User Guide

AWS Certificate Manager

Version 1.0

AWS Certificate Manager: User Guide

Table of Contents

What is AWS Certificate Manager? 1
Supported Regions 1
Pricing 2
Concepts 2
ACM Certificate 3
ACM Root CAs 5
Apex Domain 5
Asymmetric Key Cryptography 5
Certificate Authority 6
Certificate Transparency Logging 6
Domain Name System 7
Domain Names 7
Encryption and Decryption 8
Fully Qualified Domain Name (FQDN) 9
Hypertext Transfer Protocol (HTTP) 9
Public Key Infrastructure (PKI) 9
Root Certificate 10
Secure Sockets Layer (SSL) 10
Secure HTTPS 10
SSL Server Certificates 10
Symmetric Key Cryptography 11
Transport Layer Security (TLS) 11
Trust 11
What is the right AWS certificate service for my needs? 11
Certificates 12
Set up 13
Sign up for an AWS account 13
Create a user with administrative access 14
Register a domain name 15
(Optional) Configure a CAA record 15
Public certificates 18
Characteristics and limitations 18
Request a public certificate 23
Validate domain ownership 26

Private certificates ... 45
Conditions for use ... 46
Request a private certificate ... 47
Export certificate ... 50
Imported certificates ... 53
Prerequisites ... 54
Certificate format ... 55
Import certificate ... 57
Reimport certificate ... 58
List certificates ... 60
View certificate details ... 63
Delete certificates ... 67
Managed certificate renewal ... 69
Public certificates ... 70
DNS-validated domains ... 71
Email-validated domains ... 71
HTTP-validated domains ... 72
Private certificates ... 73
Automate export of renewed certificates ... 74
Test managed renewal ... 75
Check renewal status ... 76
Check the status (console) ... 77
Check the status (API) ... 78
Check the status (CLI) ... 78
Check the status using Personal Health Dashboard (PHD) ... 78
Tag resources ... 80
Tag restrictions ... 80
Managing tags ... 81
Managing tags (console) ... 81
Managing tags (CLI) ... 83
Manage tags ... 83
Integrated services ... 84
Security ... 89
Data protection ... 89
Security for certificate private keys ... 90
Identity and Access Management ... 91

Audience ... 92
Authenticating with identities ... 92
Managing access using policies ... 96
How AWS Certificate Manager works with IAM ... 98
Identity-based policy examples ... 105
ACM API permissions reference ... 109
AWS managed policies ... 111
Use condition keys ... 114
Use service-linked roles ... 119
Troubleshooting ... 123
Resilience ... 125
Infrastructure security ... 125
Granting programmative access to ACM ... 126
Best practices ... 127
Account-level separation ... 128
AWS CloudFormation ... 129
Certificate pinning ... 129
Domain validation ... 130
Adding or deleting domain names ... 130
Opting out of certificate transparency logging ... 131
Turn on AWS CloudTrail ... 132
Monitor and log ... 134
Amazon EventBridge ... 134
Supported events ... 134
Example actions ... 139
CloudTrail ... 148
Supported API actions ... 149
API calls for integrated services ... 163
CloudWatch metrics ... 168
Use AWS Certificate Manager with the SDK for Java ... 170
AddTagsToCertificate ... 170
DeleteCertificate ... 172
DescribeCertificate ... 174
ExportCertificate ... 177
GetCertificate ... 180
ImportCertificate ... 182

ListCertificates ... 186
RenewCertificate ... 188
ListTagsForCertificate ... 190
RemoveTagsFromCertificate ... 192
RequestCertificate ... 194
ResendValidationEmail ... 197
Troubleshoot ... 200
Certificate requests ... 200
Request times out ... 200
Request fails ... 201
Certificate validation ... 202
DNS validation ... 203
Email validation ... 206
HTTP validation ... 208
Certificate renewal ... 209
Preparing for automatic domain validation ... 209
Handling failures in managed certificate renewal ... 210
Managed certificate renewal for email-validated certificates ... 210
Managed certificate renewal for DNS-validated certificates ... 210
Managed certificate renewal for HTTP-validated certificates ... 212
Understanding renewal timing ... 213
Other problems ... 213
CAA records ... 213
Certificate import ... 214
Certificate pinning ... 215
API Gateway ... 215
Unexpected failure ... 216
Problems with the ACM service-linked role (SLR) ... 216
Handling exceptions ... 216
Private certificate exception handling ... 217
Quotas ... 220
General quotas ... 220
API rate quotas ... 222
Document history ... 225

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**
>
> ACM is not intended for use with a stand-alone web server. If you want to set up a stand-alone secure server on an Amazon EC2 instance, the following tutorial has instructions: Configure SSL/TLS on Amazon Linux 2023.

Topics

- Supported Regions
- Pricing for AWS Certificate Manager
- AWS Certificate Manager concepts
- What is the right AWS certificate service for my needs?

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

- 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

ACM Certificate

ACM generates X.509 version 3 certificates. Each is valid for 13 months (395 days) and contains the following extensions.

- **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.
- **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, TLS Web Client 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. All of the roots are cross-signed by the Starfield Services Root Certificate Authority certificate.

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
```

Certificate transparency logging is automatic when you request or renew a certificate unless you choose to opt out. For more information about opt out, see Opting out of 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 email or DNS-validated certificates, ACM customers 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 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.

What is the right AWS certificate service for my needs?

AWS offers two options to customers deploying managed X.509 certificates. Choose the best one for your needs.

1. **AWS Certificate Manager (ACM)**—This service is for enterprise customers who need a secure web presence using TLS. ACM certificates are deployed through Elastic Load Balancing, Amazon CloudFront, Amazon API Gateway, and other integrated AWS services. The most common application of this kind is a secure public website with significant traffic requirements. ACM also simplifies security management by automating the renewal of expiring certificates. *You are in the right place for this service.*
2. **AWS Private CA**—This service is for enterprise customers building a public key infrastructure (PKI) inside the AWS cloud and intended for private use within an organization. With AWS Private CA, you can create your own certificate authority (CA) hierarchy and issue certificates with it for authenticating users, computers, applications, services, servers, and other devices. Certificates issued by a private CA cannot be used on the internet. For more information, see the AWS Private CA User Guide.

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. 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.

> **Note**
>
> Public ACM certificates can be installed on Amazon EC2 instances that are connected to a Nitro Enclave, but not to other Amazon EC2 instances. For information about setting up a standalone web server on an Amazon EC2 instance not connected to a Nitro Enclave, see Tutorial: Install a LAMP web server on Amazon Linux 2 or Tutorial: Install a LAMP web server with the Amazon Linux AMI.

> **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
- AWS Certificate Manager public certificates
- Private certificates in AWS Certificate Manager
- Import certificates into AWS Certificate Manager
- List certificates managed by AWS Certificate Manager
- View AWS Certificate Manager certificate details

- Delete certificates managed by 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 Services integrated with ACM. The following sections discuss the steps you need to perform before using ACM.

Topics

- Sign up for an AWS account
- Create a user with administrative access
- Register a domain name for ACM
- (Optional) Configure a CAA record

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

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 or email 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 or email 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.

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 13 months (395 days).
- **Renewal:** ACM attempts to renew a private certificate automatically after 11 months.

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.

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.

Browser and application trust

ACM 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 certificates. Java also trusts ACM certificates.

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 email or DNS. For more information, see AWS Certificate Manager email validation and 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 AWS 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.

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 "<character><character>--"
a--example.com	n/a	n/a	✓	Does not start with "<character><character>--"
abc--example.com	n/a	n/a	✓	Does not start with "<character><character>--"
xn--xyz.com	Yes	Yes	✓	Valid Internationalized Domain Name (resolves to 简.com)

Domain Name	Fulfills #1	Fulfills #2	Allowe	Note
xn--example.com	Yes	No	✗	Not a valid Internationalized Domain Name
ab--example.com	No	No	✗	Must start with "xn--"

Validity Period

ACM certificates are valid for 13 months (395 days).

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

The following sections discuss how to use the ACM console or AWS CLI to request 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 wild-card name will appear in the **Subject** field and in the **Subject Alternative Name** extension of the ACM certificate.

   When you request a wild-card 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. In the **Validation method** section, choose either **DNS validation – recommended** or **Email validation**, depending on your needs.

> **Note**
>
> If you are able to edit your DNS configuration, we recommend that you use DNS domain validation rather than email validation. DNS validation has multiple benefits over email validation. See AWS Certificate Manager DNS validation.

Before ACM issues a certificate, it validates that you own or control the domain names in your certificate request. You can use either email validation or DNS validation.

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.

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.

4. In the **Key algorithm** section, chose an algorithm.
5. 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**.
6. 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 CertificateTransparencyLoggingPreference=DISABLED
```

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

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

Validate domain ownership for AWS Certificate Manager public certificates

Before the Amazon certificate authority (CA) can issue a certificate for your site, AWS Certificate Manager (ACM) must prove that you own or control all of the domain names that you specify in your request. You can choose to prove your ownership with Domain Name System (DNS) validation, email validation, or HTTP validation when you request a certificate.

> **Note**
>
> Validation applies only to publicly trusted certificates issued by ACM. 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.

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

- 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.

If you can't edit your domain's DNS database, you must use email validation instead.

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**
>
> After you create a certificate with email validation, you cannot switch to validating it with DNS. To use DNS validation, delete the certificate and then create a new one that uses DNS validation.

Topics

- AWS Certificate Manager DNS validation
- AWS Certificate Manager email validation
- AWS Certificate Manager HTTP 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**
>
> After you create a certificate with email validation, you cannot switch to validating it with DNS. To use DNS validation, delete the certificate and then create a new one that uses DNS validation.

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 five CNAMEs are chained together in your DNS configuration. If you require a longer chaining, we recommend using email validation.

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 wild-card 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	

Domain Name	Record Name	Record Type	Record Value
			_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** button is missing or disabled, see ACM Console does not display "Create records in Route 53" button.

   Choose the **Create records in Route 53** button, 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

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. You can perform verification using either email or DNS. This topic discusses email 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 wild-card 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 publicly trusted certificates issued by ACM. 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.

- After you create a certificate with email validation, you cannot switch to validating it with DNS. To use DNS validation, delete the certificate and then create a new one that uses DNS validation.

Certificate expiration and renewal

ACM certificates are valid for 13 months (395 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

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, or certificates that cover different subdomains. Since the HTTP validation token works for any AWS Region where CloudFront is available, you can re-create the same certificate in multiple Regions. 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-known/pki-validation/ x4.txt`	`https://validation. region.acm-validations.aws/ y4/.well-known/pki-validation/ x4.txt`	Unique
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/.w`	`https://validation. region.acm-validations.a`	Unique

Domain name	Redirect From	Redirect To	Comment
	ell-known/pki-validation/ *x6*.txt	ws/ *y6*/.well-known/pki-validation/ *x6*.txt	

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	Redirect To

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.

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 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.
- **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.
- The CA private key algorithm must be RSA 2048 or RSA 4096.

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

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 Services integrated with ACM.

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, click 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 wild-card name will appear in the **Subject** field and in the **Subject Alternative Name** extension of the ACM certificate.

   > **Note**
   >
   > When you request a wild-card 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, chose an algorithm.

   For information to help you choose an algorithm, see Tag AWS Certificate Manager resources.

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**
>
> You cannot export a publicly trusted certificate or its private key, regardless of whether it's issued by ACM or is imported.

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.

```
[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)"'

[Windows]
$ 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, 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 add a `KeyUsage` or `ExtendedKeyUsage` extension if the extension was not present in the previously imported certificate.
- 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 RSA PRIVATE KEY-----
Base64-encoded private key
-----END RSA 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 Services integrated with ACM.

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

List certificates managed by AWS Certificate Manager

You can use the ACM console or AWS CLI to list the certificates managed by ACM. The console can list up to 500 certificates in a page, and the CLI up to 1000.

To list your certificates using the console

1. Open the ACM console at https://console.aws.amazon.com/acm/.

2. Review the information in the certificate list. You can navigate through multiple pages of certificates using the page numbers at upper-right. Each certificate occupies a row with the following columns displayed by default for each certificate:

- **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.

By choosing the settings icon in the upper-right corner of the console, you can customize 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 list your certificates using the AWS CLI

Use the list-certificates command to list your ACM-managed certificates as shown in the following example:

```
$ aws acm list-certificates --max-items 10
```

The command returns information similar to the following:

```
{
    "CertificateSummaryList": [
        {
            "CertificateArn":
 "arn:aws:acm:Region:444455556666:certificate/certificate_ID",
            "DomainName": "example.com"
  "SubjectAlternativeNameSummaries": [
                "example.com",
                "other.example.com"
            ],
            "HasAdditionalSubjectAlternativeNames": false,
            "Status": "ISSUED",
            "Type": "IMPORTED",
            "KeyAlgorithm": "RSA-2048",
            "KeyUsages": [
                "DIGITAL_SIGNATURE",
                "KEY_ENCIPHERMENT"
            ],
            "ExtendedKeyUsages": [
                "NONE"
```

```
            ],
            "InUse": false,
            "RenewalEligibility": "INELIGIBLE",
            "NotBefore": "2022-06-14T23:42:49+00:00",
            "NotAfter": "2032-06-11T23:42:49+00:00",
            "CreatedAt": "2022-08-25T19:28:05.531000+00:00",
            "ImportedAt": "2022-08-25T19:28:05.544000+00:00"
        },...
    ]
}
```

By default, only certificates with **keyTypes** RSA_1024 or RSA_2048 and with at least one specified domain are returned. To see other certificates that you control, such as domainless certificates or certificates using a different algorithm or bit size, provide the `--includes` parameter as shown in the following example. The parameter allows you to specify a member of the Filters structure.

```
$ aws acm list-certificates --max-items 10 --includes keyTypes=RSA_4096
```

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"
    }
}
```

Delete certificates managed by AWS Certificate Manager

You can use the ACM console or the AWS CLI to delete a certificate.

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*.
- 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
```

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.

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>--"

Domain Name	Fulfills #1	Fulfills #2	Allowe	Note
abc--example.com	n/a	n/a	✓	Does not start with "<character><character>--"
xn--xyz.com	Yes	Yes	✓	Valid Internationalized Domain Name (resolves to 简.com)
xn--example.com	Yes	No	✗	Not a valid Internationalized Domain Name
ab--example.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 either DNS validation or email validation. 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 60 days prior to expiration, ACM checks for the following renewal criteria:

- 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 45 days, 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

ACM certificates are valid for 13 months (395 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 45 days, 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 60 days prior to expiration, ACM checks for the following renewal criteria:

- 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 45 days, 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.

```
{
   "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 AWS 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 AWS 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 60 days prior to expiration. If ACM cannot automatically renew your certificate, it sends certificate renewal event notices to your AWS Health Dashboard at 45 day, 30 day, 15 day, 7 day, 3 day, and 1 day intervals from expiration to inform you that you need to take action. The AWS 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 AWS Health Dashboard:

1. Log in to the AWS 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**.

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

Services integrated with ACM

AWS Certificate Manager supports a growing number of AWS services. You cannot install your ACM certificate or your private AWS Private CA certificate directly on your AWS based website or application.

> **Note**
>
> Public ACM certificates can be installed on Amazon EC2 instances that are connected to a Nitro Enclave, but not to other Amazon EC2 instances. For information about setting up a standalone web server on an Amazon EC2 instance not connected to a Nitro Enclave, see Tutorial: Install a LAMP web server on Amazon Linux 2 or Tutorial: Install a LAMP web server with the Amazon Linux AMI.

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 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**
>
> You cannot associate ACM certificates with an EC2 instance that is not connected to a Nitro Enclave.

AWS CloudFormation

AWS 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 AWS 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; AWS CloudFormation handles all of that. ACM certificates are included as a template resource, which means that AWS 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 AWS CloudFormation.

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

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

> **Note**
>
> If you create an ACM certificate with AWS CloudFormation, the AWS 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*.

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
- Best practices

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 the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

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 Services integrated with ACM.

   > **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
- Use a service-linked role (SLR) with ACM
- Troubleshooting AWS Certificate Manager identity and access

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in ACM.

Service user – If you use the ACM service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more ACM features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in ACM, see Troubleshooting AWS Certificate Manager identity and access.

Service administrator – If you're in charge of ACM resources at your company, you probably have full access to ACM. It's your job to determine which ACM features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with ACM, see How AWS Certificate Manager works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to ACM. To view example ACM identity-based policies that you can use in IAM, 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* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using

credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Methods to assume a role in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity

is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **Service role** – 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*.
  - **Service-linked role** – 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.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API

requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Use an IAM role to grant permissions to applications running on Amazon EC2 instances 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 is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

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**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

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*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose

between a managed policy or an inline policy, 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 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.

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.

Access control lists (ACLs)

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.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a

service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see Service control policies in the *AWS Organizations User Guide*.

- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. 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

IAM feature	ACM support
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Partial
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. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. 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. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. 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. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

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. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, 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 (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. 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.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. 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): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

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

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switch from a user to an IAM role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for ACM

Supports forward access sessions (FAS): Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. 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. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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
- 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 AWS 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.

```
{
            "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.

Retrieving a certificate

The following policy allows a user to retrieve a specific ACM certificate.

```
{
            "Version":"2012-10-17",
            "Statement":{
            "Effect":"Allow",
            "Action":"acm:GetCertificate",
            "Resource":"arn:aws:acm:region:account:certificate/certificate_ID"
            }
            }
```

Importing a certificate

The following policy allows a user to import a certificate.

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

```
            "Statement":{
            "Effect":"Allow",
            "Action":"acm:ImportCertificate",
            "Resource":"arn:aws:acm:region:account:certificate/certificate_ID"
            }
            }
```

Deleting a certificate

The following policy allows a user to delete a specific ACM certificate.

```
{
            "Version":"2012-10-17",
            "Statement":{
            "Effect":"Allow",
            "Action":"acm:DeleteCertificate",
            "Resource":"arn:aws:acm:region:account: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*
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	*

ACM API Operations	Required Permissions (API Operations)	Resources
ListTagsForCertificate	acm:ListTagsForCertificate	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*
PutAccountConfiguration	acm:PutAccountConfiguration	*
RemoveTagsFromCertificate	acm:RemoveTagsFromCertificate	arn:aws:acm: *region*:*account*:certificate/ *certificate_ID*
RequestCertificate	acm:RequestCertificate	arn:aws:acm: *region*:*account*:certificate/* or *
ResendValidationEmail	acm:ResendValidationEmail	arn:aws:acm: *region*:*account*:certificate/ *certificate_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, and retrieve ACM certificates.

```
{
"Version":"2012-10-17",
"Statement":{
    "Effect":"Allow",
    "Action":[
        "acm:DescribeCertificate",
        "acm:ListCertificates",
        "acm:GetCertificate",
        "acm:ListTagsForCertificate",
        "acm:GetAccountConfiguration"
    ],
    "Resource":"*"
}
}
```

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.

AWSCertificateManagerFullAccess

This policy provides full access to all ACM actions and resources.

```
{
"Version":"2012-10-17",
"Statement":[
    {
        "Effect":"Allow",
        "Action":[
            "acm:*"
        ],
        "Resource":"*"
    },
    {
        "Effect":"Allow",
        "Action":"iam:CreateServiceLinkedRole",
        "Resource":"arn:aws:iam::*:role/aws-service-role/acm.amazonaws.com/
AWSServiceRoleForCertificateManager*",
        "Condition":{
            "StringEquals":{
                "iam:AWSServiceName":"acm.amazonaws.com"
            }
        }
    },
    {
        "Effect":"Allow",
        "Action":[
            "iam:DeleteServiceLinkedRole",
            "iam:GetServiceLinkedRoleDeletionStatus",
            "iam:GetRole"
        ],
        "Resource":"arn:aws:iam::*:role/aws-service-role/acm.amazonaws.com/
AWSServiceRoleForCertificateManager*"
    }
    ]
}
```

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.

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 `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.

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:CertificateTransparencyLogging`	RequestCertificate	String (`ENABLED`, `DISABLED`)	Filter requests based on ACM certificate transparency logging preference
`acm:CertificateAuthority`	RequestCertificate	ARN	Filter requests based on certificate authorities in the ACM request

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.

```
{
    "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.

```
{
    "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`

```
{
    "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.

```
{
    "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.

```
{
    "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.

```
{
    "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.

```
{
    "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.

```
{
"Version":"2012-10-17",
    "Statement":{
        "Effect":"Deny",
        "Action":"acm:RequestCertificate",
        "Resource":"*",
        "Condition":{
            "Null" : {
                "acm:CertificateAuthority":"false"
            }
        }
    }
}
```

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
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

Region name	Region identity	Support in ACM
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

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.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

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
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	Following the instructions in Using temporary credentia

Which user needs programmatic access?	To	By
	to the AWS CLI, AWS SDKs, or AWS APIs.	ls with AWS resources in the *IAM User Guide*.
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*.

Best practices

Best practices are recommendations that can help you use AWS Certificate Manager (AWS Certificate Manager) more effectively. The following best practices are based on real-world experience from current ACM customers.

Topics

- Account-level separation
- AWS CloudFormation
- Certificate pinning
- Domain validation

- Adding or deleting domain names
- Opting out of certificate transparency logging
- 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.

```
{
   "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. AWS CloudFormation then provisions and configures those resources for you. AWS CloudFormation can provision resources that are supported by ACM such as Elastic Load Balancing, Amazon CloudFront, and Amazon API Gateway. For more information, see Services integrated with ACM.

If you use AWS 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 AWS 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 AWS CloudFormation uses to create your test environment.

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, pin your application to all available Amazon root certificates. If you're using a private certificate, pin your application to the CA's root certificate.

Domain validation

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 perform verification using either email or DNS. For more information, see AWS Certificate Manager DNS validation and AWS Certificate Manager email 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.

Opting out of certificate transparency logging

> **Important**
>
> Regardless of the actions you take to opt out of certificate transparency logging, your certificate might still be logged by any client or individual that has access to the public or private endpoint to which you bind the certificate. However, the certificate won't contain a signed certificate time stamp (SCT). Only the issuing CA can embed an SCT in a certificate.

As of April 30 2018, Google Chrome no longer trusts public SSL/TLS certificates that are not recorded in a certificate transparency log. Therefore, beginning April 24 2018, the Amazon CA began publishing all new certificates and renewals to at least two public logs. Once a certificate has been logged, it cannot be removed. For more information, see Certificate Transparency Logging.

Logging is performed automatically when you request a certificate or when a certificate is renewed, but you can choose to opt out. Common reasons for doing so include concerns about security and privacy. For example, logging internal host domain names gives potential attackers information about internal networks that would otherwise not be public. In addition, logging could leak the names of new or unreleased products and websites.

To opt out of transparency logging when you are requesting a certificate, use the `options` parameter of the request-certificate AWS CLI command or the RequestCertificate API operation. If your certificate was issued before April 24, 2018, and you want to make sure that it is not logged during renewal, you can use the update-certificate-options command or the UpdateCertificateOptions API operation to opt out.

Limitations

- You cannot use the console to enable or disable transparency logging.
- You cannot change logging status after a certificate enters its renewal period, typically 60 days before certificate expiry. No error message is generated if a status change fails.

Once a certificate has been logged, it cannot be removed from the log. Opting out at that point will have no effect. If you opt out of logging when you request a certificate and then choose later to opt back in, your certificate will not be logged until it is renewed. If you want the certificate to be logged immediately, we recommend that you issue a new one.

The following example shows you how to use the request-certificate command to disable certificate transparency when you request a new certificate.

```
aws acm request-certificate \
--domain-name www.example.com \
--validation-method DNS \
--options CertificateTransparencyLoggingPreference=DISABLED \
```

The preceding command outputs the ARN of your new certificate.

```
{
    "CertificateArn": "arn:aws:acm:region:account:certificate/certificate_ID"
}
```

If you already have a certificate, and you don't want it to be logged when it is renewed, use the update-certificate-options command. This command does not return a value.

```
aws acm update-certificate-options \
--certificate-arn arn:aws:acm:region:account:\
certificate/certificate_ID \
--options CertificateTransparencyLoggingPreference=DISABLED
```

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.

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
- Triggering 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. 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 Triggering 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"
  }
}
```

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",
        "CommonName": "example.com",
        "DomainValidationMethod" : "EMAIL" | "DNS",
        "CertificateCreatedDate" : "2018-12-22T18:43:48Z",
        "CertificateExpirationDate" : "2019-12-22T18:43:48Z",
        "InUse" : TRUE | FALSE,
        "Exported" : TRUE | FALSE
    }
 }
```

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",
       "CommonName": "example.com",
       "DomainValidationMethod" : "EMAIL" | "DNS",
       "CertificateCreatedDate" : "2019-12-22T18:43:48Z",
       "CertificateExpirationDate" : "2019-12-22T18:43:48Z",
       "DaysToExpiry" : 395,
       "InUse" : TRUE | FALSE,
```

```
        "Exported" : TRUE | FALSE
      }
}
```

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. If no customer action is taken, ACM makes further renewal attempts at 30 days, 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",
       "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
    }
}
```

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 clicking 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"
      ]
   }
}
```

Triggering 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, 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.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"LambdaCertificateExpiryPolicy1",
         "Effect":"Allow",
         "Action":"logs:CreateLogGroup",
         "Resource":"arn:aws:logs:<region>:<AWS-ACCT-NUMBER>:*"
      },
      {
         "Sid":"LambdaCertificateExpiryPolicy2",
         "Effect":"Allow",
         "Action":[
            "logs:CreateLogStream",
            "logs:PutLogEvents"
         ],
         "Resource":[
            "arn:aws:logs:<region>:<AWS-ACCT-NUMBER>: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 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 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
```

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.

The following sections provide example logs for the supported API operations.

- Adding tags to a certificate (AddTagsToCertificate)
- Deleting a certificate (DeleteCertificate)
- Describing a certificate (DescribeCertificate)
- Exporting a certificate (ExportCertificate)
- Import a certificate (ImportCertificate)
- Listing certificates (ListCertificates)
- Listing tags for a certificate (ListTagsForCertificate)
- Removing tags from a certificate (RemoveTagsFromCertificate)
- Requesting a certificate (RequestCertificate)
- Resending validation email (ResendValidationEmail)
- Retrieving a certificate (GetCertificate)

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"
      }
   ]
}
```

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 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":{
               "passphrase":{
                  "hb":[
                     42,
                     42,
                     42,
                     42,
                     42,
                     42,
                     42,
                     42,
                     42,
                     42
                  ],
                  "offset":0,
                  "isReadOnly":false,
                  "bigEndian":true,
                  "nativeByteOrder":false,
                  "mark":-1,
                  "position":0,
                  "limit":10,
                  "capacity":10,
                  "address":0
               },
               "certificateArn":"arn:aws:acm:us-east-1:123456789012:certificate/
fedcba98-7654-3210-fedc-ba9876543210"
            },
            "responseElements":{
                "certificateChain":
                "-----BEGIN CERTIFICATE-----
                base64 certificate
                -----END CERTIFICATE-----
                -----BEGIN CERTIFICATE-----
                base64 certificate
                -----END CERTIFICATE-----",
                "privateKey":"**********",
                "certificate":
```

```
                "-----BEGIN CERTIFICATE-----
                base64 certificate
                -----END CERTIFICATE-----"
            },
            "requestID":"01234567-89ab-cdef-0123-456789abcdef",
            "eventID":"fedcba98-7654-3210-fedc-ba9876543210",
            "eventType":"AwsApiCall"
         }
      }
   ]
}
```

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"
}
```

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"
      }
   ]
}
```

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":{
            "subjectAlternativeNames":[
               "example.net"
            ],
            "domainName":"example.com",
            "domainValidationOptions":[
               {
                  "domainName":"example.com",
                  "validationDomain":"example.com"
               },
```

```
                {
                   "domainName":"example.net",
                   "validationDomain":"example.net"
                }
             ],
             "idempotencyToken":"8186023d89681c3ad5"
          },
          "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",
          "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"
      }
   ]
}
```

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"
      }
   ]
}
```

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. ACM publishes metrics once per day for every certificate in an account until expiration.

The `AWS/CertificateManager` namespace includes the following metric.

Metric	Description	Unit	Dimensions
`DaysToExpiry`	Number of days until a certificate expires. ACM stops publishin g this metric after a certificate expires.	Integer	CertificateArn • Value: ARN of the certificate

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
- Deleting a certificate
- Describing a certificate
- Exporting a certificate
- Retrieve a certificate and certificate chain
- Importing a certificate
- Listing certificates
- Renewing a certificate
- Listing certificate tags
- Removing tags from a certificate
- Requesting a certificate
- Resending validation email

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);
   }
}
```

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 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 private certificate issued by a private certificate authority (CA) in the PKCS #8 format. (It is not possible to export public certificates whether they are ACM-issued or imported.) It also exports the certificate chain and private key. In the example, the passphrase for the key is stored in a local file.

```
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();
   }
}
```

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 certificates

The following example shows how to use the ListCertificates function.

```
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
    }]
}
```

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);
   }
}
```

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}]}
```

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);
   }
}
```

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

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.

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 other problems
- Handling exceptions

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

Public ACM certificates can be installed on Amazon EC2 instances that are connected to a Nitro Enclave, but not to other Amazon EC2 instances. For information about setting up a standalone web server on an Amazon EC2 instance not connected to a Nitro Enclave, see Tutorial: Install a LAMP web server on Amazon Linux 2 or Tutorial: Install a LAMP web server with the Amazon Linux AMI.

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

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
- I can't switch to DNS validation

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.

The domain registrar does not display contact information or privacy protection is enabled

For domains purchased from Route 53, privacy protection is enabled by default and your email address is mapped to a `whoisprivacyservice.org`, `contact.gandi.net`, or `identity-`

`protect.org` email address. Ensure that your registrant email address on file with your domain registrar is up to date so that the email sent to these obscured email addresses can be forwarded to an email address that you control.

> **Note**
>
> Privacy protection for some domains that your purchase with Route 53 will be enabled even if you choose to make your contact information public. For example, privacy protection for the .ca top level domain cannot be programmatically disabled by Route 53. You must contact the AWS Support Center and request that privacy protection be disabled.

After making available at least one of the eight email addresses to which AWS sends validation email and confirming that you can receive email for that address, you are ready to request a certificate through ACM. After you make a certificate request, ensure the intended email address appears in the list of email addresses in the AWS Management Console. While the certificate is in the **Pending validation** state, you can expand the list to view it by clicking the icon next to the domain name in the box labeled **Validation not complete**. You can also view the list in **Step 3: Validate** of the ACM **Request a Certificate** wizard. The listed email addresses are the ones to which email was sent.

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.

I can't switch to DNS validation

After you create a certificate with email validation, you cannot switch to validating it with DNS. To use DNS validation, delete the certificate and then create a new one that uses 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 Services integrated with ACM.
- 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 (60 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 13 months (395 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.

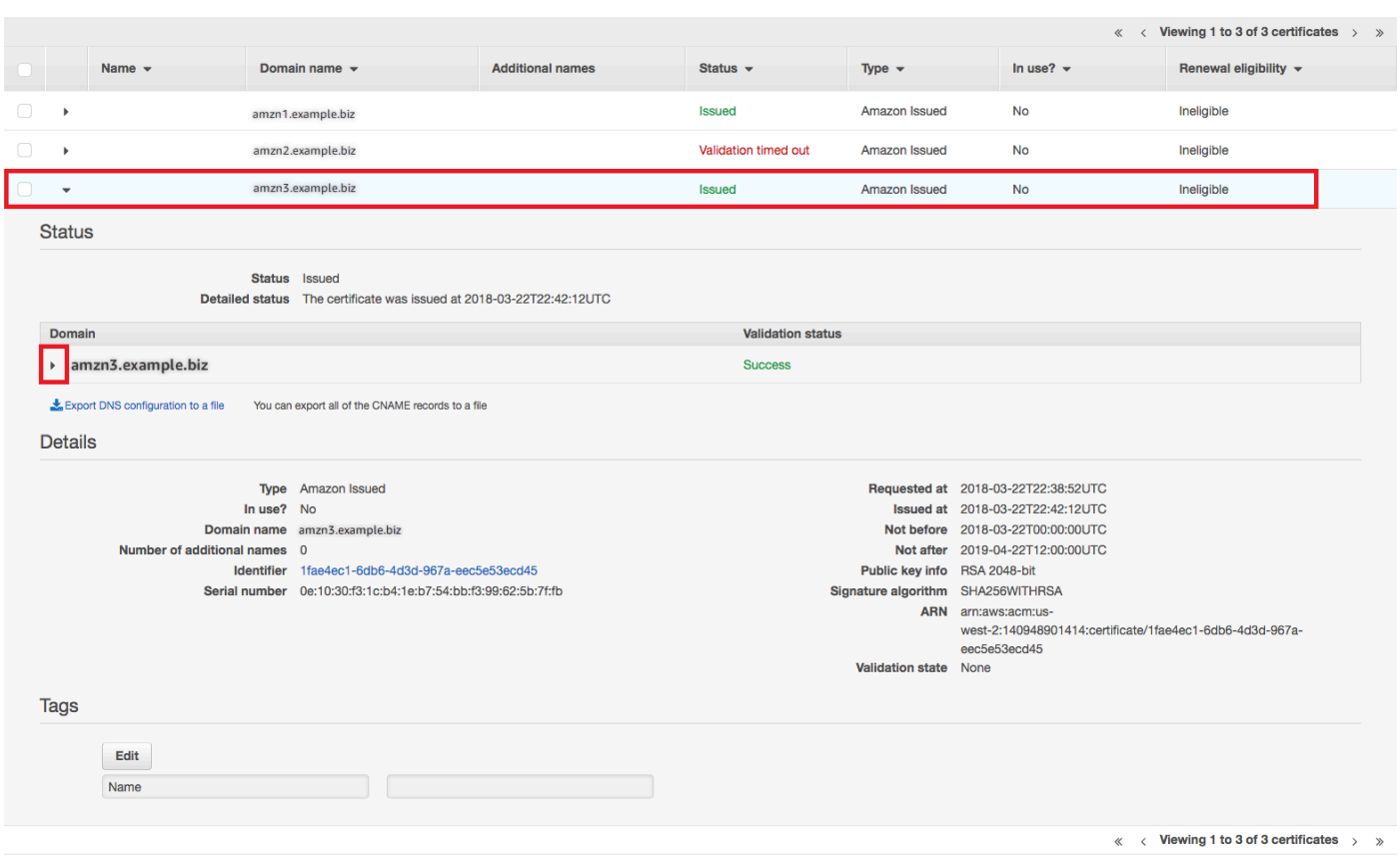
See Validate with Email for instructions on identifying which domains are in the `PENDING_VALIDATION` state and repeating the validation process for those domains.

Managed certificate renewal for DNS-validated certificates

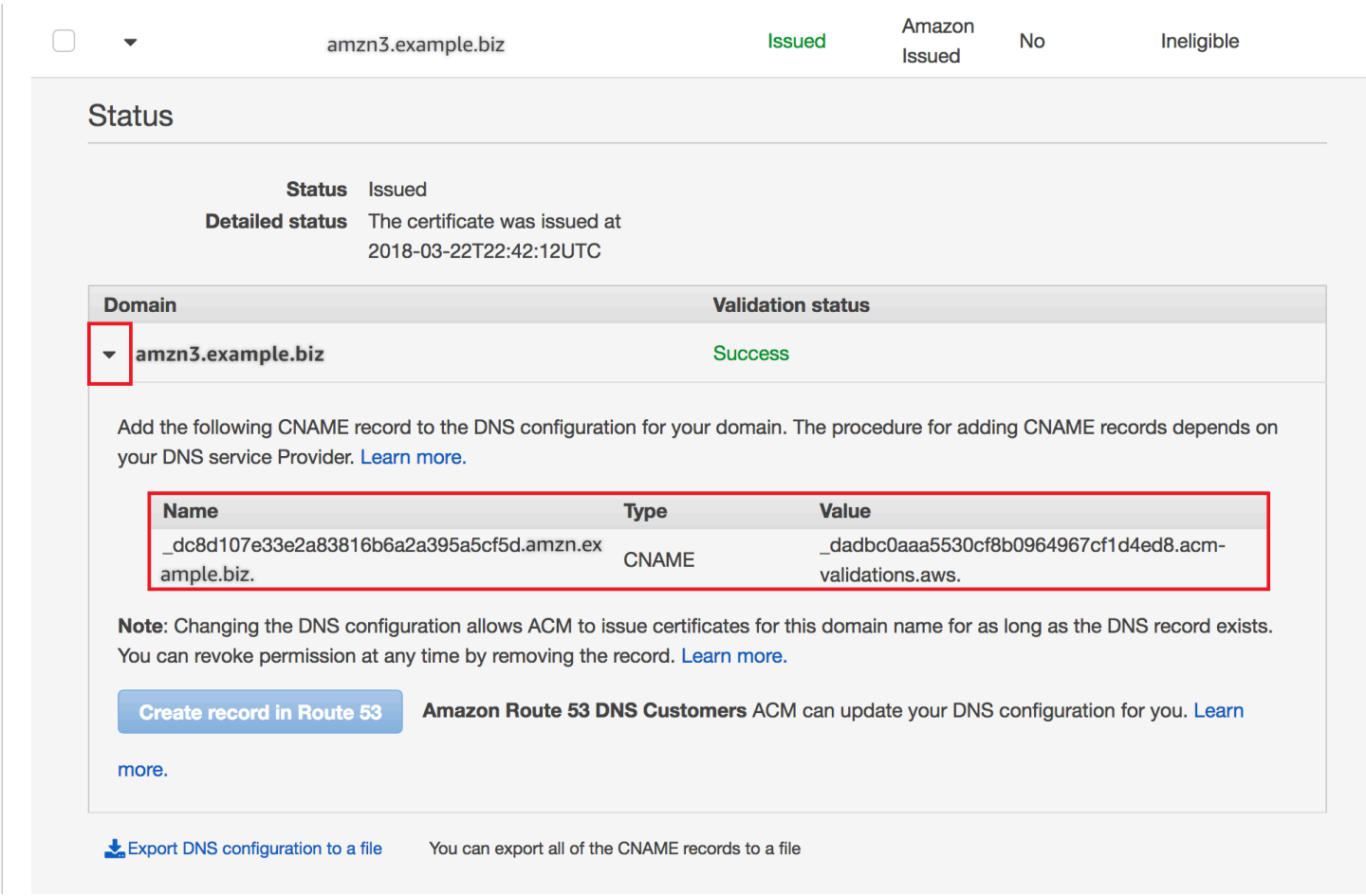
ACM does not attempt TLS validation for DNS-validated certificates. If ACM fails to renew a certificate you validated with DNS validation, it is most likely due to missing or inaccurate CNAME records in your DNS configuration. If this occurs, ACM notifies you that the certificate could not be renewed 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 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.
- 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.

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, pin your application to all available Amazon root certificates. 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 using the AWS KMS console at https://console.aws.amazon.com/kms. 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*.

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 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:

ACM failure code	Comment
	• 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.
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.

ACM failure code	Comment
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.

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.	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 certificates 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.	5,000

Item	Default quota
Number of imported certificates	2,500
Number of imported certificates per year (last 365 days)	5,000
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

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

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

API call	Requests per second
AddTagsToCertificate	5
DeleteCertificate	10
DescribeCertificate	10
ExportCertificate	10
GetAccountConfiguration	1
GetCertificate	10
ImportCertificate	1
ListCertificates	8
ListTagsForCertificate	10
PutAccountConfiguration	1
RemoveTagsFromCertificate	5
RenewCertificate	5
RequestCertificate	5
ResendValidationEmail	1
UpdateCertificateOptions	5

For more information, see AWS Certificate Manager API Reference.

Document history

The following table describes the documentation release history of AWS Certificate Manager beginning in 2018.

Change	Description	Date
ACM supports HTTP validation 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 separation 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
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-condi	August 24, 2023

	tions.html#acm-conditions-supported.	
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
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
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
Launching AWS Private CA	Launched ACM Private Certificate Manager (CM), and extension of AWS Certificate 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 transparency 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 issues with AWS Certificate Manager.	July 07, 2017
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 AWS CloudFormation to Services integrated with ACM.	May 27, 2017

Change	Description	Release Date
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
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 Services integrated with ACM.	March 21, 2017
Update	Updated the documentation about Managed certificate renewal.	February 20, 2017

Change	Description	Release Date
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