AWS Certificate Manager

User Guide

Version 1.0

AWS Certificate Manager: User Guide

Table of Contents

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 (p. 3) either by issuing them directly with ACM or by importing (p. 56) 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 (p. 62) ACM certificates signed by ACM Private CA for use anywhere in your internal PKI.

Is ACM the right service for me?

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 (p. 3). 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. **ACM 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 ACM 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 ACM Private CA User Guide.

Concepts (p. 127)

ACM certificate characteristics (p. 1)

Supported Regions (p. 3)

Services integrated with AWS Certificate Manager (p. 3)

Site seals and trust logos (p. 6)

API rate quotas (p. 8)

Best practices (p. 22)

Pricing for AWS Certificate Manager (p. 9)

ACM certificate characteristics

Public certificates provided by ACM have the characteristics described in this section.

> **Note**
> These characteristics apply only to certificates provided by ACM. They might not apply to certificates that you import into ACM (p. 56).

Domain Validation (DV)

ACM certificates are domain validated. That is, the subject field of an ACM certificate identifies a domain name and nothing more. When you request an ACM certificate, you must validate that you own or control all of the domains that you specify in your request. You can validate ownership by using email or DNS. For more information, see Email validation (p. 40) and DNS validation (p. 37).

Validity Period

The validity period for ACM certificates is 13 months (395 days).

Managed Renewal and Deployment

ACM manages the process of renewing ACM certificates and provisioning the certificates after they are renewed. Automatic renewal can help you avoid downtime due to incorrectly configured, revoked, or expired certificates. For more information, see Managed renewal for ACM certificates (p. 47).

Browser and Application Trust

ACM certificates are trusted by all major browsers including Google Chrome, Microsoft Internet Explorer and Microsoft Edge, Mozilla Firefox, and Apple Safari. Browsers that trust ACM certificates display a lock icon in their status bar or address bar when connected by SSL/TLS to sites that use ACM certificates. ACM certificates are also trusted by Java.

Multiple Domain Names

Each ACM certificate must include at least one fully qualified domain name (FQDN), and you can add additional names if you want. For example, when you are creating an ACM certificate for `www.example.com`, you can also add the name `www.example.net` if customers can reach your site by using either name. This is also true of bare domains (also known as the zone apex or naked domains). That is, you can request an ACM certificate for www.example.com and add the name example.com. For more information, see Requesting a public certificate (p. 31).

Wildcard Names

ACM allows you to use an asterisk (*) in the domain name to create an ACM certificate containing a wildcard name that can protect several sites in the same domain. For example, `*.example.com` protects `www.example.com` and `images.example.com`.

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

Algorithms

A certificate must specify an algorithm and key size. Currently, the following public key algorithms are supported by ACM (API name in parentheses):

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

Important
Note that integrated services allow only algorithms and key sizes they support to be associated with their resources. Further, their support differs depending on whether the certificate is imported into IAM or into ACM. For more information, see the documentation for each service.

- For Elastic Load Balancing, see HTTPS Listeners for Your Application Load Balancer.
- For CloudFront, see Supported SSL/TLS Protocols and Ciphers.

Exceptions

Note the following:

- ACM does not provide extended validation (EV) certificates or organization validation (OV) certificates.
- ACM does not provide certificates for anything other than the SSL/TLS protocols.
- You cannot use ACM certificates for email encryption.
- ACM allows only UTF-8 encoded ASCII for domain names, including labels that contain "xn--" (Punycode). ACM does not accept Unicode input (u-labels) for domain names.
- ACM does not currently permit you to opt out of managed certificate renewal (p. 47) for ACM certificates. Also, managed renewal is not available for certificates that you import into ACM.
- You cannot request certificates for Amazon-owned domain names such as those ending in amazonaws.com, cloudfront.net, or elasticbeanstalk.com.
- You cannot download the private key for an ACM certificate.
- You cannot directly install ACM certificates on your Amazon Elastic Compute Cloud (Amazon EC2) website or application. You can, however, use your certificate with any integrated service. For more information, see Services integrated with AWS Certificate Manager (p. 3).
- ACM does not support issuing or renewing certificates for domain names that contain hyphens as their third and fourth characters. For example, the following domain name is not permitted:

```
ab--example.com
```

Supported Regions

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.

Services integrated with AWS Certificate Manager

AWS Certificate Manager supports a growing number of AWS services. You cannot install your ACM certificate or your private ACM 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 (p. 5), 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 a user name and password, 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 frontend 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 frontend 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 clickstream 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.

Site seals and trust logos

Amazon doesn't provide a site seal or allow its trademark to be used as one:

- AWS Certificate Manager (ACM) doesn't provide a secure site seal that you can use on your website. If you want to use a site seal, you can obtain one from a third-party vendor. We recommend choosing a vendor that evaluates and asserts the security of your website or business practices.
- Amazon doesn't allow its trademark or logo to be used as a certificate badge, site seal, or trust logo. Seals and badges of this type can be copied to sites that don't use the ACM service, and can be used inappropriately to establish trust under false pretenses. To protect our customers and the reputation of Amazon, we don't allow our trademark and logo to be used in this way.

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 *AWSGeneral Reference Guide*. To request quota increases, create a case at the AWS 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 ACM Private CA do not count toward this total.	2500

Item	Default quota
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 **AWS Support Center**. Certificates signed by a CA from ACM Private CA do not count toward this total.	Twice your account quota
Number of imported certificates	2,500
Number of imported certificates per year (last 365 days)	Twice your account quota
Number of domain names per ACM certificate The default quota is 10 domain names for each ACM certificate. Your quota may be greater. The first domain name that you submit is included as the subject common name (CN) of the certificate. All names are included in the Subject Alternative Name extension. You can request up to 100 domain names. To request an increase in your quota, create a case at the AWS Support Center . 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 (p. 23). 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 Certificate Manager Private Certificate Authority (ACM 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 ACM 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 Requesting a private PKI certificate (p. 33). You can also issue private certificates by using the standalone ACM 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 ACM Private CA. For up-to-date rate quota information on `IssueCertificate`, see the endpoints and quotas for ACM Private CA.

Requests-per-second quota for each ACM API operation

API call	Requests per second
`AddTagsToCertificate`	5
`DeleteCertificate`	10
`DescribeCertificate`	10
`ExportCertificate`	5
`GetAccountConfiguration`	1
`GetCertificate`	10
`ImportCertificate`	1
`ListCertificates`	8
`ListTagsForCertificate`	10

API call	Requests per second
`PutAccountConfiguration`	1
`RemoveTagsFromCertificate`	5
`RenewCertificate`	5
`RequestCertificate`	5
`ResendValidationEmail`	1
`UpdateCertificateOptions`	5

For more information, see AWS Certificate Manager API Reference.

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.

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

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. This content includes 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 user accounts with 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 recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- 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 personal data that is stored in Amazon S3.

- If you require FIPS 140-2 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) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form 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 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.

ACM private key security

When you request a public certificate (p. 31), AWS Certificate Manager (ACM) generates a public/private key pair. For imported certificates (p. 56), 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 AWS Certificate Manager (p. 3).

   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

Access to AWS Certificate Manager (ACM) requires credentials that AWS can use to authenticate your requests. The following topics provide details on how you can use AWS Identity and Access Management (IAM) to help secure your private certificate authorities (CAs) by controlling who can access them.

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you first create an AWS account, you begin with a single 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 do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

- **IAM user** – An IAM user is an identity within your AWS account that has specific custom permissions (for example, permissions to create a directory in ACM). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

  In addition to a user name and password, you can also generate access keys for each user. You can use these keys when you access AWS services programmatically, either through one of the several SDKs or by using the AWS Command Line Interface (CLI). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use AWS tools, you must sign the request yourself. ACM supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 signing process in the *AWS General Reference*.
- **IAM role** – An IAM role is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user in that it is an AWS identity with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session. IAM roles with temporary credentials are useful in the following situations:
  - **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated users and roles in the *IAM User Guide*.
  - **AWS service access** – 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 Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **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 Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

Access control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access AWS Certificate Manager resources. For example, you must have permission to create, import, retrieve, or list certificates.

The following topics describe how to manage permissions. We recommend that you read the overview first.

Overview of managing access to your ACM resources

Every AWS Certificate Manager (ACM) resource belongs to an AWS account, and permissions to create or access the resources are defined in permissions policies in that account. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles). Some services (including ACM) also support attaching permissions policies to resources.

> **Note**
> An *account administrator* (or administrator user) is a user with administrator permissions. For more information, see Creating an Admin User and Group in the *IAM User Guide*.

When managing permissions, you decide who gets the permissions, the resources they get permissions for, and the specific actions allowed.

Topics

ACM resources and operations

In ACM, the primary resource is a *certificate*. Certificates have unique Amazon Resource Names (ARNs) associated with them as shown in the following list.

- **ACM Certificate**

  ARN format:

  ```
  arn:aws:acm:region:account:certificate/certificate_ID
  ```

  Example ARN:

  ```
  arn:aws:acm:us-
  west-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678
  ```

Understanding resource ownership

A *resource owner* is the AWS account that created a resource. That is, the resource owner is the AWS account of the *principal entity* that authenticates the request that created the resource. (A principal entity can be an AWS account root user, an IAM user, or an IAM role.) The following examples illustrate how this works.

- If you use the credentials of your AWS account root user to create an ACM certificate, your AWS account owns the certificate.
- If you create an IAM user in your AWS account, you can grant that user permission to create an ACM certificate. However, the account to which that user belongs owns the certificate.
- If you create an IAM role in your AWS account and grant it permission to create an ACM certificate, anyone who can assume the role can create a certificate. However, the account to which the role belongs owns the certificate.

Managing access to ACM certificates

A *permissions policy* describes who has access to what. This section explains the available options for creating permissions policies.

Note
This section discusses using IAM in the context of ACM. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see the IAM User Guide. For information about IAM policy syntax and descriptions, see IAM Policy Reference.

You can use IAM to create policies that apply permissions to IAM users, groups, and roles. These are called *identity–based policies*. IAM offers the following types of identity–based policies:

- **AWS managed policies** – Policies that are created and managed by AWS. These are standalone policies that you can attach to multiple users, groups, and roles in your AWS account.
- **Customer managed policies** – Policies that you create and manage in your AWS account and which you can attach to multiple users, groups, and roles. You have more precise control when using customer managed policies than you have when using AWS managed policies.
- **Inline policies** – Policies that you create and manage and which you embed directly into a single user, group, or role.

Other services, such as Amazon S3, also support resource–based permissions policies. For example, you can attach a policy to an Amazon S3 bucket to manage access permissions to that bucket. ACM does not support resource-based policies.

AWS managed policies for AWS Certificate Manager

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

Topics

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 (p. 134) 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

Customer managed policies

Customer managed policies are standalone identity–based policies that you create and which you can attach to multiple users, groups, or roles in your AWS account. You can manage and create policies using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or the IAM API. For more information, see Customer Managed Policies .

Inline policies

Inline policies are policies that you create and manage and embed directly into a single user, group, or role. The following policy examples show how to assign permissions to perform ACM actions. For more information about attaching inline policies, see Working with Inline Policies in the IAM User Guide. You can use the AWS Management Console, the AWS Command Line Interface (AWS CLI), or the IAM API to create and embed inline policies.

Topics

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"
   }
}
```

Read-only access to ACM

The following policy allows a user to describe and list an ACM certificate and to retrieve the ACM certificate and certificate chain.

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

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

Note
This policy is available as an AWS managed policy in the AWS Management Console. For more information, see AWSCertificateManagerReadOnly (p. 14). To view the managed policy in the console, go to https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AWSCertificateManagerReadOnly.

Full access to ACM

The following policy allows a user to perform any ACM action.

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

Note
This policy is available as an AWS managed policy in the AWS Management Console. For more information, see AWSCertificateManagerFullAccess (p. 15). To view the managed policy in the console, go to https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AWSCertificateManagerFullAccess.

Administrator access to all AWS resources

The following policy allows a user to perform any action on any AWS resource.

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

Note
This policy is available as an AWS managed policy in the AWS Management Console. To view the managed policy in the console, go to https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AdministratorAccess.

Using a service-linked role (SLR) with ACM

AWS Certificate Manager uses an AWS Identity and Access Management (IAM) service-linked role to enable automatic renewals of managed ACM certificates. 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 choose a private CA 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 ACM 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) (p. 124).

> **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 ACM 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 (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes
South America (São Paulo)	sa-east-1	Yes
AWS GovCloud (US-West)	us-gov-west-1	Yes

Region name	Region identity	Support in ACM
AWS GovCloud (US-East) East	us-gov-east-1	Yes

ACM API permissions: Actions and resources reference

When you set up access control (p. 12) and write permissions policies that you can attach to an IAM identity (identity-based policies), 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`	*
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/`* *

ACM API Operations	Required Permissions (API Operations)	Resources
		or *
ResendValidationEmail	acm:ResendValidationEmail	arn:aws:acm:*region*:*account*:certifica *certificate_ID*
UpdateCertificateOptions	acm:UpdateCertificateOptions	arn:aws:acm:*region*:*account*:certifica *certificate_ID*

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 the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access ACM through the network. Clients should support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). 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.

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

- Opting out of certificate transparency logging (p. 24)
- Turn on AWS CloudTrail (p. 25)

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 AWS Certificate Manager (p. 3).

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 (p. 6). 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`. 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 renewal for ACM certificates (p. 47) 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 DNS validation (p. 37) and Email validation (p. 40).

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 (p. 129).

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 RequestCertificateAPI 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 (p. 75) to see example trails for ACM actions.

Setting up

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 AWS Certificate Manager (p. 3). The following topics discuss the steps you need to perform before using ACM.

> **Note**
> In addition to using certificates provided by ACM, you can also import certificates into ACM. For more information, see Import certificates (p. 56).

Topics

Set up AWS and IAM

Before you can use ACM, you must sign up for Amazon Web Services. As a best practice, you can create an IAM user to limit the actions your users can perform.

Sign up for AWS

If you are not already an Amazon Web Services (AWS) customer, you must sign up to be able to use ACM. Your account is automatically signed up for all available services, but you are charged for only the services that you use. Also, if you are a new AWS customer, you can get started for free. For more information, see AWS Free Tier.

To sign up for an AWS account

1. Go to https://aws.amazon.com/ and choose **Sign Up**.
2. Follow the on-screen instructions.

> **Note**
> Part of the sign-up procedure includes receiving an automated telephone call and entering the supplied PIN on the telephone keypad. You must also supply a credit card number even if you are signing up for the free tier.

Create an IAM user

All AWS accounts have root user credentials (that is, the credentials of the account owner). These credentials allow full access to all resources in the account. Because you can't restrict permissions for root user credentials, we recommend that you delete your root user access keys. Then create AWS Identity and Access Management (IAM) user credentials for everyday interaction with AWS. For more information, see Lock away your AWS account (root) access keys in the *IAM User Guide*.

Note
You may need AWS account root user access for specific tasks, such as changing an AWS support plan or closing your account. In these cases, sign in to the AWS Management Console with your email and password. See Email and Password (Root User).

For a list of tasks that require root user access, see AWS Tasks That Require AWS Account Root User.

With IAM, you can securely control access to AWS services and resources for users in your AWS account. For example, if you require administrator-level permissions, you can create an IAM user, grant that user full access, and then use those credentials to interact with AWS. If you need to modify or revoke your permissions, you can delete or modify the policies that are associated with that IAM user.

If you have multiple users that require access to your AWS account, you can create unique credentials for each user and define who has access to which resources. You don't need to share credentials. For example, you can create IAM users with read-only access to resources in your AWS account and distribute those credentials to your users.

ACM also provides two AWS managed policies that you can use:

- **AWSCertificateManagerFullAccess**
- **AWSCertificateManagerReadOnly**

Note
Any activity or costs that are associated with the IAM user are billed to the AWS account owner.

Register a domain name

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. The registrar queries WHOIS to determine whether the requested FQDN is available. If it is, the registrar usually lists related names that differ by domain extension and provides you an opportunity to acquire any of the available names. 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 email for your domain

Note
The following steps are required only if you use email validation to assert that you own or control the FQDN (fully qualified domain name) specified in your certificate request. ACM requires that you validate ownership or control before it issues a certificate. You can use either email validation or DNS validation. For more information about email validation, see Email validation (p. 40).
If you are able to edit your DNS configuration, we recommend that you use DNS domain validation rather than email validation. DNS validation removes the need to configure email for the domain name. For more information about DNS validation, see DNS validation (p. 37).

Use your registrar's website to associate your contact addresses with your domain name. The registrar adds the contact email addresses to the WHOIS database and adds one or more mail servers to the mail exchanger (MX) records of a DNS server. If you choose to use email validation, ACM sends email to the contact addresses and to five common administrative addresses formed from your MX record. ACM

sends up to eight validation email messages every time you create a new certificate, renew a certificate, or request new validation mail. The validation email contains instructions for confirming that the domain owner or an appointed representative approves the ACM Certificate. For more information, see Email validation (p. 40). If you have trouble with validation email, see Troubleshoot email validation problems (p. 114).

WHOIS database

The WHOIS database contains contact information for your domain. To validate your identity, ACM sends an email to the following three addresses in WHOIS. You must make sure that your contact information is public or that email that is sent to an obfuscated address is forwarded to your real email address.

- Domain registrant
- Technical contact
- Administrative contact

MX record

When you register your domain, your registrar sends your mail exchanger (MX) record to a Domain Name System (DNS) server. An MX record indicates which servers accept mail for your domain. The record contains a fully qualified domain name (FQDN). You can request a certificate for apex domains or subdomains.

For example, if you use the console to request a certificate for abc.xyz.example.com, ACM first tries to find the MX record for that subdomain. If that record cannot be found, ACM performs an MX lookup for xyz.example.com. If that record cannot be found, ACM performs an MX lookup for example.com. If that record cannot be found or there is no MX record, ACM chooses the original domain for which the certificate was requested (abc.xyz.example.com in this example). ACM then sends email to the following five common system administration addresses for the domain or subdomain:

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

If you are using the RequestCertificate API operation or the request-certificate AWS CLI command, AWS does not perform an MX lookup. Instead, `RequestCertificate` lets you specify both your domain name and the name of a validation domain. If you specify the optional `ValidationDomain` parameter, AWS sends the preceding five email messages there rather than to your domain.

ACM always sends validation email to the five common addresses listed previously whether you are using the console, the API, or the AWS CLI. However, AWS performs an MX lookup only when you use the console to request a certificate.

If you do not receive validation email, see Not receiving validation email (p. 114) for information about possible causes and workarounds.

(Optional) Configure a CAA record

You can optionally configure a Certification Authority Authorization (CAA) DNS record to specify that AWS Certificate Manager (ACM) is allowed to issue a certificate for your domain or subdomain. After it

validates your domain, ACM checks for the presence of CAA records to make sure it can issue a certificate for you. You can choose to not configure a CAA record for your domain if you do not want to enable CAA checking.

A CAA record contains the following data fields:

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.

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.

```
Domain   Record type  Flags  Tag      Value

example.com.   CAA           0      issue   "SomeCA.com"
example.com.   CAA           0      issue   "amazon.com"
example.com.   CAA           0      issue   "amazontrust.com"
example.com.   CAA           0      issue   "awstrust.com"
example.com.   CAA           0      issue   "amazonaws.com"
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.

Issuing and managing certificates

ACM certificates can be 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") or 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 ACM 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 (p. 5), 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

Requesting a public certificate

The following sections discuss how to use the ACM console or AWS CLI to request a public ACM certificate.

If you encounter problems when requesting a certificate, see Troubleshooting certificate requests (p. 110).

To request a certificate for a private PKI using ACM Private CA, see Requesting a private PKI certificate (p. 33).

> **Note**
> Unless you choose to opt out, publicly trusted ACM certificates are automatically recorded in at least two certificate transparency databases. You cannot currently use the console to opt out. You must use the AWS CLI or the ACM API. For more information, see Opting out of certificate transparency logging (p. 24). For general information about transparency logs, see Certificate Transparency Logging (p. 129).

Topics

- Request a public certificate using the CLI (p. 33)

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.
3. 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`**).
4. In the **Select validation method** section, choose either **DNS validation** 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 DNS validation (p. 37).

   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 three contact addresses registered in the WHOIS database, and to five common system administration addresses for each domain name. You or an authorized representative must reply to one of these email messages. For more information, see Email validation (p. 40).

   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 DNS validation (p. 37).
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 Tagging AWS Certificate Manager certificates (p. 64).

   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.

```
aws acm request-certificate \
--domain-name www.example.com \
--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:account:certificate/certificate_ID"
}
```

Requesting a private PKI certificate

The following sections discuss how to use the ACM console or an ACM CLI command to request a private PKI certificate that's signed by a private certificate authority (CA) that was previously created using ACM Private CA. The CA may either reside in your account or be shared with you by a different account. For more information about creating a private CA, see Create a Private Certificate Authority.

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

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

- You must use DNS subject names. For more information, see Domain Names (p. 130).
- You can use only a 2048-bit RSA private key algorithm.
- The only supported signing algorithm is SHA256WithRSAEncryption.
- Each certificate is valid for 13 months (395 days).
- ACM renews the certificate automatically, if possible, after 11 months.

Certificates signed by a private CA do not have a domain validation step.

Note
The end date of the CA certificate for a private CA must exceed the end date of the requested certificate, or else the certificate request will fail.

The private CA must have a status of Active, and the CA private key type must be RSA 2048 or RSA 4096.

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

Topics

Configuring access to a private CA

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

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

  For single-account issuance and renewals to be enabled, the ACM Private CA administrator must grant permission to the ACM service principal to create, retrieve, and list certificates. This is done using the ACM Private CA API action `CreatePermission` or the AWS CLI command create-permission. The account owner assigns these permissions to an IAM user or group that's responsible for issuing the 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 ACM Private CA administrator must attach a resource-based policy to the CA using the ACM 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 AWS Certificate Manager (p. 3).

Request a private PKI certificate using the ACM 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.

   Note
   Private CAs with ECDSA keys are not available for issuing certificates and are grayed out in the list of CAs.

   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.

5. To add another name, 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`**).

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 Tagging AWS Certificate Manager certificates (p. 64).

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 PKI certificate using the CLI

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

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

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

```
{
    "CertificateArn": "arn:aws:acm:region:account: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.

Validating domain ownership

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 either Domain Name System (DNS) validation or with email validation at the time you request a certificate.

Note
Validation applies only to publicly trusted certificates issued by ACM. ACM does not validate domain ownership for imported certificates (p. 56) or for certificates signed by a private CA.

In general, 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.
- To be renewed, email-validated certificates require an action by the domain owner. ACM begins sending renewal notices 45 days before expiration. These notices go to the domain's WHOIS mailbox addresses and to 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 lack authorization to edit your domain's DNS database, you must use email validation (p. 40) instead.

Topics

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.

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 (p. 40) 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 renewal for ACM certificates (p. 47).

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 (p. 40).

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.	
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 (p. 112).

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

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

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

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

```
_a79865eb4cd1a6ab990a45779b4e0b96
```

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

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

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

Setting up DNS validation

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

To set up DNS validation in the console

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

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

   a. (Optional) Validate with Route 53.

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

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

   > **Note**
   > If you are in fact using Route 53 but the **Create record in Route 53** button is missing or disabled, see ACM Console does not display "Create record in Route 53" button (p. 113).

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. 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 (p. 38) before you add information to your DNS provider's database. If you do encounter problems, see Troubleshoot DNS validation problems (p. 112).

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.

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. For information about DNS validation, see DNS validation (p. 37).

Note
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.

ACM certificates are valid for 13 months (395 days). To be renewed, email-validated certificates require an action by the domain owner. ACM begins sending renewal notices 45 days before expiration, using the domain's WHOIS mailbox addresses and to 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 encounter problems using email validation, see Troubleshoot email validation problems (p. 114).

Note
Validation applies only to publicly trusted certificates issued by ACM. ACM does not validate domain ownership for imported certificates (p. 56) or for certificates signed by a private CA.

ACM sends email messages to the three contact addresses listed in the WHOIS database and to five common system addresses for each domain that you specify. That is, up to eight email messages will be sent for every domain name and subject alternative name that you include in your request. For example, if you specify only one domain name, you will receive up to eight email messages. To validate, you must act on one of these eight messages within 72 hours. If you specify three domain names, you will receive

up to 24 messages. To validate, you must act on at least three of these emails, one for each name that you specified, within 72 hours.

Email messages are sent to the following three registered contact addresses in WHOIS:

- Domain registrant
- Technical contact
- Administrative contact

> **Note**
> Some registrars allow you to hide your contact information in your WHOIS listing, and others allow you to substitute your real email address with a privacy (or proxy) address. To prevent problems with receiving the domain validation email from ACM, ensure that your contact information is visible in WHOIS. If your WHOIS listing shows a privacy email address, ensure that email sent to that address is forwarded to your real email address. Or simply list your real email address instead.

If you use the console to request a certificate, ACM performs an MX lookup to determine which servers accept email for your domain and sends mail to the following five common system addresses for first domain found. If you use the RequestCertificate API or the request-certificate AWS CLI command, ACM does not perform an MX lookup. Instead, it sends email to the domain name that you specify in the `DomainName` parameter or in the optional `ValidationDomain` parameter. For more information, see MX record (p. 28).

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

For more information about how ACM determines the email addresses for your domains, see (Optional) Configure email for your domain (p. 27).

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 prepending 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.

> **Note**
> Ensure that email is sent to the administrative addresses for an apex domain, such as `example.com`, rather than to the administrative addresses for a subdomain, such as `test.example.com`. To do that, specify the `ValidationDomain` option in the RequestCertificate API or the request-certificate AWS CLI command. This feature is not currently supported when you use the console to request a certificate.
> Even when all messages are sent to a single email address, you must respond to one message for each domain or subdomain in order to validate it and generate the certificate.

Certificate expiration and renewal

ACM certificates are valid for 13 months (395 days). To be renewed, email-validated certificates require an action by the domain owner. ACM begins sending renewal notices 45 days before expiration, using

the domain's WHOIS mailbox addresses and to 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.

See Email validation (p. 40), above, for more information.

(Optional) Resending validation mail

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 persistent problems with email validation, see the Troubleshoot email validation problems (p. 114) section in Troubleshooting (p. 110).

> **Note**
> The following information applies only to certificates provided by ACM and only to certificates that use email validation. Validation email is not required for private PKI certificates or for certificates that you imported into ACM (p. 56). For information about DNS domain validation, see DNS validation (p. 37).

To resend validation email using the console

1. Sign in to the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/home.
2. In the list of certificates, choose the **Certificate ID** of the certificate you want to validate. This action opens a details page.

   > **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.
3. In the **Domains** section, choose **Resend validation email**, select each of the domains requiring validation, and then choose **Resend**. The **Successfully resent validation emails** banner should appear.

To resend validation email using the AWS CLI

You can use the resend-validation-email command to resend email.

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

> **Note**
> The resend-validation-email command applies only to ACM certificates for which you are using email validation. Validation is not required for certificates that you have imported into ACM or for private certificates that you manage using ACM.

Listing certificates managed by ACM

You can use the ACM console or AWS CLI to list the certificates managed by ACM.

> **Note**
> If you manage 100 or more certificates, we recommend using the AWS CLI option.

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**.

> **Note**
> You can customize the columns that you want to display, as well as other settings, by choosing the gear icon in the upper-right corner of the console. For more information about the available certificate details, see Describing ACM certificates (p. 43).

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:account:certificate/certificate_ID_1",
            "DomainName": "example.com"
        },
        {
            "CertificateArn": "arn:aws:acm:region:account:certificate/certificate_ID_2",
            "DomainName": "mydomain.com"
        }
    ]
}
```

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
```

Describing ACM certificates

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:account:certificate/certificate_ID`
     - **Type** – Identifies the management category of an ACM certificate. Possible values are: **Amazon Issued | Private | Imported**. For more information, see Requesting a public certificate (p. 31), Requesting a private PKI certificate (p. 33), or Importing certificates into AWS Certificate Manager (p. 56).
     - **Status** – The certificate status. Possible values are: **Expired | Failed | Inactive | Issued | Pending validation**
     - **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 (p. 3) Possible values are: **Yes | No**
     - **Domain name** – The first fully qualified domain name (FQDN) for the certificate.
     - **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 ACMintegrated 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**
     - **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:account:certificate/certificate_ID
```

The command returns information similar to the following:

```
{
    "Certificate": {
        "CertificateArn": "arn:aws:acm:region:account: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"
    }
}
```

Deleting certificates managed by ACM

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

> **Important**
> 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 Certificate Manager Private Certificate Authority User Guide*.

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.

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:account:certificate/certificate_ID
```

Installing ACM certificates

You cannot use ACM to install a public certificate directly on your AWS based website or application. You must use one of the services integrated with ACM. For more information, see Services integrated with AWS Certificate Manager (p. 3).

ACM certificates signed by a CA in ACM Private CA and intended for your private PKI can be exported and installed manually on any system where you have administrative access. These certificates are not trusted on the public internet.

Managed renewal for ACM certificates

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 (p. 33) *and* then exported or associated with another AWS service.
- NOT ELIGIBLE if it is a private certificate issued by calling the ACM Private CA `IssueCertificate` API.
- NOT ELIGIBLE if imported (p. 56).
- NOT ELIGIBLE if already expired.
- NOT ELIGIBLE for a certificate with a domain name that contains hyphens as its third and fourth characters. For example, the following domain name is not permitted:

```
ab--example.com
```

When ACM renews a certificate, the certificate's Amazon Resource Name (ARN) remains the same. Also, ACM certificates are regional resources (p. 3). If you have certificates for the same domain name in multiple AWS Regions, each of these certificates must be renewed independently.

Topics

Renewing publicly trusted 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 (p. 37) or email validation (p. 40). 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

Managed renewal is fully automated for ACM certificates that were originally issued using DNS validation (p. 37).

At 60 days prior to expiration, ACM checks for the renewal criteria:

- The certificate is currently in use by an AWS service.
- There is a valid DNS record containing the fully qualified domain name (FQDN).
- The required CNAME token is present and accessible in the DNS record.
- Each domain and subdomain that is named in the certificate is present in the DNS record.

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

If ACM cannot automatically validate a domain name, it notifies the domain owner that manual action is needed to validate the domain and complete certificate renewal. These notifications are sent at 45 days, 30 days, 15 days, seven days, three days, and one day prior to expiration. The most common reason for automatic validation to fail is that the required CNAME has been inadvertently changed or removed.

Renewal for domains validated by email

ACM certificates are valid for 13 months (395 days). To be renewed, email-validated certificates require an action by the domain owner. ACM begins sending renewal notices 45 days before expiration, using the domain's WHOIS mailbox addresses and to 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.

For more information about validation email messages, see Email validation (p. 40)

To learn how you can respond programmatically to validation email, see Automating email validation (p. 49).

Request a domain validation email message

After you have configured contact email addresses for your domain (see (Optional) Configure email for your domain (p. 27)), you can use the AWS Certificate Manager console or the ACM 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 (p. 53).
- You didn't receive or can't find the original domain validation email message that ACM sent for certificate renewal.

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:us-
east-2:111122223333:certificate/97b4deb6-8983-4e39-918e-ef1378924e1e
                                    --domain subdomain.example.com
                                    --validation-domain example.com
```

Automating email validation

ACM certificates that were email-validated normally require manual intervention 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 template used for domain validation email messages and the workflow involved in completing the validation process.

The validation email template

Validation email messages have the following format. The content of the highlighted strings should be replaced with values that are specific to the domain being validated.

```
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
```

Note
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 first domain it finds that includes an MX record. 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 default to the apex domain and to each subdomain. The domain owner needs to validate an email message for each of these domains before ACM can issue the certificate.

  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**.

Renewing certificates in a private PKI

ACM certificates that were signed by a private CA from ACM 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 ACM Private CA using the `IssueCertificate` action of the PCA 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 (p. 102).

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 Automating export of renewed certificates (p. 51)

Automating 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":"*"
      }
   ]
}
```

   c. Add application code to your function such as the following sample in Python. This code examines the event that triggers the function to call the ExportCertificate action. For more information, see Building Lambda functions with Python.

```
import json
import boto3

acm = boto3.client('acm')
passphrase = b'\x01\x02\x03\x04\x05\x06\x07\x08\x09\x00'

def lambda_handler(event, context):
    context.log("Incoming Event : " + json.dumps(event) + "\n");

    certificateArn = event['resources'][0];

    context.log("Renewed Certificate ARN : " + certificateArn + "\n");

    description = event['detail']['eventDescription'][0]['latestDescription'];

    if "completed the renewal" in description:
       response =
 acm.export_certificate(CertificateArn=certificateArn,Passphrase=passphrase)

       certificate = response['Certificate']
       certificateChain = response['CertificateChain']
       encrPrivateKey = response['PrivateKey']

       context.log("Certificate : " + certificate + "\n");
       context.log("Certificate Chain : " + certificateChain + "\n");
       context.log("Encrypted Private Key : " + encrPrivateKey + "\n");
```

```
        #TODO Install the private key, cert and chain to where it will be consumed


    return {
        'statusCode': 200,
    }
```

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) (p. 54).

   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.

Testing 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 by ACM 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 ACM 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-pca:region:account:\
 certificate/12345678-1234-1234-1234-123456789012
```

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-pca:region:account:\
 certificate/12345678-1234-1234-1234-123456789012
```

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

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, certificate renewal 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 (p. 48). 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 (p. 31).

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 certificate status to become available.

Topics

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 (p. 93).

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:123456789012:certificate/97b4deb6-8983-4e39-918e-ef1378924e1e
```

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**.

Importing 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 (p. 3). The certificates that you import work the same as those provided by ACM, with one important exception: ACM does not provide managed renewal (p. 47) 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 CloudWatch Events (p. 67).

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 (p. 3).

For information about how to import certificates into ACM, see the following topics. If you encounter problems importing a certificate, see Certificate import problems (p. 122).

Topics

Prerequisites for importing 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-AWScertificate authority (CA), you must also include the private key of the certificate, the public key of certificate, and the certificate chain. 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):

- 1024-bit RSA (`RSA_1024`)
- 2048-bit RSA (`RSA_2048`)
- 3072-bit RSA (`RSA_3072`)

- 4096-bit RSA (`RSA_4096`)
- Elliptic Prime Curve 256 bit (`EC_prime256v1`)
- Elliptic Prime Curve 384 bit (`EC_secp384r1`)
- Elliptic Prime Curve 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 supports 1024-bit RSA, 2048-bit RSA, and Elliptic Prime Curve 256-bit keys only, 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 by the private key of an issuing CA.

  If the certificate is self-signed, you must provide the private key. The private key must be no larger than 5 KB (5,120 bytes), and it must be unencrypted.

  If the certificate is signed by a CA, you must provide the certificate chain, and the cryptographic algorithm of the certificate must match the algorithm of the CA. For example, if the CA key type is RSA, then the certificate key type must also be RSA.
- 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 (p. 57).
- 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.

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 (private certificate only)

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-----
```

Importing 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 certificates (p. 56).

Topics

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. Choose **Review and import**.
5. On the **Review and import** page, check the displayed metadata about your certificate to ensure that it is what you intended. The fields include:

   - **Domains** — A list of fully qualified domain names (FQDN) authenticated by the certificate
   - **Expires in** — The number of days until the certificate expires
   - **Public key info** — The cryptographic algorithm used to generate the key pair
   - **Signature algorithm** — The cryptographic algorithm used to create the certificate's signature
   - **Can be used with** — A list of ACM integrated services that support the type of certificate you are importing

   If everything is correct, 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.

Reimporting 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 AWS Certificate Manager (p. 3).

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)

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. Choose **Review and import**.
8. 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.

Exporting a private certificate

You can export a certificate issued by ACM 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 ACM Private CA, see AWS Certificate Manager Private Certificate Authority User Guide.

Note
You cannot export a publicly trusted ACM certificate or its private key.

Topics

Exporting 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. Select the certificate that you want to export.
4. On the **Actions** menu, choose **Export (private certificates only)**.
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:account: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:account: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 > redirector to the previous example, yielding the following.

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

Tagging AWS Certificate Manager certificates

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

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

Monitoring and logging 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

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. AWS Certificate Manager supports the following CloudWatch metrics.

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

Metric	Description	Unit	Dimensions
`DaysToExpiry`	Number of days until a certificate expires. ACM stops publishing 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

Using CloudWatch Events

You can use Amazon 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 are delivered to CloudWatch Events in near-real time. You can write simple rules to indicate which events are of interest to you and the automated actions to take when an event matches a rule. For more information, see Creating a CloudWatch Events Rule That Triggers on an Event.

CloudWatch Events are turned into actions using Amazon EventBridge. With EventBridge, 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

This topic lists and describes the ACM related events supported by Amazon EventBridge.

AWS health events

AWS health events are generated for ACM certificates that are eligible for renewal. For information about renewal eligibility, see Managed renewal for ACM certificates (p. 47).

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"
      ]
   }
}
```

ACM expiration events

ACM sends daily expiration events for all 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 re-issued and re-imported 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 CloudWatch Events in ACM (p. 69).

Expiration events have the following structure.

```
{
  "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"
  }
}
```

Triggering actions with CloudWatch Events in ACM

You can create CloudWatch rules based on these events and use the CloudWatch console to configure actions that take place when the events are detected. This section provides sample procedures for configuring CloudWatch rules and resulting actions.

Topics

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 CloudWatch Events rule and trigger an action

1. Create a CloudWatch Events rule. For more information, see Creating a CloudWatch Events Rule That Triggers on an Event.

    a. In the CloudWatch console at https://console.aws.amazon.com/cloudwatch/, 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 (p. 68) 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 (p. 70).

Responding to an event with a Lambda function

This procedure demonstrates how to use AWS Lambda to listen on CloudWatch Events, 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 CloudWatch Events. 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 CloudWatch event #
>>>>>>> events

          Event matches CloudWatch 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 (p. 70)).
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 a CloudWatch Events rule to trigger the Lambda function. For more information, see Creating a CloudWatch Events Rule That Triggers on an Event.

   In the CloudWatch console at https://console.aws.amazon.com/cloudwatch/, navigate to **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:

- IAMpolicy 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

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 root 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)

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

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

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"
}
```

Using the API (Java examples)

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

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 ACM PCA 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.

Troubleshooting

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

Troubleshooting certificate requests

Consult the following topics if you encounter problems when requesting an ACM certificate.

Topics

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 DNS validation (p. 37) or Email validation (p. 40). 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

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 that you have a working email address that is registered in WHOIS and that the address is visible when performing a standard WHOIS lookup for the domain names in the certificate request. Typically, you do this through your domain registrar.
- 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 (p. 40).

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 (Optional) Configure email for your domain (p. 27) or Not receiving validation email (p. 114). 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: Domain Not Allowed

One or more of the domain names in the certificate request was reported as an unsafe domain by VirusTotal. To correct the problem, try the following:

- Search for your domain name on the VirusTotal website. If your domain is reported as suspicious, see Google Help for Hacked Websites to learn what you can do.
- If you believe that the result is a false positive, notify the organization that is reporting the domain. VirusTotal is an aggregate of several antivirus and URL scanners and cannot remove your domain from a blacklist itself.

After you correct the problem and the VirusTotal registry has been updated, request a new certificate.

If you see this error and your domain is not included in the VirusTotal list, visit the AWS Support Center and create a case. If you don't have a support agreement, post a message to the ACM Discussion Forum.

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 AWS 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 AWS Support. If you don't have a support plan, post a new thread in the ACM Discussion Forum.

Troubleshooting 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 registered WHOIS contact addresses and other common email addresses for the requested domain. For more information, see Email validation (p. 40). If you chose DNS validation, you must write the CNAME record that ACM created for you to your DNS database. For more information, see DNS validation (p. 37).

> **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 (p. 114). 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 (p. 5), 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 DNS validation problems.

Topics

Troubleshoot DNS validation problems

Consult the following guidance if you are having trouble validating a certificate with DNS.

> **Tip**
> 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
> - **whois** — Linux, Windows

Topics

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 record 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 record**

in Route 53 button may not be available when you expect it. If this happens, check for the following possible causes.

- On the **Validation** page, you did not click the down-arrow next to your domain name.
- 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 VPCprivate hosted zone, untrusted domains in your private PKI, and self-signed certificates.

AWS does provide support for publicly untrusted domains through the ACM Private CA service.

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

When you request a certificate from ACM and choose email validation, domain validation email is sent to three contact addresses specified in WHOIS and five common administrative addresses. For more information, see Email validation (p. 40). If you are experiencing problems receiving validation email, review the suggestions that follow.

Where to look for email

Validation email is sent to contact addresses listed in WHOIS and to common administrative addresses for the domain. Email is not sent to the AWS account owner unless the owner is also listed as a domain contact in WHOIS. 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

In some cases, the domain registrant, technical, and administrative contacts in WHOIS may not be publicly available, and AWS therefore cannot reach these contacts. At your discretion, you can choose to configure your registrar to list your email address in WHOIS, although not all registrars support this option. You may be required to make a change directly at your domain's registry. In other cases, the domain contact information may be using a privacy address, such as those provided through WhoisGuard or PrivacyGuard.

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

If email contact information for your domain is not available through WHOIS, or if email sent to the contact information does not reach the domain owner or an authorized representative, we recommend that you configure your domain or subdomain to receive email sent to one or more of the common administrative addresses formed by prepending admin@, administrator@, hostmaster@, webmaster@, and postmaster@ to the requested domain name. For more information about configuring email for your domain, see the documentation for your email service provider and follow the instructions at (Optional) Configure email for your domain (p. 27). If you are using Amazon WorkMail, see Working with Users in the Amazon WorkMail Administrator Guide.

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.

Missing or incorrectly configured MX records

An MX record is a resource record in the Domain Name System (DNS) database that specifies one or more mail servers that accept email messages for your domain. If your MX record is missing or misconfigured, email can not be sent to any of the five common system administration addresses specified at Email validation (p. 40). Fix your missing or misconfigured MX record and try to resend the email or request your certificate again.

> **Note**
> Currently, we recommend that you wait at least one hour before attempting to resend the email or requesting your certificate.

Note
To bypass requiring an MX record, you can use the `ValidationDomain` option in the RequestCertificate API or the request-certificate AWS CLI command to specify the domain name to which ACM sends validation emails. If you use the API or the AWS CLI, AWS does not perform an MX lookup.

Contact the Support Center

If, after reviewing the preceding guidance, you still don't receive the domain validation email, please visit the AWS Support Center and create a case. If you don't have a support agreement, post a message to the ACM Discussion Forum.

Email sent to subdomain

If you are using the console and request a certificate for a subdomain name such as `sub.test.example.com`, then ACM checks to see if there is an MX record for `sub.test.example.com`. If not, then the parent domain `test.example.com` is checked, and so on, up to the base domain `example.com`. If an MX record is found, the search stops and a validation email is sent to the common administration addresses for the subdomain. So for example, if an MX record is found for `test.example.com`, email is sent to admin@test.example.com, administrator@test.example.com, and the other administrative addresses specified in Email validation (p. 40). If an MX record is not found in any of the subdomains, email is sent to the subdomain that you originally requested the certificate for. For a thorough discussion of how to set up your email and how ACM works with DNS and the WHOIS database, see (Optional) Configure email for your domain (p. 27).

Instead of using the console, you can use the `ValidationDomain` option in the RequestCertificate API or the request-certificate AWS CLI command to specify the domain name to which ACM sends validation emails. If you use the API or the AWS CLI, AWS does not perform an MX lookup.

Hidden contact information

A common problem occurs when you attempt to create a new certificate. Some registrars allow you to hide your contact information in your WHOIS listing. Others allow you to substitute your real email address with a privacy (or proxy) address. This prevents you from receiving validation email at your registered contact addresses.

To receive mail, ensure that your contact information is public in WHOIS, or if your WHOIS listing shows a privacy email address, ensure that email sent to the privacy address is forwarded to your real email address. After your WHOIS setup is complete and as long as your certificate request has not timed out, you can choose to resend the validation email. ACM performs a new WHOIS/MX lookup and sends validation email to your now public contact address.

Certificate renewals

If you made your WHOIS information public when you requested a new certificate and then later obfuscated your information, ACM cannot retrieve your registered contact addresses when you attempt to renew your certificate. ACM sends validation email to these contact addresses and to five common administrative addresses formed by using your MX record. To address this problem, make your WHOIS information public again and resend the validation emails. ACM performs a new WHOIS/MX lookup and sends validation email to your now public contact addresses.

WHOIS throttling

Sometimes ACM is unable to contact the WHOIS server even after you have sent multiple requests for validation email. This problem is external to AWS. That is, AWS does not control the WHOIS servers

and cannot prevent WHOIS server throttling. If you experience this problem, create a case at the AWS Support Center for help with a workaround.

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.

Troubleshoot problems with the .IO top-level domain

The .IO top-level domain is assigned to the British Indian Ocean Territory. Currently, the domain registry does not display your public information from the WHOIS database. This is true whether you have privacy protection for the domain enabled or disabled. When a WHOIS lookup is performed, only obfuscated registrar information is returned. Therefore, ACM is unable to send validation email to the following three registered contact addresses that are usually available in WHOIS.

- Domain registrant
- Technical contact
- Administrative contact

ACM does, however, send validation email to the following five common system addresses where *your_domain* is the domain name you entered when you initially requested a certificate and `.io` is the top level domain.

- administrator@*your_domain*.io
- hostmaster@*your_domain*.io
- postmaster@*your_domain*.io
- webmaster@*your_domain*.io
- admin@*your_domain*.io

To receive validation mail for an .IO domain, make sure that you have one of the preceding five email accounts enabled. If you do not, you will not receive validation email and you will not be issued an ACM certificate.

> **Note**
> We recommend that you use DNS validation rather than email validation. For more information, see DNS validation (p. 37).

Troubleshooting 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 renewal for ACM certificates (p. 47).

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 AWS Certificate Manager (p. 3).

- 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 Email validation (p. 40).
- For DNS-validated certificates, make sure that your DNS configuration contains the correct CNAME records as described in DNS validation (p. 37).

Handling failures in managed certificate renewal

When a certificate is 60 days away from expiration, ACM automatically attempts to renew it every hour. If ACM is unable to renew the certificate after 15 days, you will receive an email with further instructions on how to manually fix the renewal problem. This process differs depending on how the certificate was originally validated.

Managed certificate renewal for email-validated certificates

ACM certificates are valid for 13 months (395 days). To be renewed, email-validated certificates require an action by the domain owner. ACM begins sending renewal notices 45 days before expiration, using the domain's WHOIS mailbox addresses and to 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.

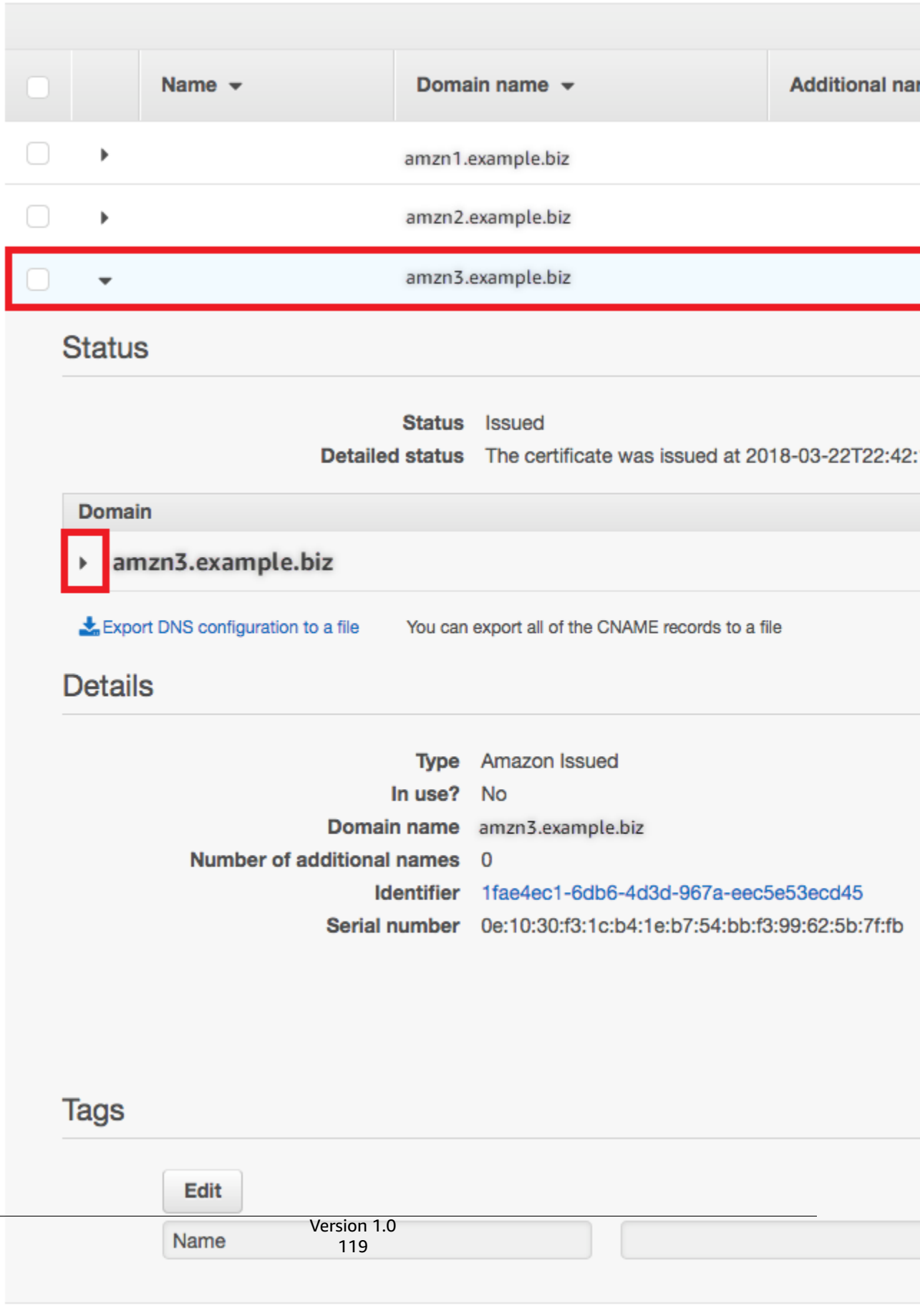
See Validate with Email (p. 40) for instructions on identifying which domains are in the `PENDING_VALIDATION` state and repeating the validation process for those domains.

Managed certificate renewal for DNS-validated certificates

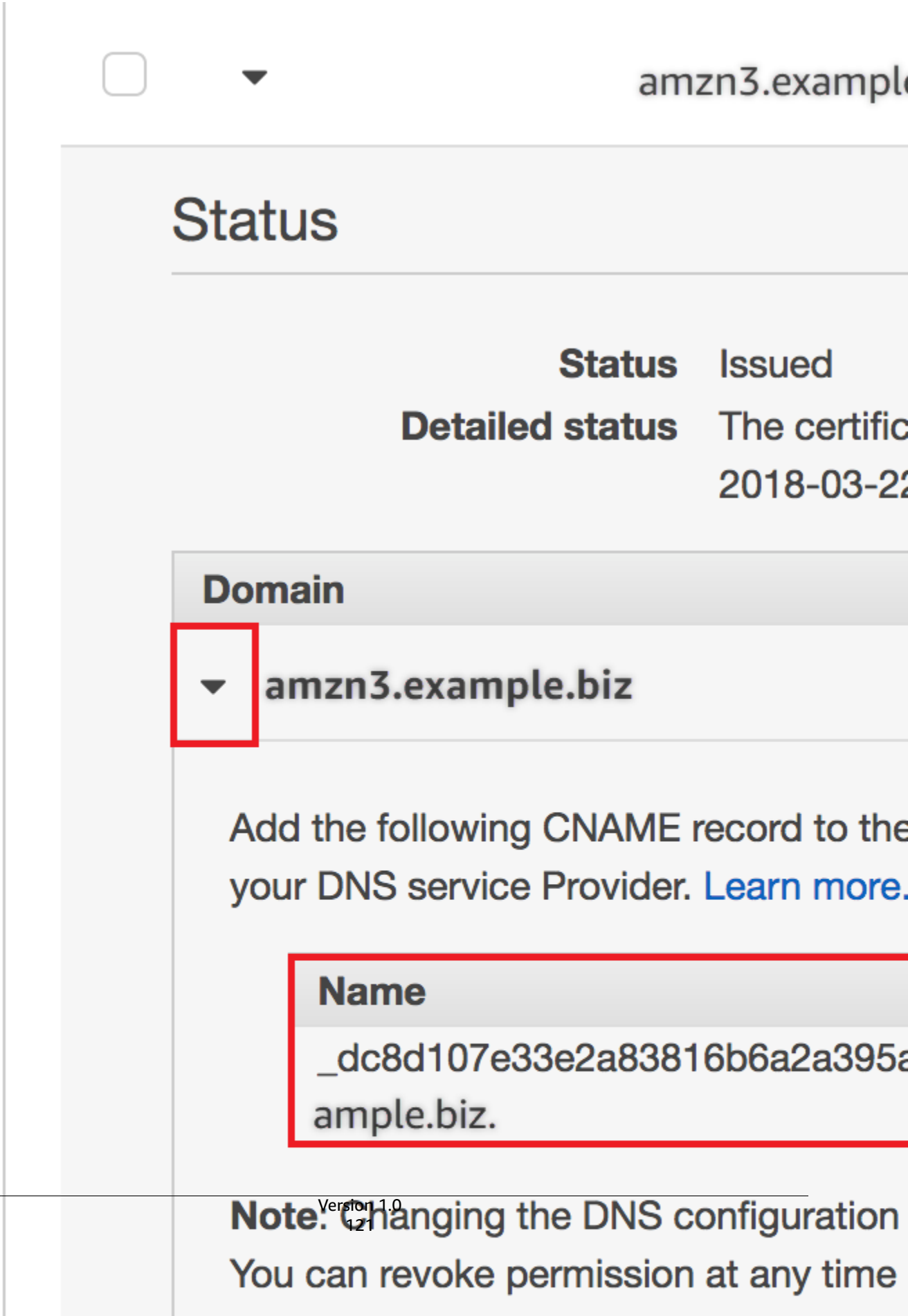
ACM does not attempt TLS validation for DNS-validated certificates. If ACM fails to renew a certificate you validated with DNS validation, it is most likely due to missing or inaccurate CNAME records in your DNS configuration. If this occurs, ACM notifies you that the certificate could not be renewed 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 DNS validation (p. 37).


Name
Domain name
Additional na
amzn1.example.biz
amzn2.example.biz
amzn3.example.biz
Status
Status
Issued
Detailed status
The certificate was issued at 2018-03-22T22:42:
Domain
amzn3.example.biz
Export DNS configuration to a file
You can export all of the CNAME records to a file
Details
Type
Amazon Issued
In use?
No
Domain name
amzn3.example.biz
Number of additional names
0
Identifier
1fae4ec1-6db6-4d3d-967a-eec5e53ecd45
Serial number
0e:10:30:f3:1c:b4:1e:b7:54:bb:f3:99:62:5b:7f:fb
Tags
Edit
Name

Choose the target certificate from the console.


amzn3.example
Status
Status
Issued
Detailed status
The certific
2018-03-22
Domain
amzn3.example.biz
Add the following CNAME record to the
your DNS service Provider. Learn more.
Name
_dc8d107e33e2a83816b6a2a395a
ample.biz.
Note: Changing the DNS configuration
You can revoke permission at any time

Expand the certificate window to find the certificate's CNAME information.

If the problem persists, contact the Support Center.

Understanding renewal timing

Managed renewal for ACM certificates (p. 47) 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.

Troubleshooting other problems

This section includes guidance for problems not related to issuing or validating ACM certificates.

Topics

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 at least one of your CAA records must contain one of the following domain names:

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

If you do not want ACM to perform CAA checking, do not configure a CAA record for your domain or leave your CAA records blank. For more information about creating a CAA record, see (Optional) Configure a CAA record (p. 28).

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 (p. 23), 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 (p. 56) 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 AWS 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 ACM 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 ACM Private CA.

Note
ACM 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 PCA `AccessDeniedException` failure code.

ACM failure code	Comment
	To remedy the problem, grant the necessary permissions to the ACM service principal using the PCA CreatePermission operation.
PCA_INVALID_DURATION	The validity period of the requested certificate exceeds the validity period of the issuing private CA. This triggers a PCA 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 PCA InvalidStateException failure code. Resolve the issue as follows: • If the CA has the status CREATING, wait for creation to finish and then install the CA certificate. • If the CA has status PENDING_CERTIFICATE, install the CA certificate. • If the CA has status DISABLED, update it to ACTIVE status. • If the CA has status DELETED, restore it. • If the CA has status EXPIRED, install a new certificate • If the CA has status FAILED, and you cannot resolve the issue, contact AWS Support.
PCA_LIMIT_EXCEEDED	The private CA has reached an issuance quota. This triggers a PCA LimitExceededException failure code. Try repeating your request before proceeding with this help. If the error persists, contact AWS Support to request a quota increase.
PCA_REQUEST_FAILED	A network or system error occurred. This triggers a PCA RequestFailedException failure code. Try repeating your request before proceeding with this help. If the error persists, contact AWS Support.
PCA_RESOURCE_NOT_FOUND	The private CA has been permanently deleted. This triggers a PCA 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 (p. 19).

Concepts

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

Topics

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 (p. 130).

Asymmetric Key Cryptography

Unlike Symmetric Key Cryptography (p. 132), 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 ACM private key security (p. 11).

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 (p. 24).

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 DNS validation (p. 37).

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. Also, subdomain names can have multiple levels. For example, in `jake.dog.animals.example.com`, the subdomains are `jake`, `dog`, and `animals` in that order.

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.

Registration

The right to use a domain name is delegated by domain name registrars. Registrars are typically accredited by the Internet Corporation for Assigned Names and Numbers (ICANN). In addition, other organizations called registries maintain the TLD databases. When you request a domain name, the registrar sends your information to the appropriate TLD registry. The registry assigns a domain name, updates the TLD database, and publishes your information to WHOIS. Typically, domain names must be purchased.

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 (p. 130).

Public Key Infrastructure

A public key infrastructure (PKI) consists of hardware, software, people, policies, documents, and procedures that are needed to create, issue, manage, distribute, use, store, and revoke digital certificates. PKI facilitates the secure transfer of information across computer networks.

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 (p. 129).

Transport Layer Security (TLS)

See Secure Sockets Layer (SSL) (p. 132).

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.

Document history

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

update-history-change	update-history-description	update-history-date
Updating key algorithm types for import (p. 134)	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. (p. 134)	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 (p. 134)	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 (p. 134)	Added cross-account support for using private CAs from ACM Private CA. For more information, see https://docs.aws.amazon.com/acm/latest/userguide/ca-access.html.	August 17, 2020
Added region support (p. 134)	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 (p. 134)	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 (p. 134)	Added ability to publish ACM public certificates into certificate transparency logs by default.	April 24, 2018
Launching ACM Private CA (p. 134)	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 (p. 134)	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 DNS validation (p. 37).	November 21, 2017
New content	Added new Java code examples to Using the API (Java examples) (p. 90).	October 12, 2017
New content	Added information about CAA records to (Optional) Configure a CAA record (p. 28).	September 21, 2017
New content	Added information about .IO domains to Troubleshooting (p. 110).	July 07, 2017
New content	Added information about reimporting a certificate to Reimporting a certificate (p. 60).	July 07, 2017
New content	Added information about certificate pinning to Best practices (p. 22) and to Troubleshooting (p. 110).	July 07, 2017
New content	Added AWS CloudFormation to Services integrated with AWS Certificate Manager (p. 3).	May 27, 2017

Change	Description	Release Date
Update	Added more information to Quotas (p. 6).	May 27, 2017
New content	Added documentation about Identity and access management for AWS Certificate Manager (p. 11).	April 28, 2017
Update	Added a graphic to show where validation email is sent. See Email validation (p. 40).	April 21, 2017
Update	Added information about setting up email for your domain. See (Optional) Configure email for your domain (p. 27).	April 6, 2017
Update	Added information about checking certificate renewal status in the console. See Check a certificate's renewal status (p. 53).	March 28, 2017
Update	Updated the documentation for using Elastic Load Balancing.	March 21, 2017
New content	Added support for AWS Elastic Beanstalk and Amazon API Gateway. See Services integrated with AWS Certificate Manager (p. 3).	March 21, 2017
Update	Updated the documentation about Managed renewal (p. 47).	February 20, 2017
New content	Added documentation about Import certificates (p. 56).	October 13, 2016
New content	Added AWS CloudTrail support for ACM actions. See Using CloudTrail with AWS Certificate Manager (p. 75).	March 25, 2016
New guide	This release introduces AWS Certificate Manager.	January 21, 2016