deleting Your Private CA.	200
Number of Private Certificates per CA (lifetime)	1,000,000
Number of ACME endpoints The maximum number of ACME endpoints per AWS account per region.	50
Number of external account bindings per ACME endpoint	1,000
Number of domain validations per ACME endpoint	1,500

API rate quotas

The following quotas apply to the ACM API for each region and account. ACM throttles API requests at different quotas depending on the API operation. Throttling means that ACM rejects an otherwise valid request because the request exceeds the operation's quota for the number of requests per second. When a request is throttled, ACM returns a `ThrottlingException` error. The following table lists each API operation and the quota at which ACM throttles requests for that operation.

> **Note**
>
> In addition to the API actions listed in the table below, ACM can also call the external `IssueCertificate` action from AWS Private CA. For up-to-date rate quota information on `IssueCertificate`, see the endpoints and quotas for AWS Private CA.

Requests-per-second quota for each ACM API operation

Per-operation request rate quotas for ACME certificate automation are tiered: read operations at 30 requests per second per account, write operations at 20 requests per second per account, and domain validation mutation operations (Create/Update/Delete AcmeDomainValidation) at 5 requests per second per account.

API call	Requests per second
`AddTagsToCertificate`	5
`CreateAcmeDomainValidation`	5
`CreateAcmeEndpoint`	20
`CreateAcmeExternalAccountBinding`	20
`DeleteAcmeDomainValidation`	5
`DeleteAcmeEndpoint`	20
`DeleteAcmeExternalAccountBinding`	20

API call	Requests per second
DeleteCertificate	10
DescribeAcmeAccount	30
DescribeAcmeDomainValidation	30
DescribeAcmeEndpoint	30
DescribeAcmeExternalAccount Binding	30
DescribeCertificate	10
ExportCertificate	10
GetAccountConfiguration	1
GetAcmeExternalAccountBindi ngCredentials	30
GetCertificate	10
ImportCertificate	1
ListAcmeAccounts	30
ListAcmeDomainValidations	30
ListAcmeEndpoints	30
ListAcmeExternalAccountBindings	30
ListCertificates	8
ListTagsForCertificate	10
ListTagsForResource	5
PutAccountConfiguration	1

API call	Requests per second
RemoveTagsFromCertificate	5
RenewCertificate	5
RequestCertificate	5
ResendValidationEmail	1
RevokeAcmeAccount	20
RevokeAcmeExternalAccountBinding	20
SearchCertificates	5
TagResource	5
UntagResource	5
UpdateAcmeDomainValidation	5
UpdateAcmeEndpoint	20
UpdateCertificateOptions	5

For more information, see AWS Certificate Manager API Reference.

ACME protocol request rate quotas

The following quotas apply to requests that ACME clients send to an ACME endpoint over the ACME protocol. These quotas are separate from the ACM API request rate quotas in the preceding section. Per-operation request rate quotas for the ACME protocol are tiered: read operations at 25 requests per second per account, write operations at 20 requests per second per account, and certificate operations (FinalizeOrder, RevokeCertificate) at 1 request per second per account. When a request exceeds the quota, the endpoint returns a rate limit error.

Requests-per-second quota for each ACME protocol operation

Operation	Requests per second
ChangeAccountKey	20
FinalizeOrder	1
GetCertificate	25
GetDirectory	25
GetOrder	25
ListOrders	25
ManageAccount	20
ManageAuthorization	25
NewAccount	20
NewNonce	25
NewOrder	20
RevokeCertificate	1

Best practices

Best practices are recommendations that can help you use AWS Certificate Manager (ACM) more effectively. The following best practices are based on real-world experience from current ACM customers.

Topics

- Account-level separation
- AWS CloudFormation
- Custom Trust Stores
- Certificate pinning
- Domain validation
- Adding or deleting domain names
- Domain name privacy
- Turn on AWS CloudTrail

Account-level separation

Use account-level separation in your policies to control who can access certificates at an account level. Keep your production certificates in separate accounts than your testing and development certificates. If you can't use account-level separation, you can restrict access to specific roles by denying `kms:CreateGrant` action in your policies. This limits which roles in an account can sign certificates at a high level. For information about grants, including grant terminology, see Grants in AWS KMS in the *AWS Key Management Service Developer Guide*.

If you want more granular control than restricting the use of `kms:CreateGrant` by account, you can limit `kms:CreateGrant` to specific certificates using kms:EncryptionContext condition keys. Specify `arn:aws:acm` as the key, and the value of the ARN to restrict. The following example policy prevents the use of a specific certificate, but allow others.

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement": [
       {
           "Sid": "VisualEditor0",
           "Effect": "Deny",
           "Action": "kms:CreateGrant",
           "Resource": "*",
           "Condition": {
               "StringEquals": {
                   "kms:EncryptionContext:aws:acm:arn": "arn:aws:acm:us-
east-1:111122223333:certificate/b26def74-1234-4321-9876-951d4c07b197"
               }
           }
       }
   ]
}
```

AWS CloudFormation

With AWS CloudFormation you can create a template that describes the AWS resources that you want to use. CloudFormation then provisions and configures those resources for you. CloudFormation can provision resources that are supported by ACM such as Elastic Load Balancing, Amazon CloudFront, and Amazon API Gateway. For more information, see Managed automation with integrated services.

If you use CloudFormation to quickly create and delete multiple test environments, we recommend that you do not create a separate ACM certificate for each environment. Doing so will quickly exhaust your certificate quota. For more information, see Quotas. Instead, create a wildcard certificate that covers all of the domain names that you are using for testing. For example, if you repeatedly create ACM certificates for domain names that vary by only a version number, such as *<version>*`.service.example.com`, create instead a single wildcard certificate for *<*>*`.service.example.com`.

> **Important**
>
> If you're using Amazon CloudFront distributions, note that HTTP validation doesn't support wildcard certificates. When including wildcard certificates in your CloudFormation templates for use with Amazon CloudFront, you must use either DNS validation or email validation. We recommend DNS validation for automated renewal capabilities.

Include the wildcard certificate in the template that CloudFormation uses to create your test environment.

Custom Trust Stores

In order to ensure connectivity to endpoints protected by ACM public certificates, we recommend the Amazon roots be included in your custom trust store. Amazon Root certificate authorities can represent different key types and algorithms. Starfield Services Root Certificate Authority - G2 is an older root that is compatible with other older trust stores and clients that can not be updated. By including all of these root CAs, you'll be able to ensure maximum compatibility for your application.

Amazon N2 certificates chain to different roots. For the roots to include in the trust store of each verifying peer, see Amazon N2 roots and trust.

Certificate pinning

Certificate pinning, sometimes known as SSL pinning, is a process that you can use in your application to validate a remote host by associating that host directly with its X.509 certificate or public key instead of with a certificate hierarchy. The application therefore uses pinning to bypass SSL/TLS certificate chain validation. The typical SSL validation process checks signatures throughout the certificate chain from the root certificate authority (CA) certificate through the subordinate CA certificates, if any. It also checks the certificate for the remote host at the bottom of the hierarchy. Your application can instead pin to the certificate for the remote host to say that *only that* certificate and not the root certificate or any other in the chain is trusted. You can add the remote host's certificate or public key to your application during development. Alternatively, the application can add the certificate or key when it first connects to the host.

> **Warning**
>
> We recommend that your application **not** pin an ACM certificate. ACM performs Managed certificate renewal in AWS Certificate Manager to automatically renew your Amazon-issued SSL/TLS certificates before they expire. To renew a certificate, ACM generates a new public-private key pair. If your application pins the ACM certificate and the certificate is successfully renewed with a new public key, the application might be unable to connect to your domain.

If you decide to pin a certificate, the following options will not hinder your application from connecting to your domain:

- Import your own certificate into ACM and then pin your application to the imported certificate. ACM doesn't try to automatically renew imported certificates.
- If you're using a public certificate or an Amazon N2 certificate, pin your application to the public keys of all of the Amazon root certificates in the root hierarchy that your certificate chains to. If you're using a private certificate, pin your application to the CA's root certificate.

Domain validation

> **Important**
>
> Throughout 2027, AWS Certificate Manager will deprecate email validation for new public certificates and renewals. Use DNS validation instead. For more information, see AWS Certificate Manager will discontinue email validation to prove domain validation for certificates on the AWS Security Blog.

Before the Amazon certificate authority (CA) can issue a certificate for your site, AWS Certificate Manager (ACM) must verify that you own or control all the domains that you specified in your request. You can verify domain ownership using DNS. For more information, see AWS Certificate Manager DNS validation.

Adding or deleting domain names

You cannot add or remove domain names from an existing ACM certificate. Instead you must request a new certificate with the revised list of domain names. For example, if your certificate has five domain names and you want to add four more, you must request a new certificate with all nine domain names. As with any new certificate, you must validate ownership of all the domain names in the request, including the names that you previously validated for the original certificate.

If you use email validation, you receive up to 8 validation email messages for each domain, at least 1 of which must be acted upon within 72 hours. For example, when you request a certificate with five domain names, you receive up to 40 validation messages, at least 5 of which must be acted upon within 72 hours. As the number of domain names in the certificate request increases, so does the work required to use email to validate domain ownership.

If you use DNS validation instead, you must write one new DNS record to the database for the FQDN you want to validate. ACM sends you the record to create and later queries the database to determine whether the record has been added. Adding the record asserts that you own or control the domain. In the preceding example, if you request a certificate with five domain names, you must create five DNS records. We recommend that you use DNS validation when possible.

Domain name privacy

Do not include confidential or sensitive information in public certificate domain names. Public certificates, including ACM public certificates, are logged to public, append-only Certificate Transparency logs. Amazon N2 certificates are not logged, but the domain names in them are still customer data and should be treated as confidential. For more information, see Certificate Transparency Logging.

Turn on AWS CloudTrail

Turn on CloudTrail logging before you begin using ACM. CloudTrail enables you to monitor your AWS deployments by retrieving a history of AWS API calls for your account, including API calls made via the AWS Management Console, the AWS SDKs, the AWS Command Line Interface, and higher-level Amazon Web Services. You can also identify which users and accounts called the ACM APIs, the source IP address the calls were made from, and when the calls occurred. You can integrate CloudTrail into applications using the API, automate trail creation for your organization, check the status of your trails, and control how administrators turn CloudTrail logging on and off. For more information, see Creating a Trail. Go to Using CloudTrail with AWS Certificate Manager to see example trails for ACM actions.

Document history

The following table describes the documentation release history of AWS Certificate Manager beginning in 2018.

Change	Description	Date
Amazon N2 certificates	You can now request Amazon N2 certificates for authentication outside web browsers, such as mutual TLS (mTLS). Use the `RootHierarchyType` and `CertificateUsage` parameters of `RequestCertificate` to request a certificate that authenticates a client, or both a client and a server, from the root hierarchy of your choice. See Amazon N2 certificates in AWS Certificate Manager.	November 19, 2026
AWS Workload Credentials Provider container support	Added support for running the AWS Workload Credentials Provider in containers on Amazon ECS, Amazon EKS, and Docker on Amazon EC2. See AWS Workload Credentials Provider.	November 17, 2026
ACME condition keys	Documented the condition key values ACM sets when an ACME client requests or revokes a certificate, the additional `acm:AddTagsToCertificate`	September 28, 2026

Change	Description	Date
	permission required by an ACME endpoint configured with certificate tags, and the `acm:Export` condition key. See Condition keys for ACME certificate requests.	
Email validation deprecation	AWS Certificate Manager will deprecate email validation throughout 2027 for both new public certificates and renewals. See AWS Certificate Manager email validation for details.	August 17, 2026
Adding best practice for domain name privacy	Advises against including confidential or sensitive information in ACM public certificate domain names.	July 29, 2026
Email-to-DNS validation migration	You can now migrate an existing email-validated public ACM certificate to DNS validation while preserving the certificate ARN. For more information, see Migrating from email to DNS validation.	July 9, 2026

Change	Description	Date
ACME certificate automation	Added support for the Automated Certificate Management Environment (ACME) protocol. You can now create ACME endpoints, configure domain validations, and generate external account bindings to automate public certificate issuance for customer-managed infrastructure using standard ACME clients. See ACME certificate automation.	June 18, 2026
AWS Workload Credentials Provider	AWS announces AWS Workload Credentials Provider, a lightweight client-side provider that automates deployment of exported certificates from ACM across AWS and non-AWS workloads. It runs on Windows and Linux and supports NGINX and Apache web servers. See AWS Workload Credentials Provider.	June 11, 2026

Change	Description	Date
Certificate transparency logging opt-out deprecated	Certificate transparency logging opt-out is no longer available. All public ACM certificates are automatically recorded in certificate transparency logs. The `CertificateTransparencyLoggingPreference` option is deprecated. With this update, ACM issued public certificates will be compliant with upcoming browser policy changes which require that all TLS server authentication certificates issued after June 15, 2026 are logged to at least one Certificate Transparency log.	June 1, 2026
ECDSA Key Usage exception for certificate reimport	ACM now allows re-import of ECDSA certificates without the `keyEncipherment` Key Usage value, in compliance with RFC 5480. For more information, see https://docs.aws.amazon.com/acm/latest/userguide/import-reimport.html.	April 3, 2026
SearchCertificates documentation	Updated certificate management documentation to use the SearchCertificates API for finding and filtering certificates. See Search certificates.	March 31, 2026

Change	Description	Date
Updated public certificate validity period	Public ACM certificates are now valid for 198 days, reduced from 13 months (395 days). With this update, ACM issued public certificates will be compliant with upcoming certificate lifetime requirements from the CA/B Forum. Per the new requirements, public certificates issued after March 15th 2026 must have a maximum validity of 200 days. The renewal window for public certificates has been updated to 45 days before expiration. Private certificates remain valid for 13 months (395 days).	February 18, 2026
Change to re-importing certificates	ACM allows re-import of a certificate into the same ARN only when the ClientAuth EKU is missing from the previous certificate. This accommodates industry changes where certificate authorities no longer issue certificates with ClientAuth EKU to comply with Chrome's root program requirements.	October 22, 2025

Change	Description	Date
Added note about issuing certificates	Added a note to the ACM certificate concept topic detailing changes to ACM certificate issuance with the TLS Web Client Authentic ation extension.	July 23, 2025
Removed reference to authentication extension	Removed the reference to the TLS Web Client Authentic ation extension from the example certificate.	July 3, 2025
AWS Certificate Manager exportable public certificates	You can export ACM public certificates.	June 17, 2025
ACM supports HTTP validatio n with CloudFront	ACM now supports HTTP validation for domain ownership verification when issuing certificates for CloudFront distributions.	April 24, 2025
Deprecation of mail exchanger (MX) email validation	The ACM console no longer supports mail exchanger (MX).	July 11, 2024
Adding best practice around account-level separation	Use account-level separatio n in your policies wherever possible. If not possible, you can restrict permissions at the account level or through encryption context condition keys in your policies.	June 11, 2024

Change	Description	Date
Upcoming deprecation of WHOIS email verification	Added a note about the deprecation of WHOIS email verification starting in June 2024.	February 5, 2024
Condition key support added	Added support for IAM Condition keys when requesting ACM certificates. For a list of supported conditions, see https://docs.aws.amazon.com/acm/latest/userguide/acm-conditions.html#acm-conditions-supported.	August 24, 2023
ECDSA support added	Added support for Elliptic Curve Digital Signature Algorithm (ECDSA) when requesting a public ACM certificate. For a list of supported key algorithms, see https://docs.aws.amazon.com/acm/latest/userguide/acm-certificate.html#algorithms.	November 8, 2022
New CloudWatch Events	Added ACM Certificate Expired, ACM Certificate Available, and ACM Certificate Renewal Action Required events. For a list of supported CloudWatch Events, see https://docs.aws.amazon.com/acm/latest/userguide/cloudwatch-events.html.	October 27, 2022

Change	Description	Date
Updating key algorithm types for import	Certificates imported into ACM may now have keys with additional RSA and Elliptic Curve algorithms. For a list of currently supported key algorithms, see https://docs.aws.amazon.com/acm/latest/userguide/import-certificate-prerequisites.html.	July 14, 2021
Promoting "Monitoring and Logging" as a separate chapter	Moved monitoring and logging documentation to its own chapter. This change covers CloudWatch Metrics, CloudWatch Events/EventBridge, and CloudTrail. For more information, see https://docs.aws.amazon.com/acm/latest/userguide/monitoring-and-logging.html.	March 23, 2021
Added CloudWatch Metrics and Events support	Added DaysToExpiry metric and event and supporting APIs. For more information, see https://docs.aws.amazon.com/acm/latest/userguide/cloudwatch-metrics.html and https://docs.aws.amazon.com/acm/latest/userguide/cloudwatch-events.html.	March 3, 2021

Change	Description	Date
Added cross-account support	Added cross-account support for using private CAs from AWS Private CA. For more information, see https://docs.aws.amazon.com/acm/latest/userguide/ca-access.html.	August 17, 2020
Added region support	Added region support for the AWS China (Beijing and Ningxia) Regions. For a complete list of supported regions, see https://docs.aws.amazon.com/general/latest/gr/rande.html#acm-pca_region.	March 4, 2020
Added renewal workflow testing	Customers can now manually test the configuration of their ACM managed renewal workflow. For more information, see Testing ACM's Managed Renewal Configuration.	March 14, 2019
Certificate transparency logging now default	Added ability to publish ACM public certificates into certificate transparency logs by default.	April 24, 2018

Change	Description	Date
Launching AWS Private CA	Launched ACM Private Certificate Manager (CM), and extension of AWS Certifica te Manager that allows users to establish a secure managed infrastructure for issuing and revoking private digital certificates. For more information, see AWS Private Certificate Authority.	April 4, 2018
Certificate transparency logging	Added certificate transpare ncy logging to Best Practices.	March 27, 2018

The following table describes the documentation release history of AWS Certificate Manager prior to 2018.

Change	Description	Release Date
New content	Added DNS validation to AWS Certificate Manager DNS validation.	November 21, 2017
New content	Added new Java code examples to Use AWS Certificate Manager with the SDK for Java.	October 12, 2017
New content	Added information about CAA records to (Optional) Configure a CAA record.	September 21, 2017
New content	Added information about .IO domains to Troubleshoot	July 07, 2017

Change	Description	Release Date
	issues with AWS Certificate Manager.	
New content	Added information about reimporting a certificate to Reimport a certificate.	July 07, 2017
New content	Added information about certificate pinning to Best practices and to Troubleshoot issues with AWS Certificate Manager.	July 07, 2017
New content	Added CloudFormation to Managed automation with integrated services.	May 27, 2017
Update	Added more information to Quotas.	May 27, 2017
New content	Added documentation about Identity and Access Management for AWS Certificate Manager.	April 28, 2017
Update	Added a graphic to show where validation email is sent. See AWS Certificate Manager email validation.	April 21, 2017
Update	Added information about setting up email for your domain. See AWS Certificate Manager email validation.	April 6, 2017

Change	Description	Release Date
Update	Added information about checking certificate renewal status in the console. See Check a certificate's renewal status.	March 28, 2017
Update	Updated the documenta tion for using Elastic Load Balancing.	March 21, 2017
New content	Added support for AWS Elastic Beanstalk and Amazon API Gateway. See Managed automation with integrated services.	March 21, 2017
Update	Updated the documentation about Managed certificate renewal.	February 20, 2017
New content	Added documentation about Imported certificates.	October 13, 2016
New content	Added AWS CloudTrail support for ACM actions. See Using CloudTrail with AWS Certificate Manager.	March 25, 2016
New guide	This release introduces AWS Certificate Manager.	January 21, 2016