Cloud Certificate Manager

# Best Practices

**Issue** 07

**Date** 2024-12-17

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

# Contents

# 1 CCM Best Practice Summary

This document describes the common application scenarios of CCM and provides detailed solution description and operation guide for each scenario to help you deploy and manage certificates in different scenarios.

## Best Practices

**Table 1-1** Best practices

| Category | Related Document |
|---|---|
| SSL Certificate | **Resolving a DNS Record on Huawei Cloud or Alibaba Cloud** |
| | **Enabling HTTPS Encryption for Websites** |
| | **Deploying SSL Certificates to the Cloud in One Click** |
| | **Using FunctionGraph to Automatically Obtain and Update ECS Server Certificates** |
| Private Certificate | **Best Practices for Private Certificate Management** |
| | **Best Practices for Private CA Management** |
| | **Best Practices of PCA Code Examples** |
| | **Building an Internal Identity Authentication System for Your Organization** |

# 2 Best Practices for SSL Certificate Manager

## 2.1 Resolving a DNS Record on Huawei Cloud or Alibaba Cloud

After you submit an SSL certificate application to the CA, you need to verify domain name ownership as required by the CA. This topic describes how to verify your domain name ownership by DNS.

### Background

After you apply for an SSL certificate from a CA, you are required to verify the domain name ownership. You need to work with the CA to complete the domain name ownership verification. After your ownership of the domain name is verified by you and approved by the CA, the status of your certificate will change.

If you do not complete the domain ownership verification, your certificate will remain in the **Pending domain name verification** state.

Domain name ownership verification by DNS is to verify domain ownership by resolving a specific DNS record on the platform hosting the domain name. When you apply for a certificate and select **DNS** for **Domain Name Verification Method**, follow the instructions in this part to complete the verification.

### Procedure

**Step 1** Obtain the host record and record value of a certificate. For details, see **Obtaining the Host Record and Record Value of a Certificate**.

**Step 2** Perform domain name ownership verification by DNS.

Domain name ownership verification by DNS is to resolve DNS records, which can be performed only on the domain name management platform that hosts your domain name. The following examples are for your reference.

- If your domain name is hosted in the DNS service on Huawei Cloud, complete the resolution by following the instructions in **Huawei Cloud DNS TXT Resolution**.
- If your domain name is hosted in the DNS service on Alibaba Cloud, complete the resolution by following the instructions in **Alibaba Cloud DNS TXT Resolution**.

**Step 3** Check whether the ownership verification takes effect. For details, see **Verifying DNS Configurations**.

**----End**

## Obtaining the Host Record and Record Value of a Certificate

**Step 1** Log in to the **management console**.

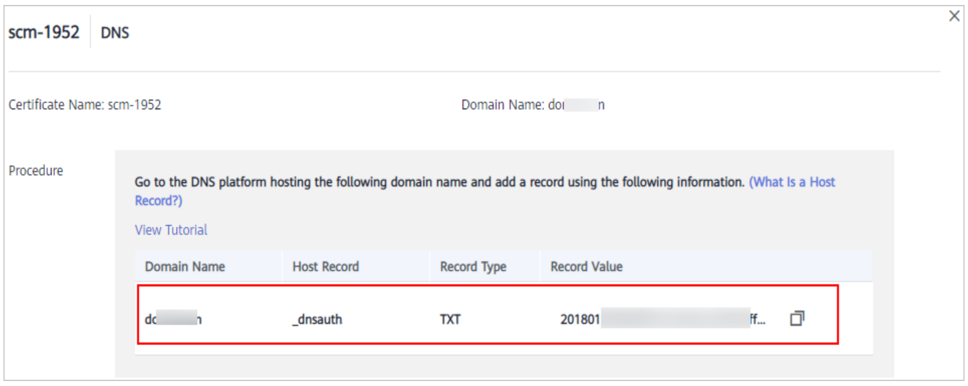
**Step 2** Click ☰ in the upper left corner of the page and choose **Security & Compliance** > **Cloud Certificate Management Service**. The service console is displayed.

**Step 3** In the navigation pane on the left, choose **SSL Certificate Manager**. In the row containing the desired certificate, click **Verify Domain Name** in the **Operation** column. The **Verify Domain Name** page is displayed.

**Step 4** On the **Verify Domain Name** page, view the content for **Host Record**, **Record Type**, and **Record Value**. **Figure 2-1** shows an example.

If **Host Record**, **Record Type**, and **Record Value** are not displayed, log in to the mailbox to view. The mailbox is the one you provide during certificate application.

**Figure 2-1** Viewing a host record



scm-1952 DNS

Certificate Name: scm-1952 Domain Name: do[redacted]n

Procedure

Go to the DNS platform hosting the following domain name and add a record using the following information. (What Is a Host Record?)

View Tutorial

| Domain Name | Host Record | Record Type | Record Value |
|---|---|---|---|
| dc[redacted]n | _dnsauth | TXT | 201801[redacted]ff... |

**----End**

## Huawei Cloud DNS TXT Resolution

Refer to this part if you are managing your domain name on Huawei Cloud.

**Step 1** Log in to the **management console**.

**Step 2** Choose **Domain Name Service** under **Network** to go to the **Domain Name Service** page.

**Step 3** In the navigation pane on the left, choose **DNS Resolution** > **Public Zones**.

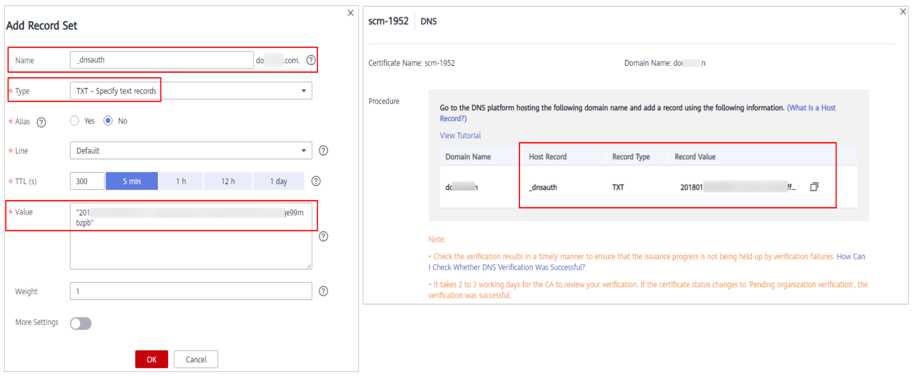
**Step 4** In the domain name list on the **Public Zones** page, click the added domain name (or the primary domain name for a multi-domain certificate) to go to the record set page.

**Step 5** In the upper right corner of the page, click **Add Record Set**. **Figure 2-2** shows an example.

NOTE

If there is a DNS record of domain name **domain3.com** in the domain name list, click **Modify** in the **Operation** column. Modify the record in the displayed **Modify Record Set** dialog box.

**Figure 2-2** Adding a record set



Add Record Set

Name: _dnsauth do[redacted].com.
* Type: TXT – Specify text records
* Alias: Yes [✓ No]
* Line: Default
* TTL (s): 300 [✓ 5 min] 1 h 12 h 1 day
* Value: "201[redacted]je99m bzpb"
Weight: 1
More Settings

scm-1952 | DNS

Certificate Name: scm-1952
Domain Name: do[redacted]n

Procedure

Go to the DNS platform hosting the following domain name and add a record using the following information. (What Is a Host Record?)

View Tutorial

| Domain Name | Host Record | Record Type | Record Value |
|---|---|---|---|
| dc[redacted]n | _dnsauth | TXT | 201801[redacted]ff... |

Note:

• Check the verification results in a timely manner to ensure that the issuance progress is not being held up by verification failures. How Can I Check Whether DNS Verification Was Successful?

• It takes 2 to 3 working days for the CA to review your verification. If the certificate status changes to 'Pending organization verification', the verification was successful.

**Table 2-1** Parameters for adding a record set

| Parameter | Description |
|---|---|
| Name | Host record returned by the domain name service provider on the domain name verification page of the certificate. |
| Type | Record type returned by the domain name service provider on the domain name verification page. |
| Alias | Select **No**. |
| Line | Select **Default**. |
| TTL (s) | Set this parameter to **5 min**. A larger TTL value indicates less frequency of DNS record synchronization and update. |
| Value | Record value returned by the domain name service provider on the domain name verification page of the certificate.<br>**NOTE**<br>Record values must be quoted with quotation marks and then pasted in the text box. |
| Keep other settings unchanged. | |

**Step 6** Click **OK**.

If the status of the record set is **Normal**, the record set is added successfully.

NOTE

- DNS configuration records can be deleted only after the certificate is issued or revoked.
- Check whether the DNS record is correctly configured. If not, the certificate cannot be issued.

**Step 7** After the verification is complete, additional time is required for the CA to verify your domain name. During this period, the certificate is in the **Pending domain name verification** state.

The certificate enters the **Pending organization verification** state only after the CA has confirmed your domain ownership.

**----End**

## Alibaba Cloud DNS TXT Resolution

If your domain name is hosted in Alibaba Cloud, you need to add a record through the Alibaba Cloud DNS console to complete the verification.

**Step 1** Log in to the Alibaba Cloud DNS console.

**Step 2** On the **Manage DNS** page, click the **Domains** tab and click the name of the domain name for which you want to configure a record.

**Step 3** Click **Add Record** and provide the following information:

- **Record Type**: Enter the record type obtained in **Obtaining the Host Record and Record Value of a Certificate**.
- **Host**: Enter the prefix of the host record obtained in **Obtaining the Host Record and Record Value of a Certificate**.
- **ISP line**: Select **Default**. You must specify an ISP line. Otherwise, your domain name may become inaccessible to some users.
- **Value**: TXT record from the SCM console. To obtain the record, refer to **Obtaining the Host Record and Record Value of a Certificate**.
- **TTL**: cache time. The smaller the value, the faster the modification takes effect. The default value is 600 seconds.

**Step 4** Click **OK**.

**----End**

## Verifying DNS Configurations

Select a command based on the record type and OS and check whether the DNS configuration takes effect.

Use TXT record **_dnsauth.domain.com** as an example.

- For TXT records
  - For Windows OSs:
    ```
    nslookup -q=TXT _dnsauth.domain.com
    ```
  - For Linux OSs:
    ```
    dig TXT _dnsauth.domain.com
    ```
  - For macOS OSs:
    ```
    dig TXT _dnsauth.domain.com
    ```

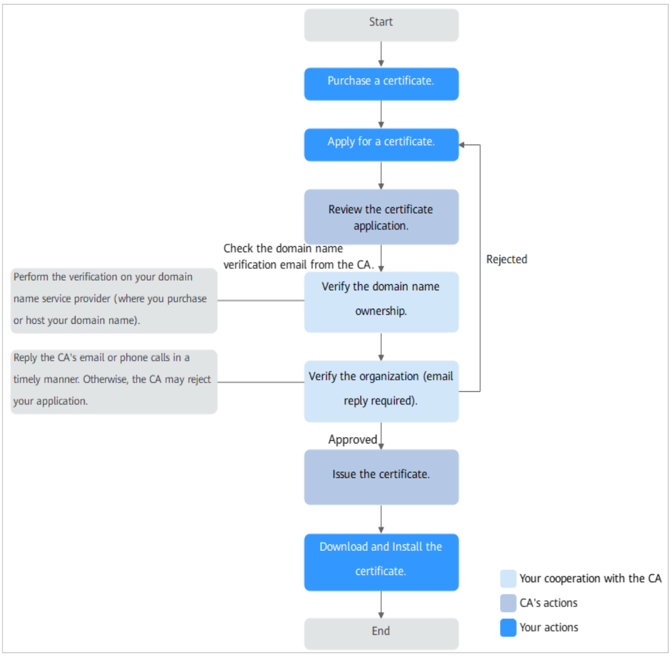
- For CNAME records
  - Windows OSs
    ```
    nslookup -q=CNAME _dnsauth.domain.com
    ```
  - Linux OSs
    ```
    dig CNAME _dnsauth.domain.com
    ```
  - macOS OSs
    ```
    dig CNAME _dnsauth.domain.com
    ```

If the value recorded in the command output (value of **text**) is the same as that returned by the domain name service provider, the configuration of domain name ownership verification has taken effect.

## 2.2 Enabling HTTPS Encryption for Websites

An SSL certificate secures the communication between the website and the client and authenticates the identity of the website. This topic walks you through how to enable HTTPS-encrypted communication with SSL certificates from the scratch.

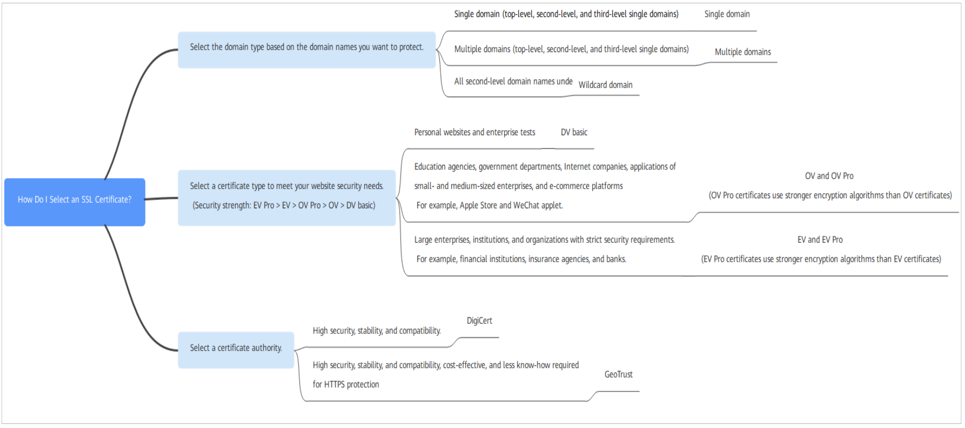
**Figure 2-3** Certificate Usage Process


Start
Purchase a certificate.
Apply for a certificate.
Review the certificate application.
Check the domain name verification email from the CA.
Rejected
Perform the verification on your domain name service provider (where you purchase or host your domain name).
Verify the domain name ownership.
Reply the CA's email or phone calls in a timely manner. Otherwise, the CA may reject your application.
Verify the organization (email reply required).
Approved
Issue the certificate.
Download and Install the certificate.
End
Your cooperation with the CA
CA's actions
Your actions

### Procedure

**Step 1** Buy an SSL certificate by referring to **Purchasing an SSL Certificate**.

**Figure 2-4** Certificate selection


How Do I Select an SSL Certificate?
Select the domain type based on the domain names you want to protect.
Single domain (top-level, second-level, and third-level single domains)
Single domain
Multiple domains (top-level, second-level, and third-level single domains)
Multiple domains
All second-level domain names unde
Wildcard domain
Select a certificate type to meet your website security needs.
(Security strength: EV Pro > EV > OV Pro > OV > DV basic)
Personal websites and enterprise tests
DV basic
Education agencies, government departments, Internet companies, applications of small- and medium-sized enterprises, and e-commerce platforms
For example, Apple Store and WeChat applet.
OV and OV Pro
(OV Pro certificates use stronger encryption algorithms than OV certificates)
Large enterprises, institutions, and organizations with strict security requirements.
For example, financial institutions, insurance agencies, and banks.
EV and EV Pro
(EV Pro certificates use stronger encryption algorithms than EV certificates)
Select a certificate authority.
High security, stability, and compatibility.
DigiCert
High security, stability, and compatibility, cost-effective, and less know-how required for HTTPS protection
GeoTrust

**Step 2** After you pay for your order, apply for the certificate. During this period, associate the certificate with the domain name you want to protect, fill in application forms with your details, and submit the application to the CA. For details, see **Applying for an SSL Certificate**.

**Step 3** After receives your application, the CA sends a domain ownership verification email to the email address you provided in the application. When you receive this email, verify your domain name ownership as instructed. For details, see **Verifying Domain Name Ownership**.

**Step 4** If you apply for an OV or EV certificate, the CA sends an organization verification email after domain name ownership is verified. The CA validates your organization identity by contacting you through the method you select. For more details, see **Verifying the Organization**.

**Step 5** The validation period varies depending on certificate types. For details, see **Certificate validation periods**.

NOTE

Your cooperation with the CA is required during domain name and organization verification. A timely response to the CA will get your certificate issued more quickly.

**Table 2-2** Certificate approval periods

| Certificate | Approval Period |
|---|---|
| EV | The CA manually reviews the information.<br>If the information is valid, the review takes 7 to 10 **working days**. |
| OV | The CA manually reviews the information.<br>If the information is valid, the review takes 3 to 5 **working days**. |

| Certificate | Approval Period |
|---|---|
| DV | No manual review is required.<br>The CA system automatically checks domain name ownership. The certificate can be issued within several hours if the CA validates your ownership. You need to ensure that your DNS configurations are valid. |

**Step 6** After the certificate is issued, download the certificate to the local PC. For details, see **Downloading an SSL Certificate**.

**Step 7** The procedure for installing an SSL certificate varies depending on the web server. The following describes how to install an SSL certificate on mainstream web servers.

- **Installing an SSL Certificate on a Tomcat Server**
- **Installing an SSL Certificate on an Nginx Server**
- **Installing an SSL Certificate on an Apache Server**
- **Installing an SSL Certificate on an IIS Server**
- **Installing an SSL Certificate on a WebLogic Server**

**----End**

### Helpful Links

- **Differences Between SSL Certificate Types**
- **How Do I Enter a Domain Name for a Certificate When Applying for an SSL Certificate?**
- **Why Does the SSL Certificate Remain in the Pending Domain Name Verification State (Application Progress Is 40%) After Domain Name Verification Is Complete?**
- **How Do I Handle the Email or Phone Call from the CA?**

## 2.3 Deploying SSL Certificates to the Cloud in One Click

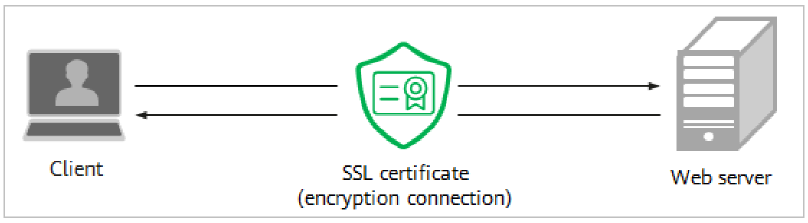
### 2.3.1 Scenario

With CCM, you can quickly deploy an SSL certificate you obtain through CCM or a third-party platform to your CDN, WAF, or ELB instance on Huawei Cloud, converting your services from HTTP into HTTPS and improving data access security.

#### Working Principles of an SSL Certificate

An SSL certificate is used in establishing encryption channels between the web server and browser and between the web server and client. The HTTPS protocol is enabled by configuring and applying SSL certificates to ensure the security of data transmission over Internet.

**Figure 2-5** Working principles of an SSL certificate



## 2.3.2 Deploying an SSL Certificate to CDN in One Click

### Prerequisites

- You have an SSL certificate that is in **Issued** or **Hosted** status in CCM.
- You have enabled Content Delivery Network (CDN).

### Notes and Constraints

To deploy a certificate to an accelerated domain name in CDN, HTTPS must be enabled for the domain name, or the certificate cannot be deployed for it.

### Adding an Acceleration Domain Name

Before deploying an SSL certificate to CDN, you need to add the domain name that will use the SSL certificate as a CDN acceleration domain name. For details, see **Adding a Domain Name**.

### (Optional) Upload Private Key

If you select **Upload a CSR** for **CSR** when applying for a certificate, you need to upload the private key to the cloud to deploy the issued certificate to other cloud services in one-click mode because the cloud does not have the private key of the certificate. If you select **System generated CSR** for **CSR** during certificate application, skip this step.

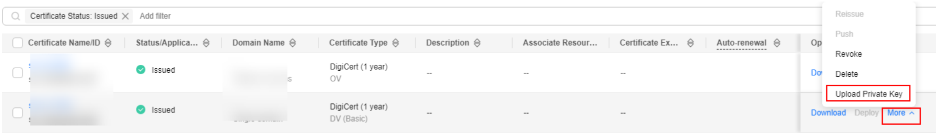
**Step 1** Log in to the **management console**.

**Step 2** Click ☰ in the upper left corner of the page and choose **Security & Compliance** > **Cloud Certificate Management Service**. The service console is displayed.

**Step 3** In the navigation pane on the left, choose **SSL Certificate Manager** > **SSL Certificates**.

**Step 4** In the **Operation** column of the target certificate, choose **More** > **Upload Private Key**.

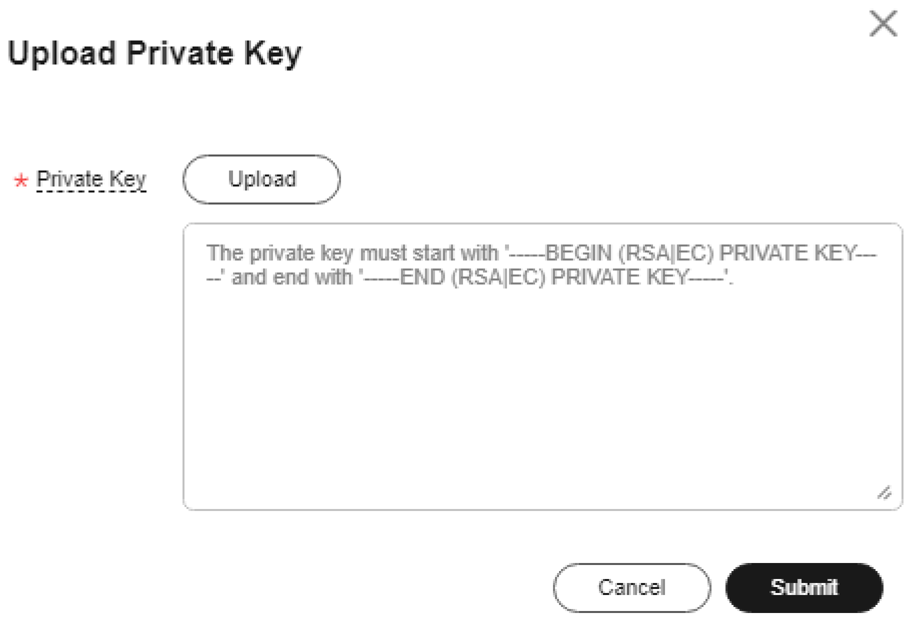
**Figure 2-6** Uploading the private key


Certificate Status: Issued Add filter

| Certificate Name/ID | Status/Applica... | Domain Name | Certificate Type | Description | Associate Resour... | Certificate Ex... | Auto-renewal | Op |
|---|---|---|---|---|---|---|---|---|
| [redacted] | Issued | [redacted] | DigiCert (1 year) OV | -- | -- | -- | | Do |
| [redacted] | Issued | [redacted] | DigiCert (1 year) DV (Basic) | -- | -- | -- | | Download Deploy More |

**Step 5** In the displayed dialog box, click **Upload** and select a local private key file, or enter the certificate private key information in the text box according to the format.

**Figure 2-7** Uploading the private key file



Upload Private Key

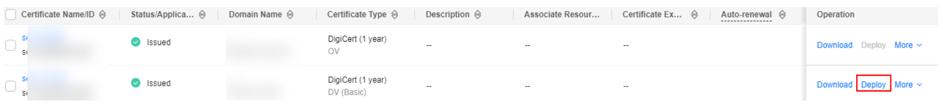
* Private Key Upload

The private key must start with '-----BEGIN (RSA|EC) PRIVATE KEY-----' and end with '-----END (RSA|EC) PRIVATE KEY-----'.

**Step 6** Click **Submit**.

After the private key is uploaded successfully, the **Deploy** button of the certificate becomes available.

**Figure 2-8** Private key uploaded



| Certificate Name/ID | Status/Applica... | Domain Name | Certificate Type | Description | Associate Resour... | Certificate Ex... | Auto-renewal | Operation |
|---|---|---|---|---|---|---|---|---|
| | Issued | | DigiCert (1 year) OV | -- | -- | -- | | Download Deploy More |
| | Issued | | DigiCert (1 year) DV (Basic) | -- | -- | -- | | Download Deploy More |

**----End**

## Deploying an SSL Certificate to CDN

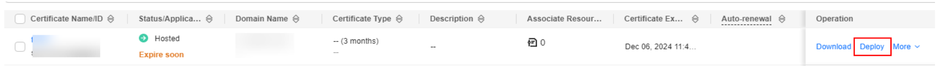
**Step 1** Log in to the **management console**.

**Step 2** Click ☰ in the upper left corner of the page and choose **Security & Compliance** > **Cloud Certificate Management Service**. The service console is displayed.

**Step 3** In the navigation pane on the left, choose **SSL Certificate Manager** > **SSL Certificates**.

**Step 4** Locate the row containing the certificate you want to deploy on other cloud product, and click **Deploy** in the **Operation** to go to the certificate deployment details page.

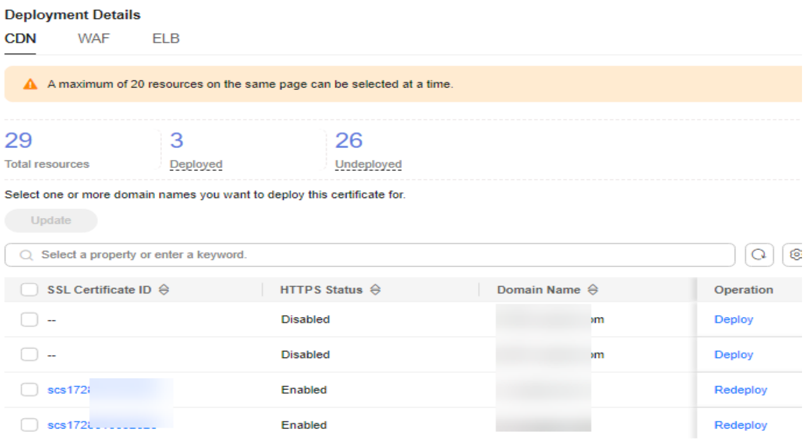
**Figure 2-9** Deploying a certificate



| Certificate Name/ID | Status/Applica... | Domain Name | Certificate Type | Description | Associate Resour... | Certificate Ex... | Auto-renewal | Operation |
|---|---|---|---|---|---|---|---|---|
| | Hosted Expire soon | | -- (3 months) -- | -- | 0 | Dec 06, 2024 11:4... | | Download Deploy More |

**Step 5** On the displayed page, select **CDN** in the **Deployment Details** area.

**Figure 2-10** Selecting CDN



Deployment Details

CDN WAF ELB

A maximum of 20 resources on the same page can be selected at a time.

29 Total resources | 3 Deployed | 26 Undeployed

Select one or more domain names you want to deploy this certificate for.

Update

Select a property or enter a keyword.

| SSL Certificate ID | HTTPS Status | Domain Name | Operation |
|---|---|---|---|
| -- | Disabled | [redacted]om | Deploy |
| -- | Disabled | [redacted]om | Deploy |
| scs172[redacted] | Enabled | [redacted] | Redeploy |
| scs172[redacted] | Enabled | [redacted] | Redeploy |

**Step 6** Select the domain name you want to deploy the certificate for and click **Deploy** or **Redeploy** in the **Operation** column.

To deply the certificates for multiple domain names, select all the target domain names and click **Batch Update** above the domain name list.

**Step 7** In the displayed dialog box, confirm the information, and click **Confirm**.

After the deployment is successful, a message is displayed, indicating that the deployment is successful. Go to the deployment record page to view the result.

**----End**

### Replacing a Certificate Before It Expires

An SSL certificate issued by any CA around the world is valid for one year. You need to update an SSL certificate in a timely manner. Once your new certificate is issued, replace the old one with it by referring to **Deploying an SSL Certificate to CDN**.

## 2.3.3 Deploying an SSL Certificate to WAF in One Click

### Prerequisites

- You have an SSL certificate that is in **Issued** or **Hosted** status in CCM.
- You have enabled Web Application Firewall (WAF).

### Adding a Domain Name to WAF

Before deploying an SSL certificate to WAF, you need to add the domain name that will use the SSL certificate to WAF. For details, see:

- **Adding a Domain Name to WAF (Cloud Mode)**
- **Adding a Domain Name to WAF (Dedicated Mode)**

**NOTICE**

When a domain name is added to WAF, **HTTPS** must be select for **Client Protocol**.

## (Optional) Upload Private Key

If you select **Upload a CSR** for **CSR** when applying for a certificate, you need to upload the private key to the cloud to deploy the issued certificate to other cloud services in one-click mode because the cloud does not have the private key of the certificate. If you select **System generated CSR** for **CSR** during certificate application, skip this step.

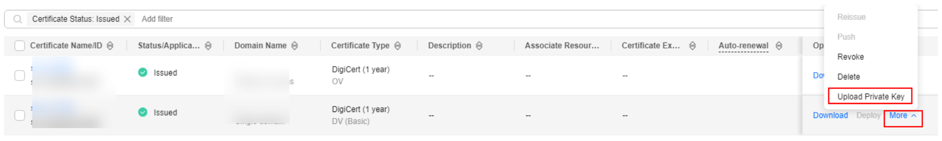
**Step 1** Log in to the management console.

**Step 2** Click ☰ in the upper left corner of the page and choose **Security & Compliance** > **Cloud Certificate Management Service**. The service console is displayed.

**Step 3** In the navigation pane on the left, choose **SSL Certificate Manager** > **SSL Certificates**.

**Step 4** In the **Operation** column of the target certificate, choose **More** > **Upload Private Key**.

**Figure 2-11** Uploading the private key

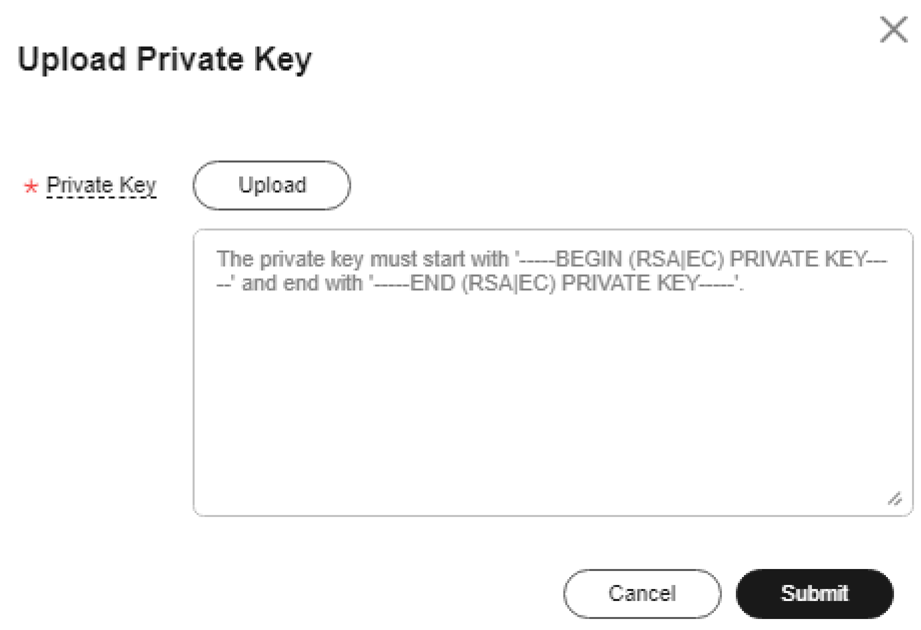


Certificate Status: Issued Add filter

| Certificate Name/ID | Status/Applica... | Domain Name | Certificate Type | Description | Associate Resour... | Certificate Ex... | Auto-renewal | Op |
|---|---|---|---|---|---|---|---|---|
| [redacted] | Issued | [redacted] | DigiCert (1 year) OV | -- | -- | -- | | Do |
| [redacted] | Issued | [redacted] | DigiCert (1 year) DV (Basic) | -- | -- | -- | | Download Deploy More |

**Step 5** In the displayed dialog box, click **Upload** and select a local private key file, or enter the certificate private key information in the text box according to the format.

**Figure 2-12** Uploading the private key file



Upload Private Key

* Private Key

Upload

The private key must start with '-----BEGIN (RSA|EC) PRIVATE KEY-----' and end with '-----END (RSA|EC) PRIVATE KEY-----'.

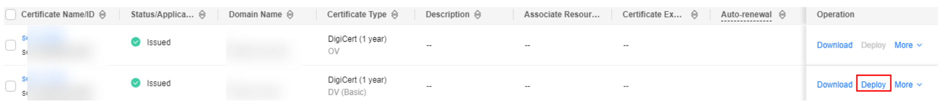
Cancel

Submit

**Step 6** Click **Submit**.

After the private key is uploaded successfully, the **Deploy** button of the certificate becomes available.

**Figure 2-13** Private key uploaded



| Certificate Name/ID | Status/Applica... | Domain Name | Certificate Type | Description | Associate Resour... | Certificate Ex... | Auto-renewal | Operation |
|---|---|---|---|---|---|---|---|---|
| [redacted] | Issued | [redacted] | DigiCert (1 year) OV | -- | -- | -- | | Download Deploy More |
| [redacted] | Issued | [redacted] | DigiCert (1 year) DV (Basic) | -- | -- | -- | | Download Deploy More |

**----End**

## Deploying an SSL Certificate to WAF

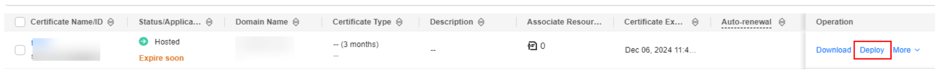
**Step 1** Log in to the **management console**.

**Step 2** Click ☰ in the upper left corner of the page and choose **Security & Compliance** > **Cloud Certificate Management Service**. The service console is displayed.

**Step 3** In the navigation pane on the left, choose **SSL Certificate Manager** > **SSL Certificates**.

**Step 4** Locate the row containing the certificate you want to deploy on other cloud product, and click **Deploy** in the **Operation** to go to the certificate deployment details page.

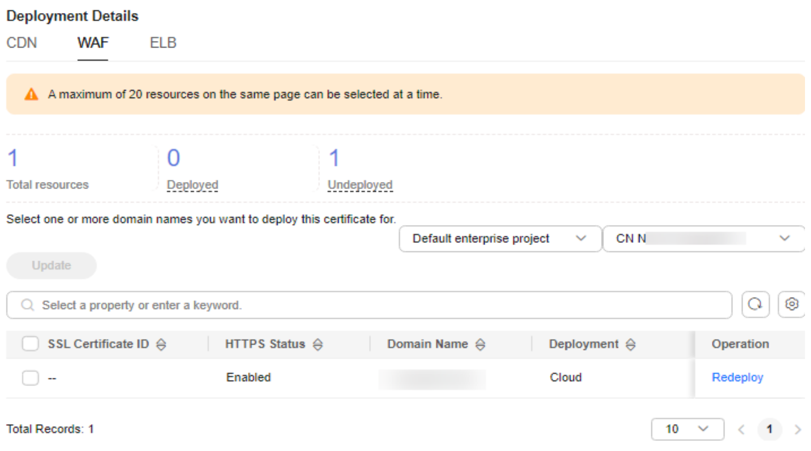
**Figure 2-14** Deploying a certificate



| Certificate Name/ID | Status/Applica... | Domain Name | Certificate Type | Description | Associate Resour... | Certificate Ex... | Auto-renewal | Operation |
|---|---|---|---|---|---|---|---|---|
| [redacted] | Hosted Expire soon | [redacted] | -- (3 months) -- | -- | 0 | Dec 06, 2024 11:4... | | Download Deploy More |

**Step 5** On the displayed page, select **WAF** in the **Deployment Details** area.

**Figure 2-15** Selecting WAF



Deployment Details

CDN WAF ELB

A maximum of 20 resources on the same page can be selected at a time.

1 Total resources | 0 Deployed | 1 Undeployed

Select one or more domain names you want to deploy this certificate for.

Default enterprise project | CN N[redacted]

Update

Select a property or enter a keyword.

| SSL Certificate ID | HTTPS Status | Domain Name | Deployment | Operation |
|---|---|---|---|---|
| -- | Enabled | [redacted] | Cloud | Redeploy |

Total Records: 1

10 < 1 >

**Step 6** Click ▼ on the right of the enterprise project or region name and select the enterprise project or region to be deployed.

**Step 7** Select the domain name you want to deploy the certificate for and click **Redeploy** in the **Operation** column.

To deply the certificates for multiple domain names, select all the target domain names and click **Batch Update** above the domain name list.

**Step 8** In the displayed dialog box, confirm the information, and click **Confirm**.

When the certificate is deployed, the **Deployment** column for the domain name reads **Deployed**.

**----End**

### Replacing a Certificate Before It Expires

An SSL certificate issued by any CA around the world is valid for one year. You need to update an SSL certificate in a timely manner. Once your new certificate is issued, replace the old one with it by referring to **Deploying an SSL Certificate to WAF**.

## 2.3.4 Deploying an SSL Certificate to ELB in One Click

### Prerequisites

- You have an SSL certificate that is in **Issued** or **Hosted** status in CCM.
- You have enabled Elastic Load Balance (ELB).

### Notes and Constraints

- You can use SCM to update the certificate deployed on listeners in ELB. If you update an SSL certificate in SCM, the certificate content and private keys are updated in ELB accordingly. ELB then updates the certificate content and private keys on all listeners where the certificate is deployed for.
- If an ELB certificate is used for multiple domain names, ensure that the new certificate you want to update in SCM for ELB must match with those domain

names. If they do not match, the domain names in the new certificate will overwrite the ones in the original certificate after the update.

For example, the primary domain name and additional domain name of the new certificate are example01.com and example02.com, respectively, and the domain names associated with the original certificate in ELB are example01.com and example03.com. When you update the certificate in SCM, the domain names associated with the certificate in ELB are updated to example01.com and example02.com.

## Creating a Listener and a Load Balancer

Before you start, you need to create a load balancer and listener in ELB. For details, see:

- Creating a Load Balancer
  - **Creating a Shared Load Balancer**.
  - **Creating a Dedicated Load Balancer**
- **Adding an HTTPS Listener**

## Configuring an SSL Certificate in ELB

If you deploy an SSL certificate on ELB for the first time, you need to configure the certificate on ELB so that you can deploy the SSL certificate to ELB using SCM. For details about creating a certificate in ELB, see **Creating a Certificate**.

NOTICE

When creating a certificate, ensure that the domain name you enter must be the same as that included in the SSL certificate.

## (Optional) Upload Private Key

If you select **Upload a CSR** for **CSR** when applying for a certificate, you need to upload the private key to the cloud to deploy the issued certificate to other cloud services in one-click mode because the cloud does not have the private key of the certificate. If you select **System generated CSR** for **CSR** during certificate application, skip this step.

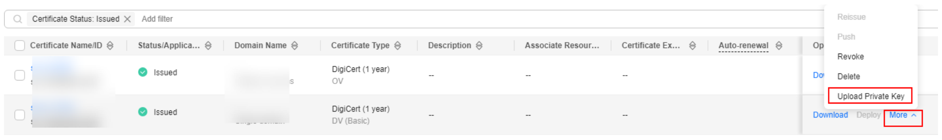
**Step 1** Log in to the **management console**.

**Step 2** Click ☰ in the upper left corner of the page and choose **Security & Compliance** > **Cloud Certificate Management Service**. The service console is displayed.

**Step 3** In the navigation pane on the left, choose **SSL Certificate Manager** > **SSL Certificates**.

**Step 4** In the **Operation** column of the target certificate, choose **More** > **Upload Private Key**.

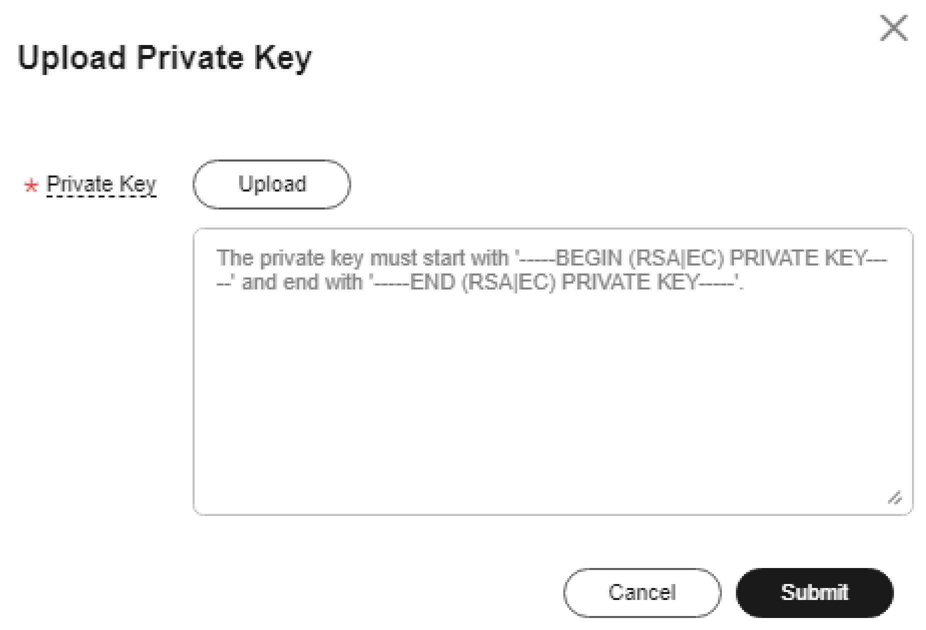
**Figure 2-16** Uploading the private key



| Certificate Name/ID | Status/Applica... | Domain Name | Certificate Type | Description | Associate Resour... | Certificate Ex... | Auto-renewal | Op |
|---|---|---|---|---|---|---|---|---|
| [illegible] | Issued | [illegible] | DigiCert (1 year) OV | -- | -- | -- | | Do |
| [illegible] | Issued | [illegible] | DigiCert (1 year) DV (Basic) | -- | -- | -- | | Download Deploy More |

**Step 5** In the displayed dialog box, click **Upload** and select a local private key file, or enter the certificate private key information in the text box according to the format.

**Figure 2-17** Uploading the private key file



Upload Private Key

* Private Key Upload

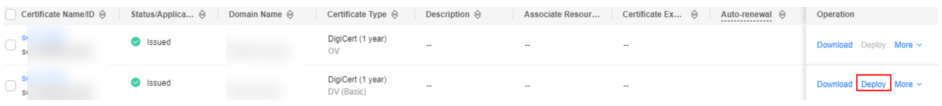
The private key must start with '-----BEGIN (RSA|EC) PRIVATE KEY-----' and end with '-----END (RSA|EC) PRIVATE KEY-----'.

Cancel Submit

**Step 6** Click **Submit**.

After the private key is uploaded successfully, the **Deploy** button of the certificate becomes available.

**Figure 2-18** Private key uploaded



| Certificate Name/ID | Status/Applica... | Domain Name | Certificate Type | Description | Associate Resour... | Certificate Ex... | Auto-renewal | Operation |
|---|---|---|---|---|---|---|---|---|
| [illegible] | Issued | [illegible] | DigiCert (1 year) OV | -- | -- | -- | | Download Deploy More |
| [illegible] | Issued | [illegible] | DigiCert (1 year) DV (Basic) | -- | -- | -- | | Download Deploy More |

**----End**

## Deploying an SSL Certificate to ELB

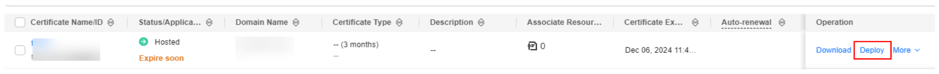
**Step 1** Log in to the management console.

**Step 2** Click ☰ in the upper left corner of the page and choose **Security & Compliance** > **Cloud Certificate Management Service**. The service console is displayed.

**Step 3** In the navigation pane on the left, choose **SSL Certificate Manager** > **SSL Certificates**.

**Step 4** Locate the row containing the certificate you want to deploy on other cloud product, and click **Deploy** in the **Operation** to go to the certificate deployment details page.

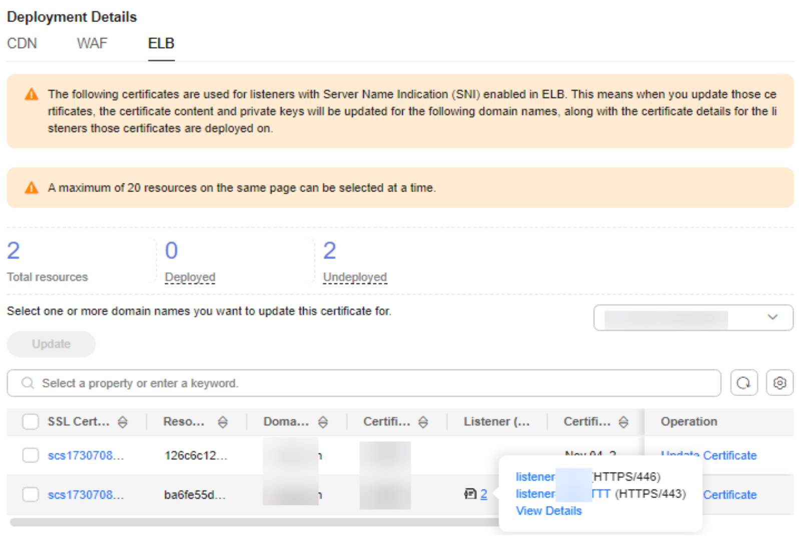
**Figure 2-19** Deploying a certificate



| Certificate Name/ID | Status/Applica... | Domain Name | Certificate Type | Description | Associate Resour... | Certificate Ex... | Auto-renewal | Operation |
|---|---|---|---|---|---|---|---|---|
| | Hosted Expire soon | | -- (3 months) -- | -- | 0 | Dec 06, 2024 11:4... | | Download Deploy More |

**Step 5** On the displayed page, select **ELB** in the **Deployment Details** area.

**Figure 2-20** Selecting ELB



Deployment Details

CDN WAF ELB

The following certificates are used for listeners with Server Name Indication (SNI) enabled in ELB. This means when you update those certificates, the certificate content and private keys will be updated for the following domain names, along with the certificate details for the listeners those certificates are deployed on.

A maximum of 20 resources on the same page can be selected at a time.

2 Total resources | 0 Deployed | 2 Undeployed

Select one or more domain names you want to update this certificate for.

Update

Select a property or enter a keyword.

| SSL Cert... | Reso... | Doma... | Certifi... | Listener (... | Certifi... | Operation |
|---|---|---|---|---|---|---|
| scs1730708... | 126c6c12... | | | | Nov 04, 2... | Update Certificate |
| scs1730708... | ba6fe55d... | | | 2 | | Update Certificate |

listener (HTTPS/446)
listener (HTTPS/443)
View Details

**Step 6** Click ▼ on the right of the **Region** drop-down list and select the region where you want to deploy the certificate.

**Step 7** Select the domain name you want to update the certificate for and click **Update Certificate** in the **Operation** column.

To update the certificates for multiple domain names, select all the target domain names and click **Batch Update** above the domain name list.

**Step 8** In the displayed dialog box, confirm the information, and click **Confirm**.

If a message indicating that the certificate is updated successfully is displayed, the SSL certificate is updated for ELB.

**----End**

## Replacing a Certificate Before It Expires

An SSL certificate issued by any CA around the world is valid for one year. You need to update an SSL certificate in a timely manner. Once your new certificate is issued, replace the old one with it by referring to **Deploying an SSL Certificate to ELB**.

## 2.4 Using FunctionGraph to Automatically Obtain and Update ECS Server Certificates

### Scenario

This topic illustrates how to use FunctionGraph to automatically obtain and update an ECS server certificate. For server information, see **Table 2-3**. This approach helps you update new certificates issued upon renewals for your ECSs. The entire process requires no manual operations.

**Table 2-3** Example

| | |
|---|---|
| **Server Type** | Nginx |
| **Programming Language** | Python 3.9 |

### Notes and Constraints

- You have enabled an Elastic Cloud Server (ECS) and installed an SSL certificate on the ECS.
- Your SSL certificate has been purchased and renewed through CCM.

### Step 1: Creating an Agency

To use FunctionGraph to update the ECS server certificate, you need to grant the SCM FullAccess and IAM ReadOnlyAccess permissions to FunctionGraph.

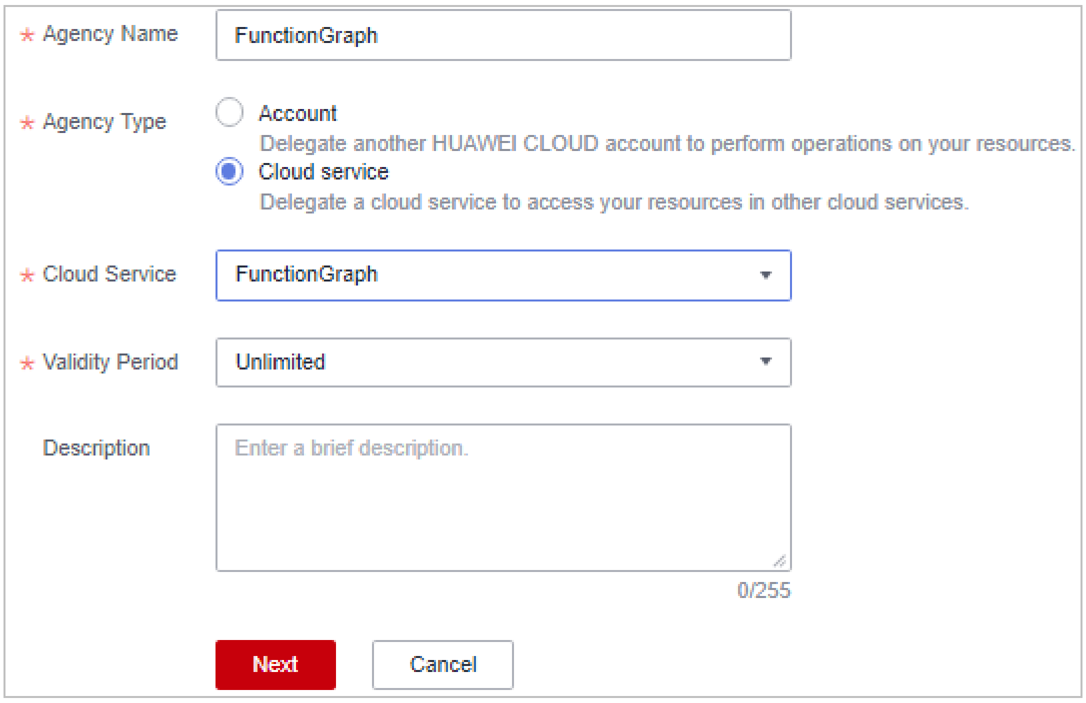
**Step 1** **Log in to the management console.**

**Step 2** Click ☰ in the upper left corner of the page and choose **Management & Governance** > **Identity and Access Management**.

**Step 3** In the navigation pane on the left, choose **Agencies**. In the upper right corner of the **Agencies** page, click **Create Agency**.

**Step 4** On the **Create Agency** page, set the agency information by referring to **Table 2-4**.

**Figure 2-21** Create Agency



| | |
|---|---|
| * Agency Name | FunctionGraph |
| * Agency Type | Account<br>Delegate another HUAWEI CLOUD account to perform operations on your resources.<br>[✓ Cloud service]<br>Delegate a cloud service to access your resources in other cloud services. |
| * Cloud Service | FunctionGraph |
| * Validity Period | Unlimited |
| Description | Enter a brief description.<br>0/255 |

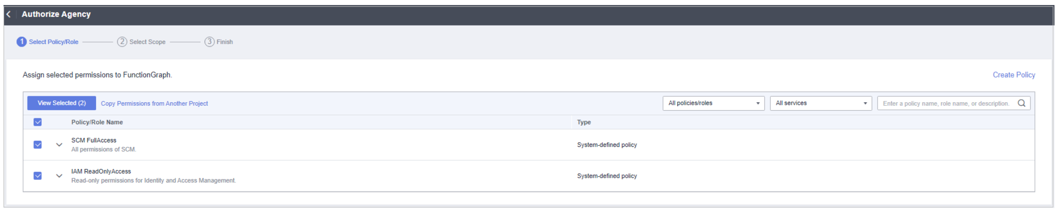
**Table 2-4** Parameters for creating an agency

| Parameter | Description |
|---|---|
| Agency Name | User-defined agency name. |
| Agency Type | Select **Cloud service**. |
| Cloud Service | Choose **FunctionGraph**. |
| Validity Period | Select **Unlimited**. |
| Description | This parameter is optional. You can customize the required information. |

**Step 5** Click **Next**.

**Step 6** Select the **SCM FullAccess** and **IAM ReadOnlyAccess** permissions you want to grant to FunctionGraph.

**Figure 2-22** Selecting permissions



Assign selected permissions to FunctionGraph.

| ✓ | Policy/Role Name | Type |
|---|---|---|
| ✓ | SCM FullAccess<br>All permissions of SCM. | System-defined policy |
| ✓ | IAM ReadOnlyAccess<br>Read-only permissions for Identity and Access Management. | System-defined policy |

**Step 7** Click **Next** to set the permission scope.

**Step 8** Click **OK**.

**----End**

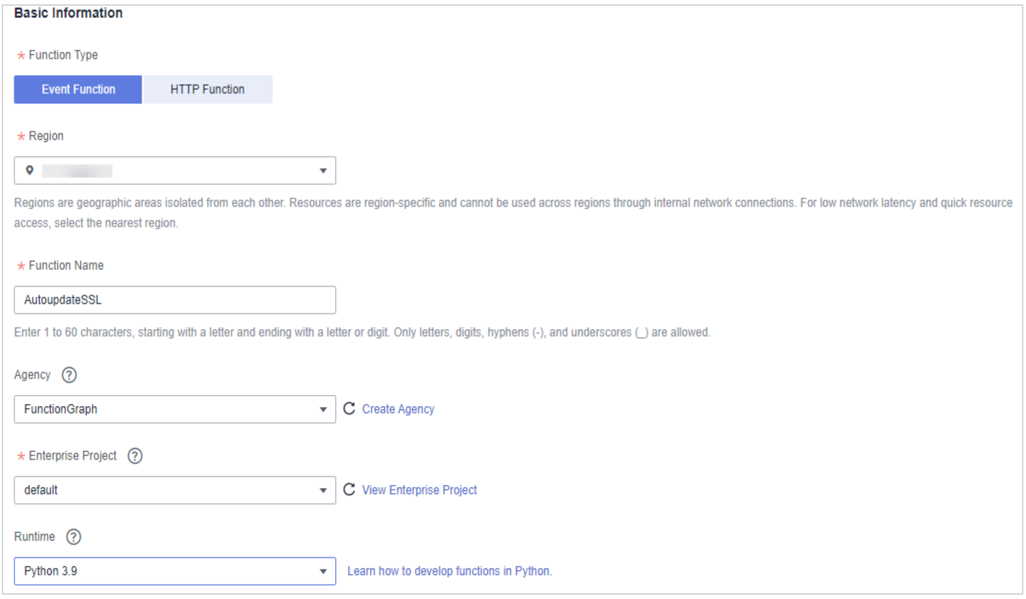
## Step 2. Create a Function from Scratch

**Step 1** **Log in to the management console.**

**Step 2** Click ☰ in the upper left corner of the page and choose **Compute** > **FunctionGraph**.

**Step 3** Click **Create** in the upper right corner of the **Functions** area.

**Step 4** Create a function by referring to **Table 2-5**.



**Basic Information**

* Function Type

Event Function | HTTP Function

* Region

Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region.

* Function Name

AutoupdateSSL

Enter 1 to 60 characters, starting with a letter and ending with a letter or digit. Only letters, digits, hyphens (-), and underscores (_) are allowed.

Agency

FunctionGraph

Create Agency

* Enterprise Project

default

View Enterprise Project

Runtime

Python 3.9

Learn how to develop functions in Python.

**Figure 2-23** Creating a function from scratch

**Table 2-5** Parameters for creating a function from scratch

| Parameter | Description |
|---|---|
| Function Type | Select **Event Function**. |
| Region | Select a region where you will deploy your code. |
| Function Name | Name of the custom function. |
| Agency | Select the agency you created in **Step 1: Creating an Agency**. |

| Parameter | Description |
|---|---|
| Enterprise Project | If you have enabled the enterprise project function, select the enterprise project to which you want to add a function.<br>If you have not enabled the enterprise project function yet, the parameter will not be displayed. You can enable this function by referring to **Enabling the Enterprise Project Function** if necessary. |
| Runtime | Select a language for the function. In this example, **Python 3.9** is selected. |

**Step 5** Click **Create**. The **Functions** page is displayed, and an empty function is created.

**----End**

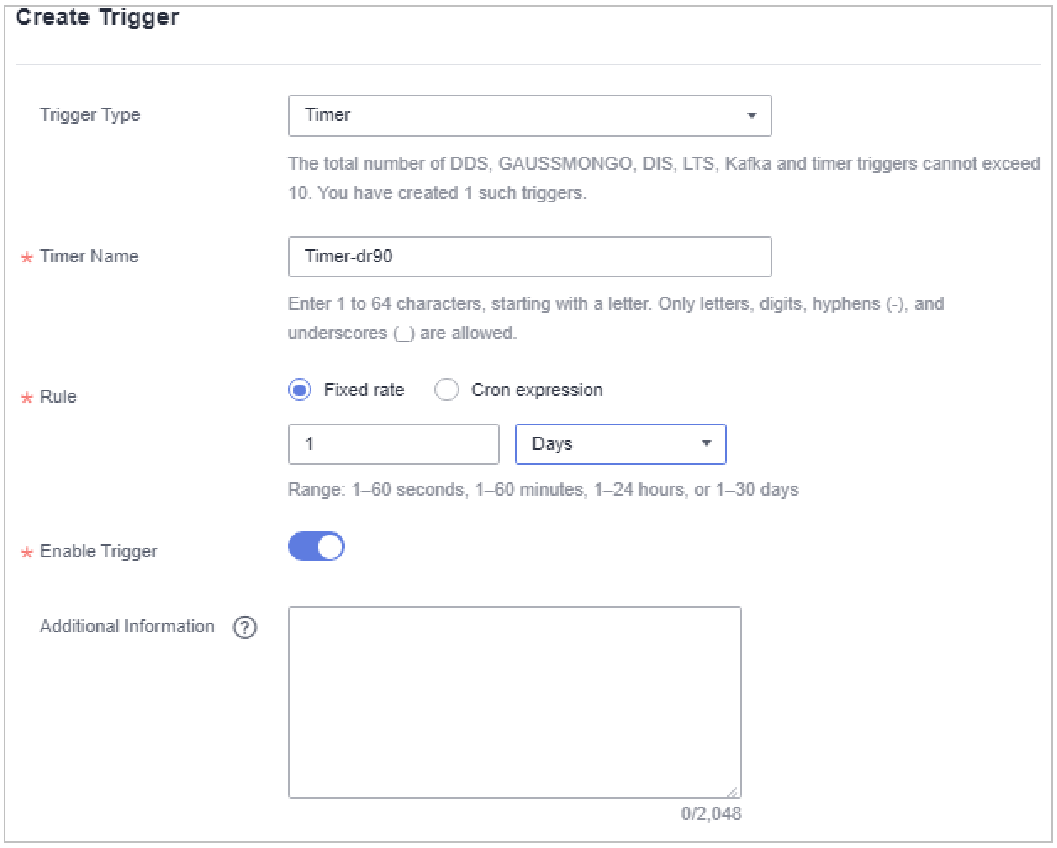
## Step 3: Create a Timer Trigger

Create a timer trigger to trigger a function at a fixed interval.

**Step 1** On the displayed page, choose **Configuration** > **Triggers**.

**Step 2** Click **Create Trigger** and set the parameters by referring to **Table 2-6**.

**Figure 2-24** Create Trigger


Create Trigger

Trigger Type: Timer

The total number of DDS, GAUSSMONGO, DIS, LTS, Kafka and timer triggers cannot exceed 10. You have created 1 such triggers.

* Timer Name: Timer-dr90

Enter 1 to 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.

* Rule: [✓ Fixed rate] Cron expression

1 Days

Range: 1–60 seconds, 1–60 minutes, 1–24 hours, or 1–30 days

* Enable Trigger

Additional Information

0/2,048

**Table 2-6** Parameters for setting a timer trigger

| Parameter | Description |
|---|---|
| Trigger Type | Select **Timer**. |
| Timer Name | Customize a name. |
| Rule | Select **Fixed rate** and set a value to it based on your needs. |
| Enable Trigger | ● |
| Additional Information | This parameter is optional. You can enter anything for the trigger. |

**Step 3** Click **OK**.

**----End**

## Step 4: Create and Configure a Function Dependency

The function code for deploying a certificate to an ECS depends on **paramiko**. You need to create and configure the paramiko dependency for the function.

This section uses Python 3.9 as an example to describe how to create and configure a dependency for a function. For details about how to create a dependency package in other languages, see **How Do I Create Dependencies?**

Creating a Dependency for a Python Function

**Step 1** The Python version in the environment must be the same as the runtime version of the corresponding function.

For example, Python 3.9.0 or later is recommended for Python 3.9, Python 2.7.12 or later is recommended for Python 2.7, and Python 3.6.3 or later is recommended for Python 3.6.

**Step 2** Install the paramiko dependency package for Python 3.9 and specify the local installation path of the dependency package to **/tmp/paramiko**:

```
pip install paramiko --root /tmp/paramiko
```

**Step 3** Switch to the **/tmp/paramiko** directory:

```
cd /tmp/paramiko/
```

**Step 4** Go to the **site-packages** directory (generally, **usr/lib64/python3.9/site-packages/**) and then run the following command:

```
zip -rq paramiko.zip *
```

The generated package is the required dependency package.

NOTE

To install the local wheel installation package, run the following command:

```
pip install piexif-1.1.0b0-py2.py3-none-any.whl --root /tmp/piexif
//Replace piexif-1.1.0b0-py2.py3-none-any.whl with the actual installation package name.
```

Configuring the Dependency Package

**Step 5** **Log in to the management console.**

**Step 6** Click ☰ in the upper left corner of the page and choose **Compute** > **FunctionGraph**.

**Step 7** In the navigation pane on the left, choose **Functions** > **Dependencies**.

**Step 8** Click **Create Dependency**. In the **Create Dependency** pane sliding out from the left, set details by referring to **Configuring a dependency package**.

**Table 2-7** Configuring a dependency package

| Parameter | Description |
|---|---|
| Name | Dependency name, which is used to identify different packages. |
| Code Entry Mode | Select **Upload ZIP**. |
| Upload File | Select the dependency .zip file. |
| Runtime | Select the function language. In this example, **Python 3.9** is selected. |

| Parameter | Description |
|---|---|
| Description | Description of the dependency. This parameter is optional. |

**Step 9** Click **OK**.

**Step 10** In the navigation pane on the left, choose **Functions** > **Function List**.

**Step 11** Click the name of the desired function.

**Step 12** On the displayed function details page, click the **Code** tab, click **Add** in the **Dependencies** area.

**Step 13** Select the private dependency package created in 8 and click OK.

**----End**

## Step 5: Configure a Code Source in a Function

The following shows how to configure function code online. For more details, see **Creating a Deployment Package**.

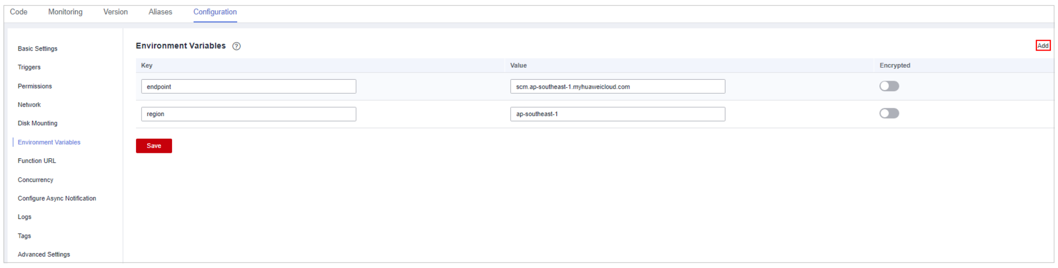
**Step 1** In the navigation pane on the left, choose **Functions** > **Function List**.

**Step 2** Click the name of the desired function.

**Step 3** Choose **Configuration** > **Environment Variables**.

**Step 4** Click **Add** and configure variables **endpoint** and **region**, as shown in **Figure 2-25**.

**Figure 2-25** Setting Environment Variables



Environment Variables

| Key | Value | Encrypted |
|---|---|---|
| endpoint | scm.ap-southeast-1.myhuaweicloud.com | |
| region | ap-southeast-1 | |

Environment variable 1:

- **Key**: **endpoint**
- **Value**: **scm.ap-southeast-1.myhuaweicloud.com**

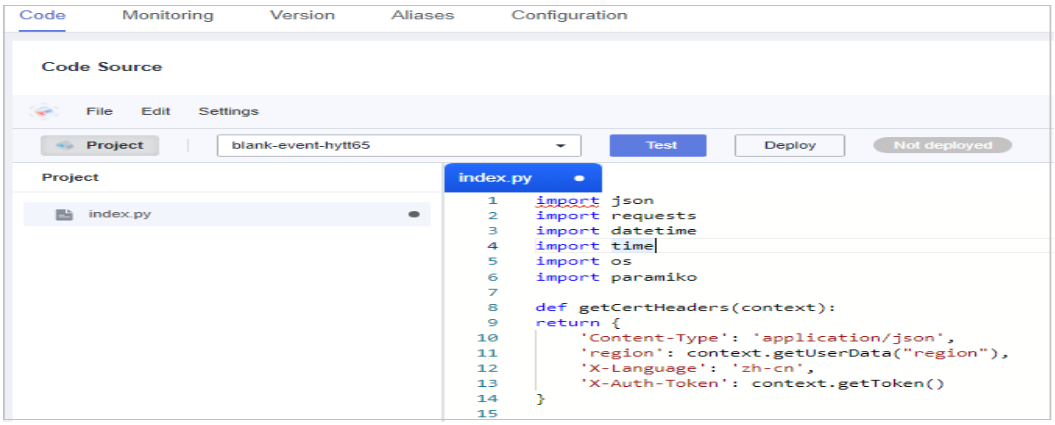
Environment variable 2:

- **Key**: **region**
- **Value**: ap-southeast-1

**Step 5** Click **Save** and click the **Code** tab.

**Step 6** On the **Code** tab page, integrate the following two pieces of code and add them to a code source file.

**Figure 2-26** Adding code



```
import json
import requests
import datetime
import time
import os
import paramiko

def getCertHeaders(context):
return {
    'Content-Type': 'application/json',
    'region': context.getUserData("region"),
    'X-Language': 'zh-cn',
    'X-Auth-Token': context.getToken()
}
```

The following is an example of the application code for obtaining the SSL certificate of the current account:

```
import json
import requests
import datetime
import time

def getCertHeaders(context):
    return {
        'Content-Type': 'application/json',
        'region': context.getUserData("region"),
        'X-Language': 'en-us',
        'X-Auth-Token': context.getToken()
    }

def isValidCert(cert):
# TODO can be customized based on service scenarios. The following is only an example.
    certDomain = cert.get('domain')

# Check whether the certificate is a renewal certificate used for the corresponding domain name.
    if (certDomain != 'XXXX'):
        return False

# The following instance sorts the certificates that were issued yesterday and whose domain names meet
the requirements.
    currentTime = time.localtime()
    currentTimeStr = str(currentTime[0]) + ',' + str(currentTime[1]) + ','  + str(currentTime[2])

    certTime = datetime.datetime.strptime(cert.get('expire_time'), '%Y-%m-%d %H:%M:%S.%f')

# Obtain the certificate issuing time.
    certTimeStr = str(certTime.year - int(cert.get('validity_period')/12)) + ',' + str(certTime.month) + ','  +
str(certTime.day - 1)
    return currentTimeStr == certTimeStr

def getCertList(context):
    preUrl = 'https://' + context.getUserData("endpoint")
    url = preUrl + '/v3/scm/certificates?
order_status=ISSUED&content=&sort_key=certUpdateTime&sort_dir=DESC&limit=&enterprise_project_id='
    certHeaders = getCertHeaders(context)

    rep = requests.get(url, headers = certHeaders)
    totalCount = json.loads(rep.text).get('total_count')
    discuss = int(totalCount/10)
    reminder = totalCount-discuss*10
    rep = []
    for i in range(discuss):
        tempUrl = url + '&offset=' + str(10*i)
        tempRep = requests.get(tempUrl, headers = certHeaders)
        for cert in json.loads(tempRep.text).get('certificates'):
            if(isValidCert(cert)):
                rep.append(cert)
    if reminder > 0:
        tempUrl = url + '&offset=' + str(totalCount-reminder)
        tempRep = requests.get(tempUrl, headers = certHeaders)
        for cert in json.loads(tempRep.text).get('certificates'):
            if(isValidCert(cert)):
                rep.append(cert)
    return json.dumps(rep)


def exportCert(context, certId):
    preUrl = 'https://' + context.getUserData("endpoint")
    url = '/v3/scm/certificates/'+ certId + '/export'
    rep = requests.post(preUrl + url, headers = getCertHeaders(context))
    os.makedirs("/tmp/" + certId)
    entireCertificate = json.loads(rep.text).get('entire_certificate')
    entireCertFileName = '/tmp/' + certId + '/certificate.pem'
    certFile = open(entireCertFileName,'w')
    certFile.write(entireCertificate)
    privateKey = json.loads(rep.text).get('private_key')
    privateKeyFileName = '/tmp/' + certId + '/privateKey.key'
    keyFile = open(privateKeyFileName,'w')
    keyFile.write(privateKey)


def handler (event, context):
# TODO needs to be invoked by the function based on the service background. The following example is for
reference only.
    totalRep = getCertList(context)
    certList = json.loads(totalRep)
    certIdList = []
    for cert in certList:
        exportCert(context, cert.get("id"))
        certIdList.append(cert.get("id"))
    for cert in certList:
        deploy('**.***.*', 22, 'root', '*.', '/tmp', '/tmp', certIdList)
```

The following is an example of the application code for deploying an SSL certificate to an ECS (Nginx):

```
import os
import paramiko
import time

def isExists(path, function):
    path = path.replace("\\","/")
    try:
        function(path)
    except Exception as error:
        return False
    else:
        return True


def copy(ssh, sftp, local, remote):
    if isExists(remote, function=sftp.chdir):
        filename = os.path.basename(os.path.normpath(local))
        remote = os.path.join(remote, filename).replace("\\","/")
    if os.path.isdir(local):
        isExists(remote, function=sftp.mkdir)
        for file in os.listdir(local):
            localfile = os.path.join(local, file).replace("\\","/")
            copy(ssh=ssh, sftp=sftp, local=localfile, remote=remote)
    if os.path.isfile(local):
        try:
            ssh.exec_command("rm -rf %s"%(remote))
            sftp.put(local,remote)
        except Exception as error:
            print('put:', local, "==>",remote, 'FAILED')
        else:
            print('put:', local, "==>",remote, 'success')

def deploy(ip, port, username, password, local, remote, certIdList):
    transport = paramiko.Transport((ip,port))
    transport.connect(username=username, password=password)
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh._transport = transport
    ftp_client = paramiko.SFTPClient.from_transport(transport)

    for certId in certIdList:
        copy(ssh=ssh, sftp=ftp_client, local=local + '/' + certId, remote=remote)

#The prerequisite is that the certificate location has been written into the nginx.conf file.
    cmd="/usr/local/nginx/sbin/nginx -s reload"
    stdin,stdout,stderr = ssh.exec_command(cmd)
    ssh.close()
```

**NOTE**

- The preceding two pieces of code are examples. Modify them based on site requirements. *XXXX* in the code indicates the domain name. Replace it with the actual domain name.
- The following functions involved in the example code need to be edited based on your service background. They must be placed at the end of the code source file:

```
def handler (event, context):
# TODO needs to be invoked by the function based on the service background.
```

**Step 7** Click **Test** to test the function.

For details, see **Test Management**.

**Step 8** After the code source is added and tested, the function runs based on the triggering rule set by the timer trigger. If a renewal certificate is issued, the function automatically obtains the certificate and updates it to the ECS.

**Step 9** On the function details page, choose **Monitoring** > **Metrics** to view the function running status.

You can view metrics such as the **Invocations**, **Duration**, **Errors**, and **Throttles**. For details, see **Function Monitoring**.

**----End**

# 3 Best Practices for Private Certificate Management

## 3.1 Best Practices for Private Certificate Management

### 3.1.1 Managing the Private Certificate Lifecycle

**Table 3-1** describes operations during the private certificate lifecycle management.

**Table 3-1** Private certificate lifecycle management description

| Operation | Description | Remarks |
|---|---|---|
| Applying for a private certificate | Private certificates are classified into client certificates and server certificates based on the role of an entity in communications. Before applying for a private certificate, ensure that you have created a private CA that can be used to issue certificates. | • Private certificates are billed by how many certificates you apply for. Once a private certificate is issued, it cannot be refunded.<br>• The common name of a private certificate can be duplicate. To distinguish certificates, you can specify a unique name for your private certificates. |

| Operation | Description | Remarks |
|---|---|---|
| Exporting a private certificate | Export a private certificate (including the private key) that has been issued. You can select the certificate format. | Keep the private keys of private certificates secure. If the private key is disclosed, revoke and replace the private certificate in time.<br>**NOTICE**<br>If any CA certificate in the certificate chain path is permanently deleted, the private certificate cannot be exported. |
| Revoking a private certificate | You can revoke any private certificate you no longer need for any reason. Revoking private certificates in a timely manner prevents abuse of private certificates. | Certificate abuse may cause security problems.<br>**NOTICE**<br>If the parent CA does not enable the CRL configuration, the private certificate revocation status cannot be queried. This means a revoked private certificate can still pass the validation. |
| Deleting a private certificate | You can delete a private certificate anytime. | You can delete a private certificate in any state.<br>**NOTICE**<br>This operation will immediately delete all information about the private certificate from the database. This operation is irreversible. Exercise caution when performing this operation. |

## 3.1.2 Private Certificate Statuses

**Table 3-2** describes the private certificate statuses.

**Table 3-2** Private certificate status description

| Private Certificate Status | Certificate Can Be Exported | Certificate Quota Used |
|---|---|---|
| Issued | Yes | Yes |
| Revoked | No | Yes |
| Expired | No | Yes |

### 3.1.3 Rotating Your Private Certificate

Private certificates (including private keys) are deployed on service nodes and are frequently used for encrypted communications. To prevent private key leakage, the validity period and rotation period of private certificates should be configured based on your service security requirements. Private certificate rotation means using a new private certificate to replace the old one. For example, if a private certificate is used for an encrypted meeting that is highly confidential, the validity period of the private certificate is usually at the hour level. If a private certificate is deployed on a web server, the validity period is usually at the year level. Currently, the validity period of SSL certificates issued by an international certificate authority is basically one year.

The rotation period of a private certificate is set based on its expiration date. The basic principle for certificate rotation is to replace the old private certificate with the new one on the corresponding working node before the old one expires, preventing communications interruption caused by the expiration of the private certificate.

CAUTION

- To prevent service interruption caused by the expiration of the old private certificate, enough time should be reserved to ensure that a private certificate can be successfully rotated as a longer period may be required to re-rotate or manually rotate an old private certificate once the first rotation fails due to uncontrollable factors.
- If the replaced old private certificate still has a long validity period, revoke it to prevent abuse.
- If the old and new private certificates have different root CAs, add the new root CA to the root CA trust list.

## 3.2 Best Practices for Private CA Management

### 3.2.1 Designing a Private CA Hierarchy

With PCA, you can create a hierarchy of certificate authorities with up to seven levels. The root CA can have any number of branches and have as many as six levels of subordinate CAs (or child CAs or intermediate CAs) on each branch. A well-designed CA hierarchy can:

- Keep the entire public key infrastructure (PKI) system more reasonable and secure.
- Make fine-grained control over certificates a reality.
- Make the PKI system more suitable for your own business structure, facilitating workloads migrations and expansion.

**Table 3-3** describes each structure that you can create in the PCA service. You can design the CA hierarchy to meet your business needs.

**Table 3-3** CA hierarchies

| CA Hierarchy | Description | Remarks |
|---|---|---|
| Single-level CA hierarchy | The root CA directly issues private certificates. | This structure does not comply with security specifications and is often used in non-production environments, such as environments for development and testing when a full chain of trust is not required.<br>The root CA is frequently used, and the risk of key leakage is high. Once the key of the root CA is leaked, all certificates issued by the root CA must be discarded, and all terminals must quickly remove the leaked root CA from the trusted root certificate list, which is time- and labor-consuming. The worst of this is that services are severely interrupted. **Figure 3-1** shows a single-level CA hierarchy. |
| Two-level CA hierarchy | The root CA issues level-2 subordinate CAs, and the subordinate CAs (with the path length set to 0) issue private certificates. | This structure is a common CA hierarchy.<br>The CA hierarchy has only two levels. The certificate chain uses less resources during transmission and certificate validation compared with other complex hierarchies. The root CA is isolated from the private certificates, and subordinate CAs are used to issue certificates. If a subordinate CA is compromised, you only need to revoke and replace all certificates below the subordinate CA. There is no impact on other subordinate CAs. The terminal does not need to remove the root CA. This thereby narrow the impact scope of leakage events. **Figure 3-2** shows a two-level CA hierarchy. |
| Three-level CA hierarchy | The root CA issues level-2 subordinate CAs, and the level-2 subordinate CAs (with the path length set to 1) issue level-3 subordinate CAs. The level-3 subordinate CAs (with the path length set to 0) issue private certificates. | This structure is also a common CA hierarchy. It is suitable for organizations with complex structures.<br>The three-level structure enables fine-grained control over certificate distribution and management. The PKI system has more levels to separate the root CA and private certificates, which can better protect the confidentiality of root CA keys. **Figure 3-3** shows a three-level CA hierarchy. |

| CA Hierarchy | Description | Remarks |
|---|---|---|
| CA hierarchies with four to seven levels | The root CA issues level-2 subordinate CAs (path length range: 2 to 5). Level-2 CAs issue level-3 subordinate CAs (path length range: 1 to 4), and level-3 CAs issue level-4 subordinate CAs (path length range: 0 to 3). The bottom level subordinate CAs are responsible for issuing private certificates. | This structure is seldom used.<br>Although the CA hierarchy is very fine-grained, it has too many levels between the root CA and the private certificates. This leads to a large certificate chain file, which uses heavy network transmission overhead and takes long time for certificate validation. **Figure 3-4** shows the CA hierarchies with four to seven levels. |

**NOTE**

A path length determines how many levels of subordinate CAs the current CA can issue. It is used to control the certificate chain length.

**Figure 3-1** Single-level CA hierarchy

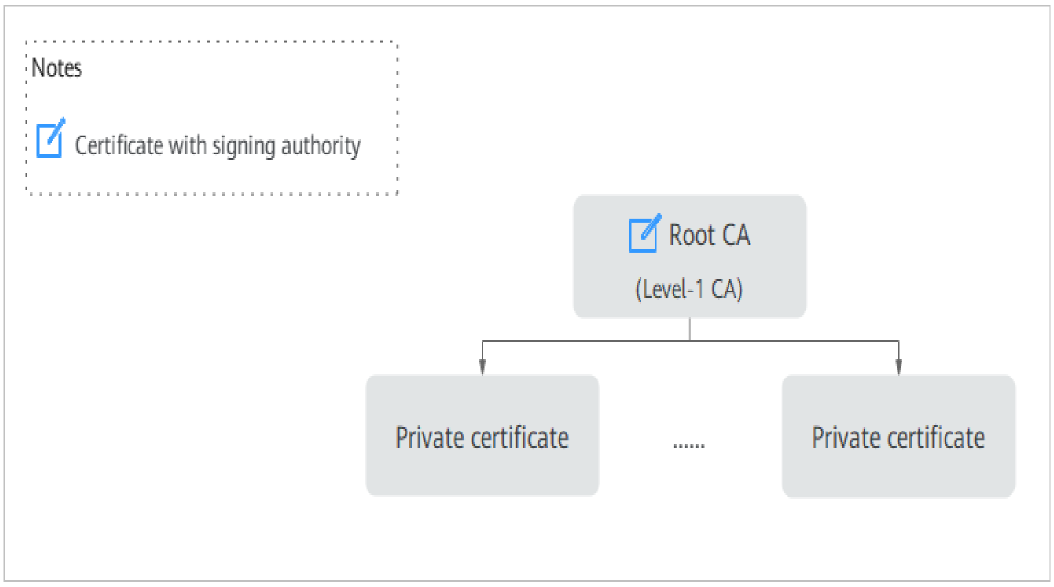


**Figure 3-2** Two-level CA hierarchy

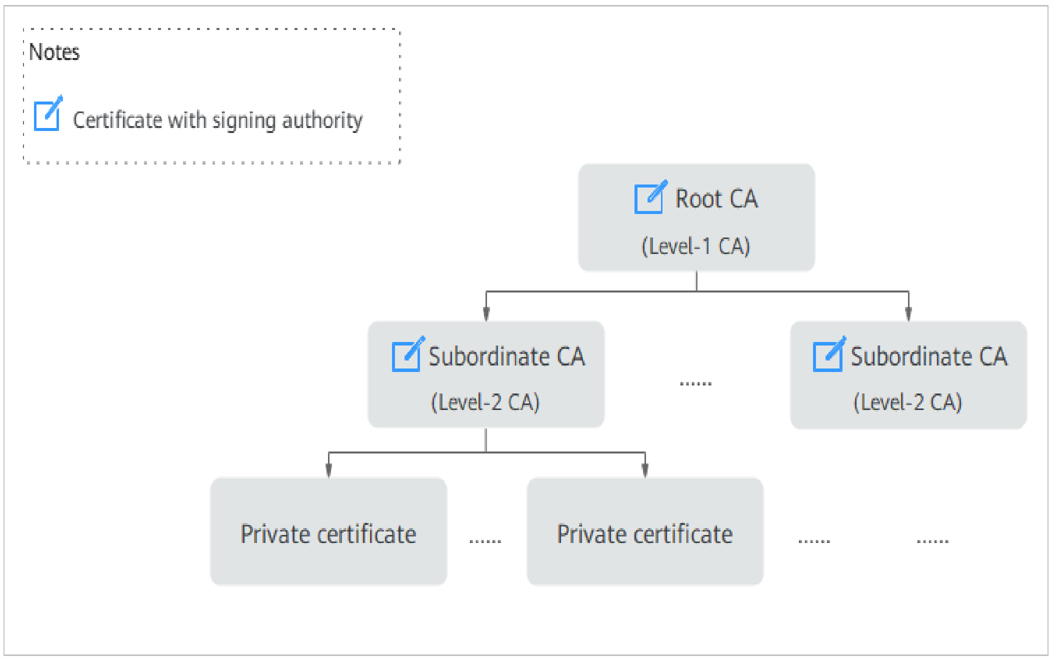


**Figure 3-3** Three-level CA hierarchy

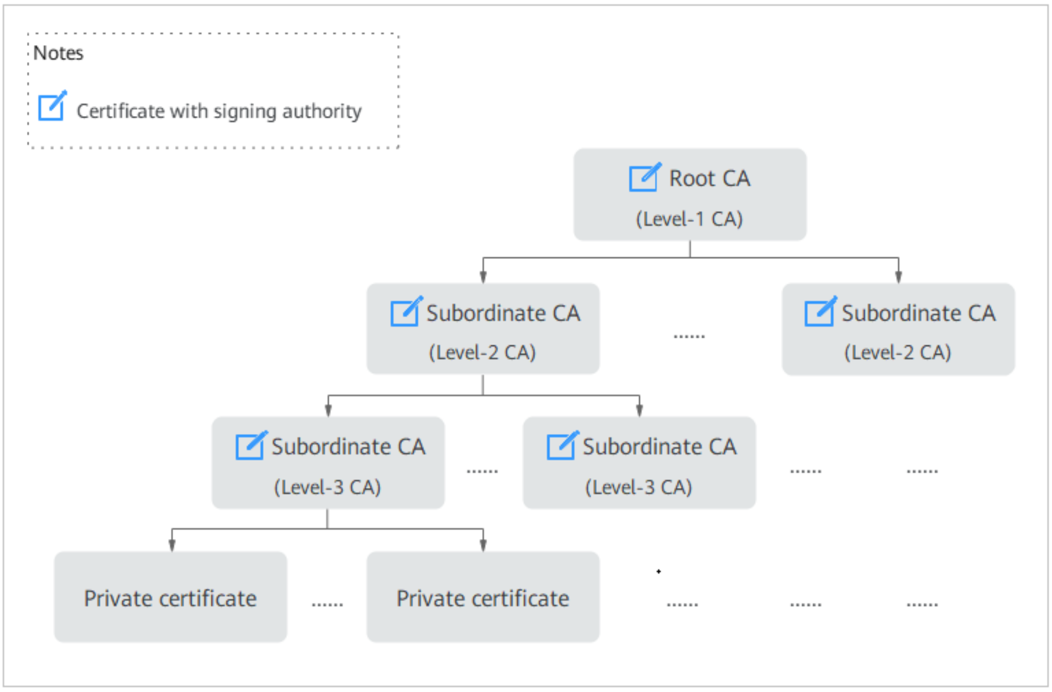
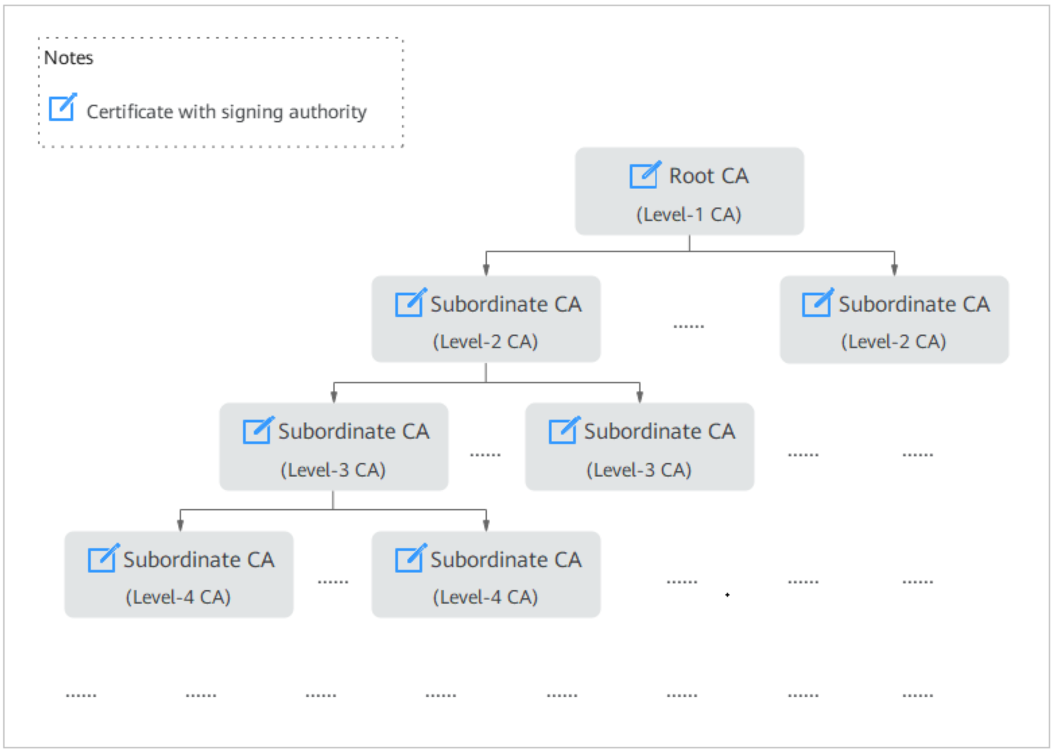


**Figure 3-4** Four- to seven-level CA hierarchies



## 3.2.2 Private CA Statuses

PCA manages the private CA lifecycle by status. **Figure 3-5** shows how statuses are determined for private CAs. A private CA in different statuses has different functions, as described in **Table 3-4**.

**Figure 3-5** CA lifecycle

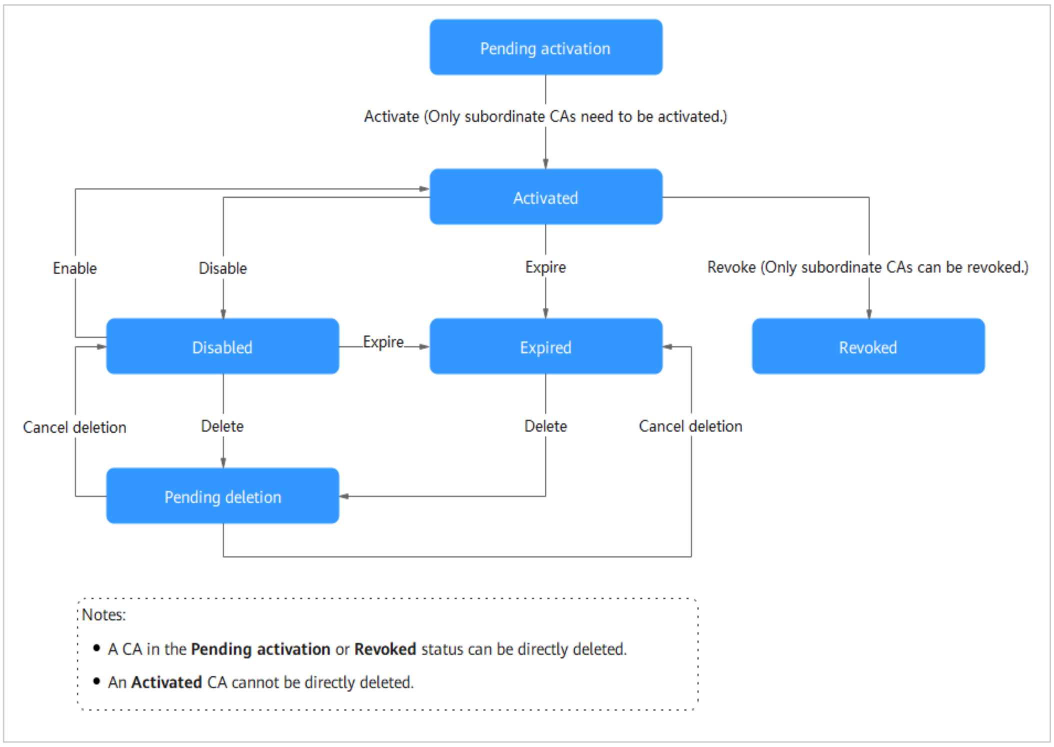

Pending activation
Activate (Only subordinate CAs need to be activated.)
Activated
Enable
Disable
Expire
Revoke (Only subordinate CAs can be revoked.)
Disabled
Expire
Expired
Revoked
Cancel deletion
Delete
Delete
Cancel deletion
Pending deletion
Notes:
A CA in the Pending activation or Revoked status can be directly deleted.
An Activated CA cannot be directly deleted.

**Table 3-4** Functions and statuses of private CAs

| Status | Issuing a Certificate | Revoking a Certificate | Exporting a Certificate | Importing a Certificate | Exporting a CSR | CA Quota Used | Billed |
|---|---|---|---|---|---|---|---|
| Pending activation | No | No | No | Yes | Yes | Yes | No |
| Activated | Yes | Yes | Yes | No | No | Yes | Yes |
| Disabled | No | Yes | Yes | No | No | Yes | Yes |
| Pending deletion | No | No | No | No | No | Yes | No |
| Expired | No | No | Yes | No | No | Yes | Yes |
| Revoked | No | No | No | No | No | Yes | No |

CAUTION

- Only subordinate CAs can be in the **Pending activation** or **Revoked** status. Subordinate CAs in the **Pending activation** status are not billed. A root CA will never be in the **Revoked** status as it is a self-signed certificate.
- If you cancel the deletion of a private CA in the **Pending deletion** status, it will be billed for the actual pending deletion period.
- Expired private CAs still use the quota of private CAs so they will be billed until they are completely deleted. If you no longer use a private CA, delete it in a timely manner to avoid money waste.

## 3.2.3 Managing the Private CA Lifecycle

### Creating a CA

Private CAs are classified into root CAs and subordinate CAs. You can specify the type of the CA you want to create. The root CA is directly created from the digital signature certificate. A subordinate CA subordinates to its parent CA. Before creating a subordinate CA, create its parent CA. With PCA, you can create:

- A root CA. After a root CA is created, it is in the **Activated** status by default. A root CA. The key of a root CA is used only for digital signature, issuing certificates, and signing certificate revocation lists (CRLs), which cannot be customized. It means a root CA can be used only to issue certificates, revoke certificates, and sign CRLs.
- A subordinate CA and activate it. In this manner, after a subordinate CA is created, it is in the **Activated** status. By default, the key usage of a subordinate CA is the same as that of the root CA, but you can customize the key usage of a subordinate CA.
- A subordinate CA but do not activate it. After a subordinate CA is created, it is in the **Pending activation** status. A subordinate CA in this state is not ready for any use until it is activated, and you can delete it directly.

NOTE

The common name of a private CA can be duplicate. Identifiers are recommended for you to distinguish CAs, for example, ROOT CA G0 and ROOT CA G1.

### Activating a CA

A subordinate CA in the **Pending activation** status cannot be used until you activate it. Once a subordinate CA is activated, the billing starts, and there is no way to let it go back to the **Pending activation** status.

### Disabling a CA

After you disable a private CA, it cannot issue certificates, but it can still revoke certificates and sign CRLs. Only activated private CAs can be disabled. After you disable a private CA, its status changes to **Disabled**.

Generally, if a CA is about to expire, it is disabled to ensure that new certificates are issued by a new CA. The old CA can still revoke certificates it issued. Old certificates can still work before they are replaced by the new ones.

## Enabling a CA

You can enable a **Disabled**private CA and use it to issue certificates. After you enable a **Disabled** private CA, its status changes to **Activated**.

## Deleting a CA

You can delete a private CA. To prevent misoperations, the PCA service offers different policies for you to respond to the deletion of CAs in different statuses.

- **Disabled** and **Expired**: Only scheduled deletion is allowed. You can schedule a delay of 7 to 30 days for actual deletion of a CA. During the scheduled deletion period, the CA is in the **Pending deletion** status. If a CA is in the **Pending deletion** status, you can cancel the deletion to restore the certificate to the **Disabled** or **Expired** status. Once the scheduled deletion time is triggered, the CA is deleted as planned and cannot be restored.
- **Pending activation** or **Revoked**: CAs in these statuses can only be deleted immediately. Once a CA is deleted, it is deleted immediately and cannot be restored.
- **Activated**: An activated CA cannot be directly deleted. To delete it, disable it first.

---

⚠ CAUTION

After a private CA is deleted permanently, all certificates under it cannot be revoked, all private certificates issued by it or its subordinate CAs cannot be exported, and the CRL cannot be updated. Exercise caution when performing this operation.

- Before deleting a private CA, check whether the private CA is still in use and whether your PKI system will be unavailable after the deletion.
- Before deleting a private CA, if the private CA is no longer used, revoke all its certificates that have not expired and remove them from the trust list of all terminals. (If the private CA is a subordinate CA, revoke it and then delete it.)

---

## Canceling Deletion of A Private CA

Restore the private CA in the **Pending deletion** status to the state before the deletion.

---

⚠ CAUTION

If you cancel a scheduled deletion, the pending deletion period of the private CA will still be billed. Exercise caution when performing this operation.

---

### Revoking a Private CA

You can revoke a subordinate CA that is no longer used or whose key material has been leaked. A revoked subordinate CA is useless and cannot be restored. If the CRL configuration is enabled for the parent CA, you can query the revocation information in the CRL of the parent CA.

CAUTION

- Revoking a CA is a risky operation. Exercise caution when performing this operation.
- During the validation, the certificate revocation list (CRL) is queried to check whether the certificate is revoked. Otherwise, a revoked certificate may be used during communications, which incurs security risks.
- If a private CA is revoked, all certificates issued by it or its child CAs are put into the CRL and no longer trusted. Any validation of certificate chains containing the revoked private CA fails.

### Procedure for Handling CA Expiration

When a private CA expires, the private CA status will be changed to **Expired**.

## 3.2.4 Managing a CRL

The PCA service has the following restrictions on certificate revocation list (CRL) management:

- The CRL is released only when the CRL configuration is enabled during the creation of a private CA.

  NOTICE

  If the parent CA does not enable CRL configuration, revoked certificates will not be put into a CRL. This means a revoked certificate can still pass certificate chain validation, which incurs security risks. If you expect to revoke certificates, enable the CRL configuration.

- CRLs can be released only to the OBS bucket authorized by you. Customizing other storage paths is not allowed.
- After CRL configuration is enabled, the access policy of a signed CRL depends on the access policies you configure for the OBS bucket that storing the CRL. You can customize the access policy for the authorized bucket.
- Once a certificate is revoked, it cannot be restored.
- Revoked certificates are not trusted as their information is written into the CRL.
- After a certificate is revoked, the PCA service writes the certificate information into the CRL (if the CRL is enabled by the parent CA) within 30 minutes and updates the CRL on the OBS bucket. If the CRL fails to be released, the system attempts to generate the CRL again 15 minutes later.

- A scheduled task for releasing new CRLs will fail to be executed in any of the following cases: The private CA has been deleted; the private CA has expired; the OBS bucket has been deleted; or the authorization for the OBS bucket has been canceled.
- If the private CA does not revoke any sub-certificate within the validity period of the CRL, a new CRL is generated only after the validity period expires (which may be delayed for about 30 minutes). The validity period of a CRL can be 7 to 30 days.
- Appropriate revocation reasons can make the revocation information in the CRL more accurate.

  The default revocation reason in the PCA service is in the **UNSPECIFIED** field. **Table 3-5** describes the revocation reasons you can select.

**Table 3-5** Revocation reasons and meaning

| Reason for Revocation | Reason Code in RFC 5280 | Description |
|---|---|---|
| UNSPECIFIED | 0 | Default value. No reason is specified for revocation. |
| KEY_COMPROMISE | 1 | The certificate key material has been leaked. |
| CERTIFICATE_AUTHORITY_COMPROMISE | 2 | Key materials of the CA have been leaked in the certificate chain. |
| AFFILIATION_CHANGED | 3 | The subject or other information in the certificate has been changed. |
| SUPERSEDED | 4 | The certificate has been replaced. |
| CESSATION_OF_OPERATION | 5 | The entity in the certificate or certificate chain has ceased to operate. |
| CERTIFICATE_HOLD | 6 | The certificate should not be considered valid currently and may take effect in the future. |
| PRIVILEGE_WITHDRAWN | 9 | The certificate no longer has the right to declare its listed attributes. |

| Reason for Revocation | Reason Code in RFC 5280 | Description |
|---|---|---|
| ATTRIBUTE_AUTHORITY_COMPROMISE | 10 | The authority that warrants the attributes of the certificate may have been compromised. |

NOTE

The naming of revocation reasons in the PCA service is different from that in international standards. You can use the revocation reason code to query the description of revocation reasons in **RFC 5280**.

## 3.2.5 Rotating a Private CA

- Private CA rotation is a process of replacing a CA that is about to expire with a new one.
- The administrator of a private CA must set a proper validity period for the private CA.

  If the validity period is too long, the risk of key material leakage increases. If the validity period is too short, the private CA is frequently rotated, increasing the service overhead.
- To ensure smooth service switchover, plan the rotation scheme of private CAs.

### Procedure

**Step 1** Create a CA, disable the old CA, and do not use the old CA to issue certificates. Use the new CA to issue new certificates, replace the certificates issued by the old CA with the new certificates, and deploy the certificates on the corresponding service nodes.

NOTICE

- Before the old CA is replaced, the service system must trust both the new and old CAs.
- If a subordinate CA is to be replaced, the service node can automatically trust both the new and old CAs as long as the new and old subordinate CAs have the same root CA.
- If the root CA is to be replaced, put the new root CA in the trusted root certificate list of the service node before replacing the old one to ensure that the newly issued certificate is trusted.

**Step 2** After the new certificate is in place, revoke and delete the old certificate, including the old CA.

**----End**

NOTE

- A proper periodic private CA rotation scheme can ensure that certificates are continuously updated and prevent private keys from being cracked. An emergency private CA rotation scheme can prevent service loss caused by emergencies, such as private key leaks and CAs that become untrusted.
- The new CA should have some identifiable version tags, such as ROOT CA G0----->ROOT CA G1, added to the subject name so that the new and old CAs can be quickly identified during private CA rotation.

## 3.3 Best Practices of PCA Code Examples

### 3.3.1 Prerequisites

Prepare basic authentication information.

- **ACCESS_KEY**: access key of the Huawei Cloud account
- **SECRET_ACCESS_KEY**: secret access key of the Huawei Cloud account
- **DOMAIN_ID**: Huawei Cloud account ID. For details, see **Basic Concepts**.
- **CCM_ENDPOINT**: CCM endpoints (PCA is included in CCM). For details, see **Regions and Endpoints**.

NOTE

- Huawei Cloud provides unified SDKs. You can add dependencies or download SDKs to call Huawei Cloud APIs to access Huawei Cloud applications, resources, and data.
- For details about the unified SDKs for the PCA service, see **SDK Center**.
- For details about how to use unified SDKs, see **Cloud Java Software Development Kits (Java SDK)**.

### 3.3.2 Example Code for Managing Private CAs

#### 3.3.2.1 Creating a CA

A maximum of 1,000 CAs can be created for each user.

For details about the parameters for creating a private CA, see **Parameters for Creating a CA**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.CreateCertificateAuthorityRequest;
import com.huaweicloud.sdk.ccm.v1.model.CreateCertificateAuthorityRequestBody;
import com.huaweicloud.sdk.ccm.v1.model.CreateCertificateAuthorityResponse;
import com.huaweicloud.sdk.ccm.v1.model.CrlConfiguration;
import com.huaweicloud.sdk.ccm.v1.model.DistinguishedName;
import com.huaweicloud.sdk.ccm.v1.model.Validity;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

/**
* Create a CA.
 */
public class CreateCertificateAuthorityExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before
running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =  System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

       // 3. Make request parameters.
       // (1) Type of the CA certificate you want to create. ROOT for root CAs and SUBORDINATE for
subordinate CAs
        String CAType = "ROOT";
        // (2) CA key algorithm
        String keyAlgorithm = "RSA2048";
        // Signature hash algorithm
        String signatureAlgorithm = "SHA512";

        /*
         * (4) Determining CA validity period
         * - type: time type. The options are YEAR, MONTH, DAY, and HOUR.
        * - value: corresponding value.
         */
        Validity validity = new Validity();
        validity.setType("YEAR");
        validity.setValue(20);

        /*
        * (5) Define the unique identifier of the CA.
        * - organization: organization name.
        * - organizationalUnit: department name.
         * - country: abbreviation of a country. The value can contain only two characters, for example, US for
the United States.
        * - state: province or city name.
        * - locality: city name.
        * - commonName: CA name (CN)
         */
        DistinguishedName subjectInfo = new DistinguishedName();
        subjectInfo.setOrganization("your organization");
        subjectInfo.setOrganizationalUnit("your organizational unit");
        subjectInfo.setCountry("CN");
        subjectInfo.setState("your state");
        subjectInfo.setLocality("your locality");
        subjectInfo.setCommonName("your CA name");

        /*
        * (6) CRL configuration information
         * - enabled: whether to enable the CRL configuration.
        * - obsBucketName: OBS bucket name, which is used to release the CRLs. OBS buckets must be
authorized.
        * - crlName: name of the CRL file. If this parameter is not specified, the CA ID is used as the file name
by default.
         * - validDays: CRL update period.
         */
        CrlConfiguration crlConfiguration = new CrlConfiguration();
        crlConfiguration.setEnabled(false);
        crlConfiguration.setObsBucketName("your OBS buck name");
        crlConfiguration.setCrlName("your CRL file name");
        crlConfiguration.setValidDays(7);

       // (7) Assign values to the attributes of the request body.
        CreateCertificateAuthorityRequestBody requestBody = new CreateCertificateAuthorityRequestBody();
        requestBody.setType(CAType);
        requestBody.setKeyAlgorithm(keyAlgorithm);
        requestBody.setSignatureAlgorithm(signatureAlgorithm);
        requestBody.setValidity(validity);
        requestBody.setDistinguishedName(subjectInfo);
        requestBody.setCrlConfiguration(crlConfiguration);

       // 4. Construct a request body.
        CreateCertificateAuthorityRequest request = new
CreateCertificateAuthorityRequest().withBody(requestBody);

       // 5. Start to send the request.
        CreateCertificateAuthorityResponse response;
        try {
            response = ccmClient.createCertificateAuthority(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

       // 6. Obtain the ID of the CA that is successfully created.
        String caId = response.getCaId();

        System.out.println(caId);
    }

}
```

### 3.3.2.2 Deleting a CA

Only private CAs in the **Pending activation**, **Revoked**, **Disabled**, or **Expired** state can be deleted.

Private CAs in the **Pending activation** or **Revoked** state can be directly deleted. Private CAs in the **Disabled** or **Expired** state can only be deleted by scheduled deletion tasks.

For details, see **Parameters for Deleting a CA**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.DeleteCertificateAuthorityRequest;
import com.huaweicloud.sdk.ccm.v1.model.DeleteCertificateAuthorityResponse;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

/**
 * Delete a CA.
 * (1) Only private CAs in the Pending activation, Revoked, Disabled, or Expired state can be deleted.
 * (2) Private CAs in the PENDING or REVOKED state will be directly deleted. Private CAs in the DISABLED
or EXPIRED state can only be deleted by scheduled deletion tasks.
 */
public class DeleteCertificateAuthorityExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before
running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

       // 3. Make request parameters.
      // (1) ID of the CA to be disabled
        String caId = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";
       // (2) Time for delaying the deletion. Note: This parameter is a string.
        String pendingDays = "7";

       // 4. Construct a request body.
        DeleteCertificateAuthorityRequest request = new DeleteCertificateAuthorityRequest()
                .withCaId(caId)
                .withPendingDays(pendingDays);

       // 5. Start to send the request.
        DeleteCertificateAuthorityResponse response;
        try {
            response = ccmClient.deleteCertificateAuthority(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6. Obtain the response message. After the deletion succeeds, no response is returned and the
returned status code is 204.
        System.out.println(response.getHttpStatusCode());
    }

}
```

### 3.3.2.3 Disabling a CA

A disabled CA cannot be used to issue certificates, but can be used to revoke certificates and issue CRLs.

For details, see **Parameters for Disabling a CA**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.DisableCertificateAuthorityRequest;
import com.huaweicloud.sdk.ccm.v1.model.DisableCertificateAuthorityResponse;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

/**
 * A disabled CA cannot be used to issue certificates, but can be used to revoke certificates and issue CRLs.
 */
public class DisableCertificateAuthorityExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before
running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY =  System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

       // 3. Make request parameters.
      //ID of the CA certificate to be disabled
        String caId = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";

       // 4. Construct a request body.
        DisableCertificateAuthorityRequest request = new DisableCertificateAuthorityRequest()
                .withCaId(caId);

       // 5. Start to send the request.
        DisableCertificateAuthorityResponse response;
        try {
            response = ccmClient.disableCertificateAuthority(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6. Obtain the response message. After the CA is disabled, no response is returned and the returned
status code is 204.
        System.out.println(response.getHttpStatusCode());
    }

}
```

### 3.3.2.4 Enabling a CA

A private CA in the **Disabled** state can be enabled. After it is enabled, its status changes to **Activated**.

For details, see **Parameters for Enabling a CA**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.EnableCertificateAuthorityRequest;
import com.huaweicloud.sdk.ccm.v1.model.EnableCertificateAuthorityResponse;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

/**
 * Enable the CA and change the CA status from Disabled to Activated. */
public class EnableCertificateAuthorityExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before
running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY =  System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

       // 3. Make request parameters.
      // ID of the CA you want to enable.
        String caId = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";

       // 4. Construct a request body.
        EnableCertificateAuthorityRequest request = new EnableCertificateAuthorityRequest()
                .withCaId(caId);

       // 5. Start to send the request.
        EnableCertificateAuthorityResponse response;
        try {
            response = ccmClient.enableCertificateAuthority(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6. Obtain the response message. After the CA is enabled, no response is returned and the returned
status code is 204.
        System.out.println(response.getHttpStatusCode());
    }

}
```

### 3.3.2.5 Exporting a CA

You can export a private CA certificate, including the certificate body and certificate chain.

**NOTE**

A root certificate is self-signed, and the certificate chain is null.

For details, see **Parameters for Exporting a CA**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.ExportCertificateAuthorityCertificateRequest;
import com.huaweicloud.sdk.ccm.v1.model.ExportCertificateAuthorityCertificateResponse;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

/**
 * Export a CA certificate, including the certificate body and certificate chain. A root CA certificate is self-
signed, so the certificate chain is null.
 */
public class ExportCertificateAuthorityExample {
    /**
     * Basic authentication information:
```

```
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before
running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY =  System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

       // 3. Make request parameters.
      //ID of the CA certificate you want to export.
        String caId = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";

       // 4. Construct a request body.
        ExportCertificateAuthorityCertificateRequest request = new
ExportCertificateAuthorityCertificateRequest()
                .withCaId(caId);

       // 5. Start to send the request.
        ExportCertificateAuthorityCertificateResponse response;
        try {
            response = ccmClient.exportCertificateAuthorityCertificate(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

       // 6. Obtain the response message.
       // (1) Obtain the certificate body in PEM format.
        String caCertificate = response.getCertificate();
       // (2) Obtain the certificate chain in PEM format.
        String caCertificateChain = response.getCertificateChain();

        System.out.println(response);
    }

}
```

### 3.3.2.6 Canceling Deletion of a CA

You can cancel the deletion of a private CA. The CA status will change from **Pending deletion** to the status it is in before the deletion.

For details, see **Parameters for Canceling the Deletion of a CA**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.RestoreCertificateAuthorityRequest;
import com.huaweicloud.sdk.ccm.v1.model.RestoreCertificateAuthorityResponse;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

/**
```

```
 * Cancel the scheduled deletion of a CA to change the CA status from Pending deletion to Disabled.
 */
public class RestoreCertificateAuthorityExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before
running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY =  System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

       // 3. Make request parameters.
      // ID of the CA certificate you want to operate.
        String caId = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";

       // 4. Construct a request body.
        RestoreCertificateAuthorityRequest request = new RestoreCertificateAuthorityRequest()
                .withCaId(caId);

       // 5. Start to send the request.
        RestoreCertificateAuthorityResponse response;
        try {
            response = ccmClient.restoreCertificateAuthority(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6. Obtain the response message. After the schedule deletion is cancelled, no response is returned
and the returned status code is 204.
        System.out.println(response.getHttpStatusCode());
    }

}
```

### 3.3.2.7 Obtaining CA Details

Obtain private CA details, including the private CA name, department name, type, and status.

For details, see **Parameters for Obtaining Private CA Details**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.ShowCertificateAuthorityRequest;
import com.huaweicloud.sdk.ccm.v1.model.ShowCertificateAuthorityResponse;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

/**
```

```
 *  Obtain details of a CA.
 */
public class ShowCertificateAuthorityExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before
running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

       // 3. Make request parameters.
      // ID of the CA certificate you want to query.
        String caId = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";

       // 4. Construct a request body.
        ShowCertificateAuthorityRequest request = new ShowCertificateAuthorityRequest()
                .withCaId(caId);

       // 5. Start to send the request.
        ShowCertificateAuthorityResponse response;
        try {
            response = ccmClient.showCertificateAuthority(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

       // 6. Obtain the response message.
      // Obtain the certificate status. Use the corresponding Getter function to obtain the values of other
attributes.
        String caStatus = response.getStatus();

        System.out.println(response);
    }

}
```

### 3.3.2.8 Querying CA Quotas

Query the total number of CA quotas and the number of used CA quotas.

For details, see **Parameters for Querying CA Quotas**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.Resources;
import com.huaweicloud.sdk.ccm.v1.model.ShowCertificateAuthorityQuotaRequest;
import com.huaweicloud.sdk.ccm.v1.model.ShowCertificateAuthorityQuotaResponse;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;
import java.util.List;

/**
 * Query the CA quota.
 */
public class ShowCertificateAuthorityQuotaExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before
running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =  System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

       // 3. Construct a request body.
        ShowCertificateAuthorityQuotaRequest request = new ShowCertificateAuthorityQuotaRequest();

       // 4. Start to send the request.
        ShowCertificateAuthorityQuotaResponse response;
        try {
            response = ccmClient.showCertificateAuthorityQuota(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

       // 5. Obtain the quota usage.
        List<Resources> quotas = response.getQuotas().getResources();
      // Total quota
        int caQuota = quotas.get(0).getQuota();
       //Used quota
        int used = quotas.get(0).getUsed();

        System.out.println(response);
    }

}
```

## 3.3.3 Example Code for Managing Private Certificates

### 3.3.3.1 Applying for a Certificate

To apply for a private certificate, you must have a private CA that is in the **Activated** state.

For details, see **Parameters for Applying for a Certificate**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.CertDistinguishedName;
import com.huaweicloud.sdk.ccm.v1.model.CreateCertificateRequest;
import com.huaweicloud.sdk.ccm.v1.model.CreateCertificateRequestBody;
import com.huaweicloud.sdk.ccm.v1.model.CreateCertificateResponse;
import com.huaweicloud.sdk.ccm.v1.model.ExtendedKeyUsage;
import com.huaweicloud.sdk.ccm.v1.model.SubjectAlternativeName;
import com.huaweicloud.sdk.ccm.v1.model.Validity;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

import java.util.ArrayList;
import java.util.List;

/**
* To issue a private certificate, there is at least one private CA in the Activated state.
 */
public class createCertificateExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before running this example, configure environment variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =  System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

       // 3. Make request parameters.
       // (1) ID of the CA that issues the private certificates. The CA must be in the ACTIVED state.
        String issuerId = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";
        // (2) Private key algorithm
        String keyAlgorithm = "RSA2048";
        // Signature hash algorithm
        String signatureAlgorithm = "SHA512";

        /*
         * (4) Determine the certificate validity period.
         * - type: time type. The options are YEAR, MONTH, DAY, and HOUR.
        * - value: corresponding value.
         */
        Validity validity = new Validity();
        validity.setType("MONTH");
        validity.setValue(2);

        /*
        * (5) Define the unique identifier of the CA certificate.
        * - organization: organization name.
        * - organizationalUnit: department name.
         * - country: abbreviation of a country. The value can contain only two characters, for example, US for the United States.
        * - state: province or city name.
        * - locality: city name.
        * - commonName: CA name (CN)
         */
        CertDistinguishedName subjectInfo = new CertDistinguishedName();
        subjectInfo.setOrganization("your organization");
        subjectInfo.setOrganizationalUnit("your organizational unit");
        subjectInfo.setCountry("CN");
        subjectInfo.setState("your state");
        subjectInfo.setLocality("your locality");
        subjectInfo.setCommonName("your dns");

        /*
        * (6) Key usage. Generally, only keyAgreement and digitalSignature are specified for server 
certificates, which are optional.
        * - digitalSignature: The key is used as a digital signature.
         * - nonRepudiation: The key is used for non-repudiation.
        * - keyEncipherment: The key is used to encrypt key data.
        * - dataEncipherment: The key is used to encrypt data.
            - **keyAgreement**: The key is used for key negotiation.
          * - keyCertSign: The key can issue certificates.
          * - cRLSign: The key can issue a certificate revocation list (CRL).
         * - encipherOnly: The key is used only for encryption.
         * - decipherOnly: The key is used only for decryption.
         */
        List<String> keyUsages = new ArrayList<>();
        keyUsages.add("digitalSignature");
        keyUsages.add("keyAgreement");

        /*
         * (7) Alternative name of the subject: DNS, IP address, URI, and email address are supported 
currently. This parameter is optional.
         *  SubjectAlternativeName:
         *        type: type
        *         value: corresponding value.
         */
        List<SubjectAlternativeName> subjectAlternativeName = new ArrayList<>();
      // a. Add a standby DNS server.
        SubjectAlternativeName alterNameDNS = new SubjectAlternativeName();
        alterNameDNS.setType("DNS");
        alterNameDNS.setValue("*.example.com");
        subjectAlternativeName.add(alterNameDNS);
     // b. Add a standby IP address.
        SubjectAlternativeName alterNameIP = new SubjectAlternativeName();
        alterNameIP.setType("IP");
        alterNameIP.setValue("127.0.0.1");
        subjectAlternativeName.add(alterNameIP);
     // b. Add a standby email.
        SubjectAlternativeName alterNameEmail = new SubjectAlternativeName();
        alterNameEmail.setType("EMAIL");
        alterNameEmail.setValue("myEmail@qq.com");
        subjectAlternativeName.add(alterNameEmail);
        ExtendedKeyUsage extendedKeyUsage = new ExtendedKeyUsage();
        extendedKeyUsage.setClientAuth(true);
        extendedKeyUsage.setServerAuth(true);


       // (8) Assign values to the attributes of the request body.
       // For details about the restrictions on the value of each attribute, see https://
support.huaweicloud.com/en-us/api-ccm/CreateCertificate.html.
        CreateCertificateRequestBody requestBody = new CreateCertificateRequestBody();
        requestBody.setIssuerId(issuerId);
        requestBody.setKeyAlgorithm(keyAlgorithm);
        requestBody.setSignatureAlgorithm(signatureAlgorithm);
        requestBody.setValidity(validity);
        requestBody.setDistinguishedName(subjectInfo);
        requestBody.setKeyUsages(keyUsages);
        requestBody.setSubjectAlternativeNames(subjectAlternativeName);
        requestBody.setExtendedKeyUsage(extendedKeyUsage);
       // 4. Construct a request body.
        CreateCertificateRequest request = new CreateCertificateRequest()
                .withBody(requestBody);

       // 5. Start to send the request.
        CreateCertificateResponse response;
        try {
            response = ccmClient.createCertificate(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

       // 6. Obtain the response message.
        String certId = response.getCertificateId();
        System.out.println(certId);
   }

}
```

### 3.3.3.2 Deleting a Certificate

Private certificates cannot be deleted by scheduled deletion tasks. When you delete a private certificate, it will be directly deleted.

For details, see **Parameters for Deleting a Certificate**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.DeleteCertificateRequest;
import com.huaweicloud.sdk.ccm.v1.model.DeleteCertificateResponse;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

/**
 * When you delete a certificate, it will be directly deleted. Scheduled deletion is not supported for private 
certificates.
 */
public class DeleteCertificateExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the 
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before 
running this example, configure environment variables HUAWEICLOUD_SDK_AK and 
HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =  System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

        // 3. Make request parameters.
      // (1) ID of the private certificate you want to delete.
        String certId = "5554a381-af92-4336-a943-811396c87616";

       // 4. Construct a request body.
        DeleteCertificateRequest request = new DeleteCertificateRequest().withCertificateId(certId);

       // 5. Start to send the request.
        DeleteCertificateResponse response;
        try {
            response = ccmClient.deleteCertificate(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6. Obtain the response message. After the deletion succeeds, no response is returned and the
returned status code is 204.
        System.out.println(response.getHttpStatusCode());
    }

}
```

## 3.3.3.3 Exporting a Certificate

You can export a private certificate, including the certificate body and certificate chain. You can export the certificate in the format you need.

For details, see **Parameters for Exporting a Certificate**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.ExportCertificateRequest;
import com.huaweicloud.sdk.ccm.v1.model.ExportCertificateRequestBody;
import com.huaweicloud.sdk.ccm.v1.model.ExportCertificateResponse;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

/**
 * Export the private certificate, including the certificate body and certificate chain. You can select the
certificate format.
 */
public class ExportCertificateExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before
running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK.
     */
    private static final String ACCESS_KEY =  System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();
```

```
       // 3. Make request parameters.
      // (1) ID of the end-entity certificate you want to export.
        String certId = "5554a381-af92-4336-a943-811396c87616";

        /*
          (2) Define the export format. (SDKs support only uncompressed files.)
         - isCompressed: whether to compress the file. The value is a string. The options are true and false.
SDKs support only false.
          - type: export format. Currently, only the following formats are supported in calling SDKs:
                APACHE: This parameter is recommended if you want to use the certificate for an Apache server.
                NGINX: This parameter is recommended if you want to use the certificate for an Nginx server.
                  OTHER: This parameter is recommended if you want to download a certificate in PEM format.
         */
        ExportCertificateRequestBody requestBody = new ExportCertificateRequestBody();
        requestBody.setType("NGINX");
        requestBody.setIsCompressed("false");

       // 4. Construct a request body.
        ExportCertificateRequest request = new ExportCertificateRequest()
                .withCertificateId(certId)
                .withBody(requestBody);

       // 5. Start to send the request.
        ExportCertificateResponse response;
        try {
            response = ccmClient.exportCertificate(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

       // 6. Obtain the response message.
       // (1) Obtain the certificate body in PEM format.
        String certificate = response.getCertificate();
       // (2) Obtain the certificate chain in PEM format.
        String certificateChain = response.getCertificateChain();
        System.out.println(response);
    }

}
```

### 3.3.3.4 Revoking a Certificate

All its records will be cleared and cannot be recovered, including private CA records. Therefore, exercise caution when performing this operation.

For details, see **Parameters for Revoking a Private Certificate**.

```
import com.huaweicloud.sdk.ccm.v1.CcmClient;
import com.huaweicloud.sdk.ccm.v1.model.RevokeCertificateRequest;
import com.huaweicloud.sdk.ccm.v1.model.RevokeCertificateRequestBody;
import com.huaweicloud.sdk.ccm.v1.model.RevokeCertificateResponse;
import com.huaweicloud.sdk.core.auth.GlobalCredentials;

/**
 * Revoke a private certificate.
 */
public class RevokeCertificateExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account
     * - DOMAIN_ID: Huawei Cloud account ID.
    * - CCM_ENDPOINT: Endpoint address for accessing Huawei Cloud CCM (PCA is included in CCM).
......*Hard-coded or plaintext AK and SK are risky. For security, encrypt your AK and SK and store them in the
configuration file or environment variables.
     * In this example, the AK and SK are stored in environment variables for identity authentication. Before
running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK.
```

```
  */
    private static final String ACCESS_KEY =  System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String DOMAIN_ID = "<DomainID>";
    private static final String CCM_ENDPOINT = "<CcmEndpoint>";

    public static void main(String[] args) {
      // 1. Prepare the credentials for accessing Huawei Cloud. PCA is a global service.
        final GlobalCredentials auth = new GlobalCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withDomainId(DOMAIN_ID);

        // 2. Initialize the SDK and transfer the credentials and endpoint address of CCM.
        final CcmClient ccmClient = CcmClient.newBuilder()
                .withCredential(auth)
                .withEndpoint(CCM_ENDPOINT).build();

       // 3. Make request parameters.
      // (1) ID of the end-entity certificate you want to revoke.
        String certId = "5554a381-af92-4336-a943-811396c87616";

      // (2) Enter the revocation reason (an enumerated value. For details, see the corresponding parameter
in the API document). The corresponding revocation code will be written into the certificate revocation list.
      //Default value: UNSPECIFIED
        RevokeCertificateRequestBody requestBody = new RevokeCertificateRequestBody();
        requestBody.setReason("UNSPECIFIED");

       // 4. Construct a request body.
        RevokeCertificateRequest request = new RevokeCertificateRequest()
                .withCertificateId(certId)
                .withBody(requestBody);

       // 5. Start to send the request.
        RevokeCertificateResponse response;
        try {
            response = ccmClient.revokeCertificate(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6. Obtain the response message. After the revocation succeeds, no response is returned and the
returned status code is 204.
        System.out.println(response.getHttpStatusCode());
    }

}
```

## 3.4 Building an Internal Identity Authentication System for Your Organization

### Scenarios

CCM can help you build a complete CA hierarchy so that you can issue and manage self-signed private certificates within your organization.

Now, you can follow the procedure below to apply for a private certificate in CCM and deploy it on your server.

### Background

An internally-hosted complete CA hierarchy is called a self-built Public Key Infrastructure (PKI) system.

When building a PKI system, we need to design its structure based on the application scenarios to facilitate subsequent management.

- CA level design

  A path length determines how many levels of subordinate CAs the current CA can issue. It is used to control the certificate chain length.

  A well-designed CA hierarchy can make fine-grained control over certificates a reality. With PCA, you can build a CA hierarchy with up to seven levels. For details, see **Designing a Private CA Hierarchy**.
- Certificate rotation

  Before a certificate expires, it should be replaced by a new one to prevent service interruption caused by certificate expiration. This process is called certificate rotation.

  The certificate validity period determines the certificate rotation interval. A proper certificate validity period can reduce the risk of key material leakage and certificate rotation costs. For more details, see **PCA Certificate Validity Period**.
- Certificate revocation management

  If the key of a certificate is disclosed or the certificate is no longer for some reasons, the certificate should be revoked. To that end, you need to enable the certificate revocation list (CRL) when you create a CA. Beyond that, your service should be designed to check the certificate revocation status.

## Step 1: Creating a Private Root CA or Subordinate CA

Perform the following steps to create a root CA or subordinate CA. For details about how to create a private CA, see **Creating a Private CA**.

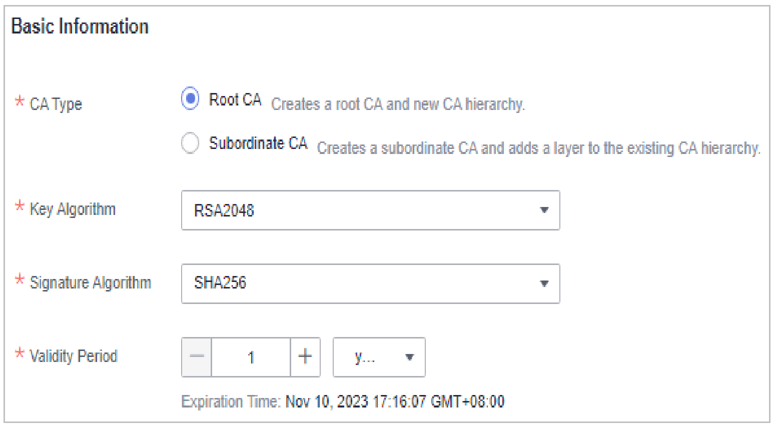
**Step 1** **Log in to the management console.**

**Step 2** Click ☰ in the upper left corner of the page and choose **Security** > **Cloud Certificate Management Service**. In the navigation pane, choose **Private Certificate Management** > **Private CA**.

**Step 3** In the upper left corner of the private CA list, click **Create CA** to switch to the **Create CA** page.

**Step 4** Configure the CA information.

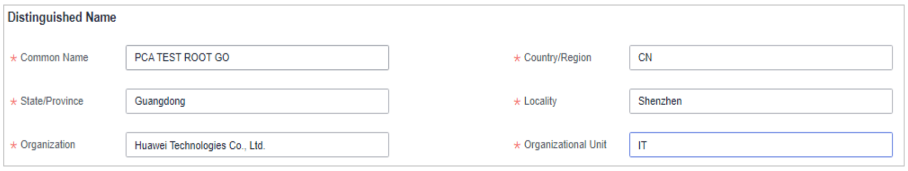
1. Configure basic information.



Basic Information

| | |
|---|---|
| * CA Type | [✓ Root CA] Creates a root CA and new CA hierarchy. |
| | Subordinate CA Creates a subordinate CA and adds a layer to the existing CA hierarchy. |
| * Key Algorithm | RSA2048 |
| * Signature Algorithm | SHA256 |
| * Validity Period | 1 y... |
| | Expiration Time: Nov 10, 2023 17:16:07 GMT+08:00 |

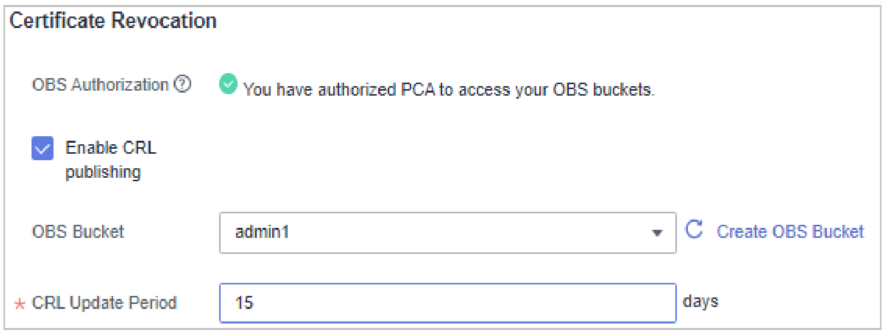
2. Configure the Distinguished Name (DN) of the certificate.



Distinguished Name

| | | | |
|---|---|---|---|
| * Common Name | PCA TEST ROOT GO | * Country/Region | CN |
| * State/Province | Guangdong | * Locality | Shenzhen |
| * Organization | Huawei Technologies Co., Ltd. | * Organizational Unit | IT |

3. Configure certificate revocation details.



Certificate Revocation

| | |
|---|---|
| OBS Authorization | You have authorized PCA to access your OBS buckets. |
| ✓ Enable CRL publishing | |
| OBS Bucket | admin1 |
| * CRL Update Period | 15 days |

**Step 5** Click **Next**.

**Step 6** After confirming the information about the private CA, click **Confirm and Create**.

**----End**

## Step 2: Activating a Subordinate Private CA with its Parent Private CA

A subordinate CA must be activated, or it cannot issue certificates. The following describes how to activate a subordinate CA in a two-level CA hierarchy system by using an internal private CA. To activate a subordinate CA by using an external private CA, refer to **Activating a Private CA**

**Step 1** **Log in to the management console.**

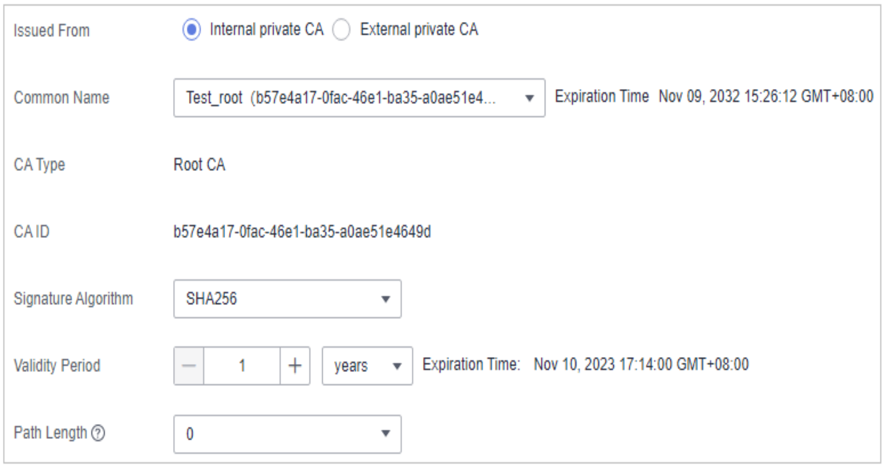
**Step 2** Click ☰ in the upper left corner of the page and choose **Security** > **Cloud Certificate Management Service**. In the navigation pane, choose **Private Certificate Management** > **Private CA**.

**Step 3** Locate the row of the subordinate private CA and click **Activate** in the **Operation** column. In the **Install CA Certificate and Activate CA** page, configure the required parameters. **Figure 3-6** shows an example.

NOTE

The path length varies depending on how many layers of your CA hierarchy has. For details, see **Designing a Private CA Hierarchy**.

**Figure 3-6** A private CA



| | |
|---|---|
| Issued From | (● Internal private CA) ○ External private CA |
| Common Name | Test_root (b57e4a17-0fac-46e1-ba35-a0ae51e4... Expiration Time Nov 09, 2032 15:26:12 GMT+08:00 |
| CA Type | Root CA |
| CA ID | b57e4a17-0fac-46e1-ba35-a0ae51e4649d |
| Signature Algorithm | SHA256 |
| Validity Period | 1 years Expiration Time: Nov 10, 2023 17:14:00 GMT+08:00 |
| Path Length | 0 |

**Step 4** Confirm the configuration and click **OK**.

**----End**

## Step 3: Creating a Private Certificate

For details, see **Applying for a Private Certificate**

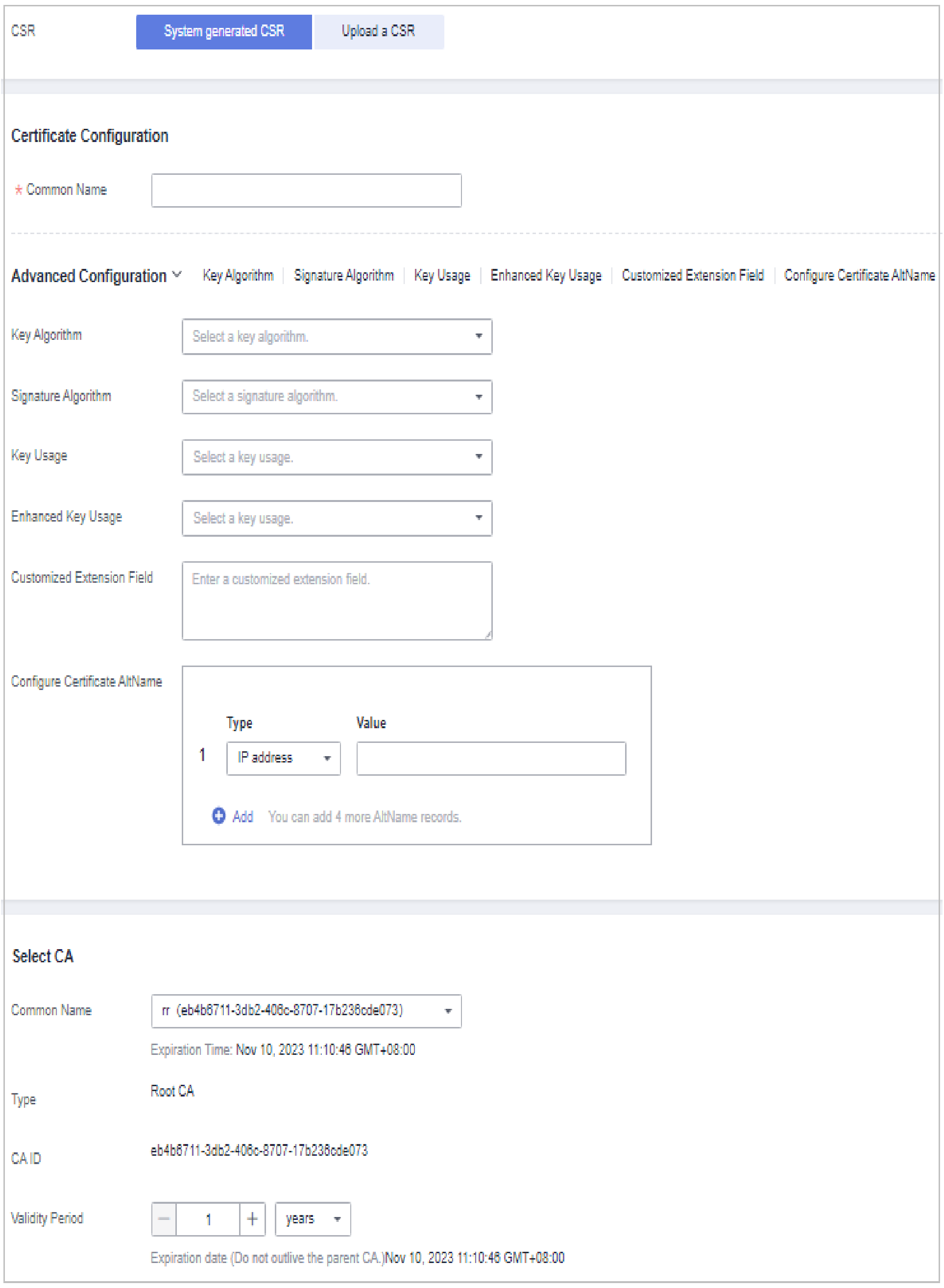
**Step 1** **Log in to the management console.**

**Step 2** Click ☰ in the upper left corner of the page and choose **Security** > **Cloud Certificate Management Service**. In the navigation pane, choose **Private Certificate Management** > **Private Certificate**.

**Step 3** In the upper left corner of the private certificate list, click **Apply for Certificate**.

**Step 4** Enter required certificate information.

**Figure 3-7** System generated CSR


CSR: System generated CSR | Upload a CSR

Certificate Configuration

* Common Name

Advanced Configuration: Key Algorithm | Signature Algorithm | Key Usage | Enhanced Key Usage | Customized Extension Field | Configure Certificate AltName

Key Algorithm: Select a key algorithm.

Signature Algorithm: Select a signature algorithm.

Key Usage: Select a key usage.

Enhanced Key Usage: Select a key usage.

Customized Extension Field: Enter a customized extension field.

Configure Certificate AltName

| | Type | Value |
|---|---|---|
| 1 | IP address | |

Add You can add 4 more AltName records.

Select CA

Common Name: rr (eb4b6711-3db2-406c-8707-17b236cde073)

Expiration Time: Nov 10, 2023 11:10:46 GMT+08:00

Type: Root CA

CA ID: eb4b6711-3db2-406c-8707-17b236cde073

Validity Period: 1 years

Expiration date (Do not outlive the parent CA.)Nov 10, 2023 11:10:46 GMT+08:00

**Step 5** Confirm the information and click **OK**.

After you submit your application, the system will return to the private certificate list page. Message "Certificate xxx applied for successfully." is displayed in the upper right corner of the page, indicating that the private certificate application is successful.

**----End**

## Step 4: Trusting the Root CA.

Before installing a private certificate, you need to add the root CA to the trusted root certificate authorities of the client or server.

- One-way authentication

  To win more trust from the client for your server, you need to add the root CA for the server certificate to the client-end trusted CAs.
- Two-way authentication

  To enable two-way authentication between a server and a client, each side needs to add the root CA of the other side to their own trusted root CA store.

Download the private root CA certificate and obtain a root CA certificate file named ***Root CA name_certificate.pem***. For details, see **Exporting a Private Root CA**.

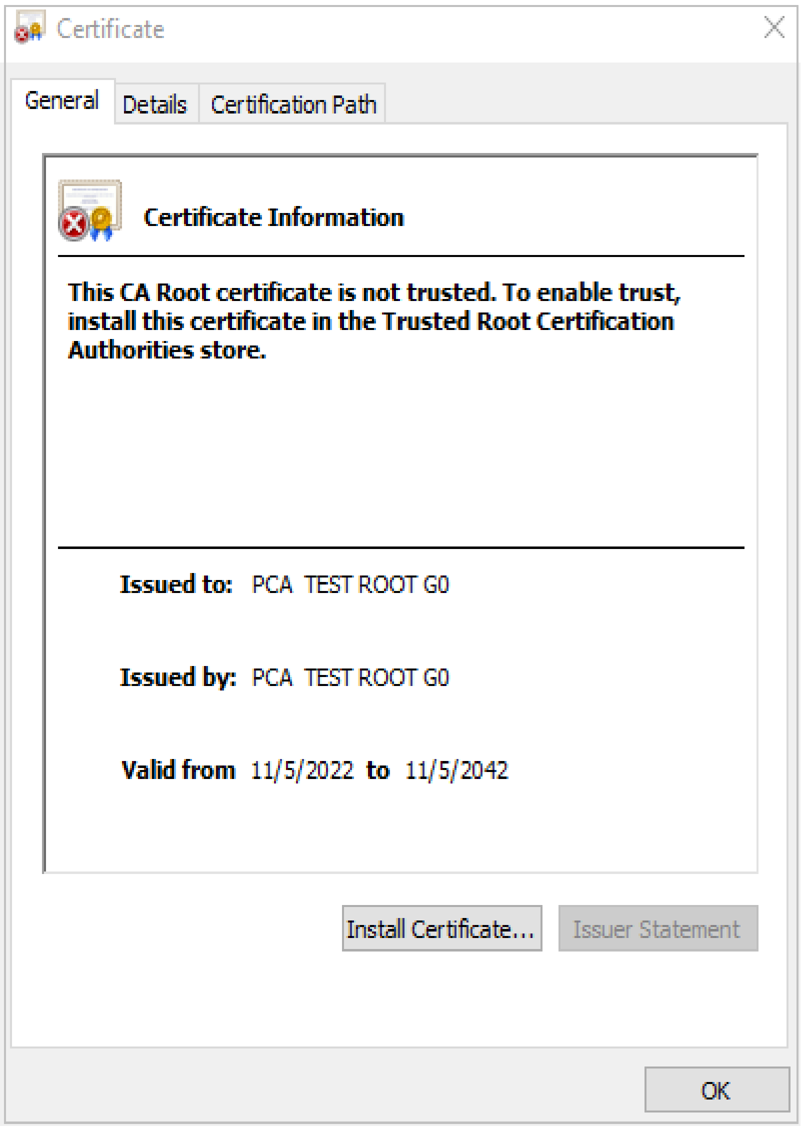
Use either of the following methods to add the root CA to the trusted root CA store based on the operating system:

NOTE

Root CA **PCA TEST ROOT GO** is used as an example.

- **For Windows OSs**
  a. Change the file name extension of the root CA certificate from .pem to .crt. and double-click the certificate file. The root CA certificate information shows that the root certificate is untrusted.

**Figure 3-8** Root CA not trusted



Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: PCA TEST ROOT G0

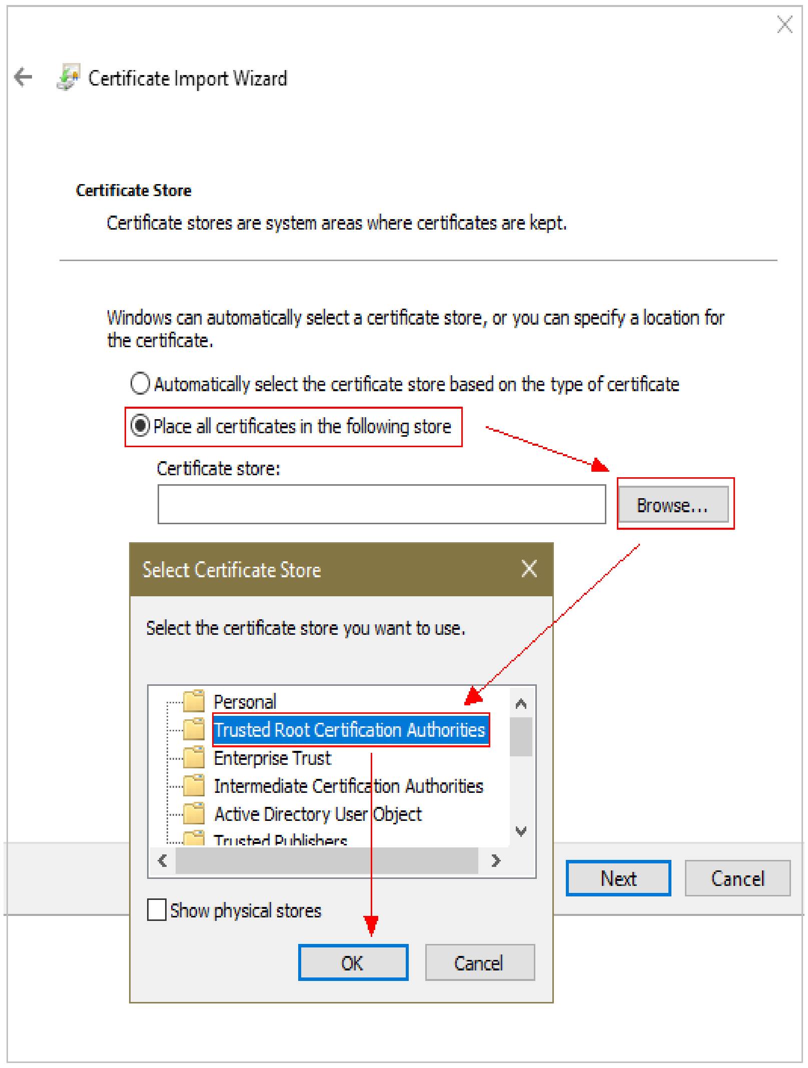
Issued by: PCA TEST ROOT G0

Valid from 11/5/2022 to 11/5/2042

b. Click **Install Certificate**, select a certificate storage location based on the certificate usage, and click **Next**.

c. Select **Place all certificates in the following store** and click **Browse**. Then, select **Trusted Root Certification Authorities** and click **OK**.

**Figure 3-9** Storing a root certificate



Certificate Import Wizard

**Certificate Store**

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

○ Automatically select the certificate store based on the type of certificate

● Place all certificates in the following store

Certificate store:

Select Certificate Store

Select the certificate store you want to use.

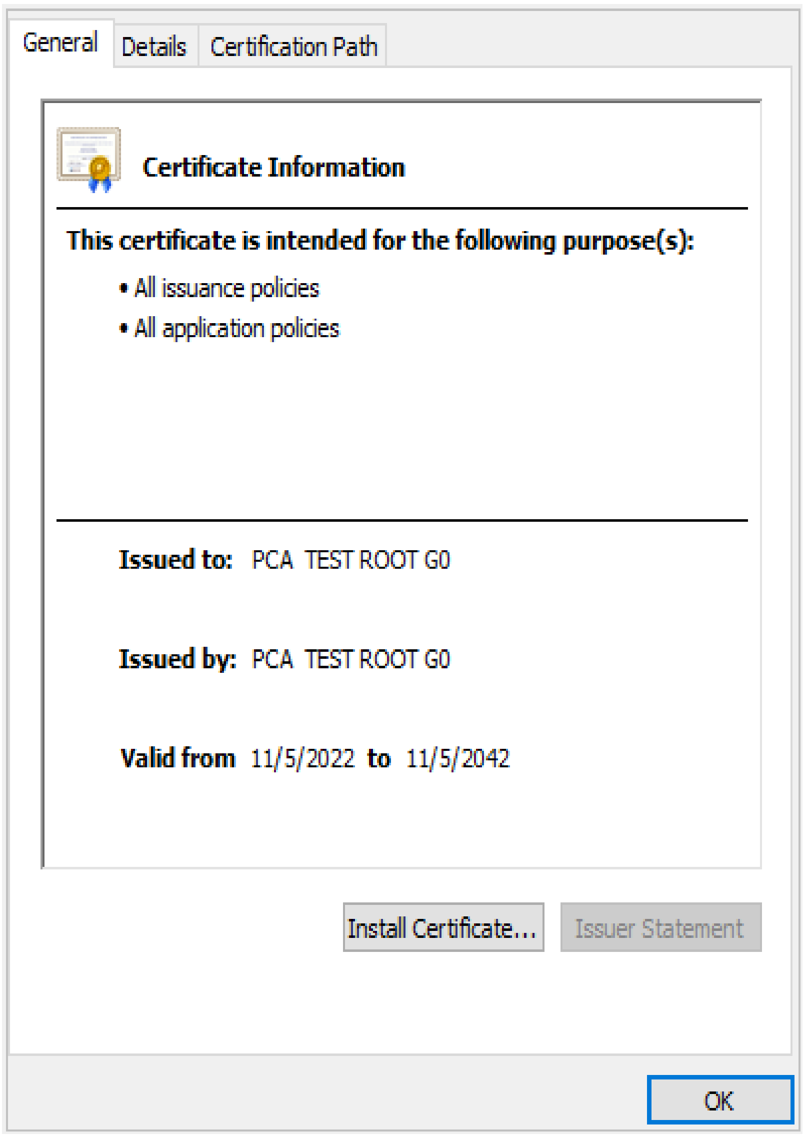
- Personal
- Trusted Root Certification Authorities
- Enterprise Trust
- Intermediate Certification Authorities
- Active Directory User Object
- Trusted Publishers

☐ Show physical stores

d. Click **Next**, and then click **OK**. A dialog box is displayed, indicating that Windows will trust all certificates issued by the private root CA. Click **Yes**.

e. Double-click the root CA certificate file. If the **Certificate Information** area shows that the system trusts the root CA certificate, the root CA is added to the trusted root CAs.

**Figure 3-10** Trusted root CA



**Certificate Information**

**This certificate is intended for the following purpose(s):**

- All issuance policies
- All application policies

**Issued to:** PCA TEST ROOT G0

**Issued by:** PCA TEST ROOT G0

**Valid from** 11/5/2022 **to** 11/5/2042

- **For Linux OSs**

  The path for and method of storing root CA certificates vary depending on Linux OS versions. The following procedure use CentOS 6 as an example:

  a. Copy the root CA certificate file to the **/home/** directory.

  b. If **ca-certificates** is not installed on the server, run the following command to install **ca-certificates**:

     **yum install ca-certificates**

  c. Copy the root CA certificate to the **/etc/pki/ca-trust/source/anchors/** directory:

     **cp /home/root.crt /etc/pki/ca-trust/source/anchors/**

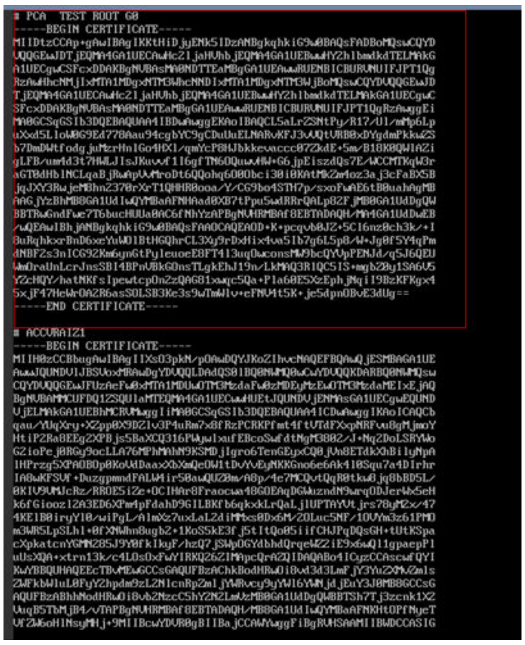
  d. Add the root CA certificate to the trusted root certificate file:

     **update-ca-trust extract**

  e. Check whether the information about the newly added root CA certificate is included in the command output:

     **view /etc/pki/tls/certs/ca-bundle.crt**

**Figure 3-11** Root CA certificate added to the trusted CA list



```
# PCA  TEST ROOT G0
-----BEGIN CERTIFICATE-----
MIIDtzCCAp+gAwIBAgIKKtHiDjyENk5IDzANBgkqhkiG9w0BAQsFADBoMQswCQYD
VQQGEwJDTjEQMA4GA1UECAwHc2l jaHVhbjEQMA4GA1UEBwwHY2hlbmdkdTELMAkG
A1UECgwCSFcxDDAKBgNVBAsMA0NDTTEaMBgGA1UEAwwRUENBICBURUNUIFJPT1Qg
RzAwHhcNMjIxMTA1MDgxNTM3WhcNNDIxMTA1MDgxNTM3WjBoMQswCQYDVQQGEwJD
TjEQMA4GA1UECAwHc2ljaHVhbjEQMA4GA1UEBwwHY2hlbmdkdTELMAkGA1UECgwC
SFcxDDAKBgNVBAsMA0NDTTEaMBgGA1UEAwwRUENBICBURUNUIFJPT1QgRzAwggEi
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCL5aLrZSNtPy/R17/Ul/mMp6Lp
uXxd5LloW0G9Ed778Aau94cgbYC9gCDuUuELNARvKFJ3vUQtURB0xDYgdmPkkwZS
b7DmDWtfodgjuMzrHnlGo4HXl/qmYcP8HJbkkevaccc07ZkdE+5m/B18K0QWlAZi
gLFB/um4d3t7HWLJIsJKuvvf1I6gfTN6OQuwAHW+G6jpEiszdQs7E/WCCMTKqW3r
aGT8dHblNCLqaBjRwApVVMroDt6QQohq6O0Obci30i0KAtMkZm4oz3aj3cFaBX5B
jqJXY3RwjeMBhn2370rXrT1QHHR0ooa/Y/CG9bo4STH7p/sxoFwAE6tB0uahAgMB
AAGjYzBhMB0GA1UdIwQYMBaAFNHAad0XB7tPpu5wdRRrQALp0ZFjMB0GA1UdDgQW
BBTRwGndFwe7T6bucHUUa0AC6fNhYzAPBgNVHRMBAf8EBTADAQH/MA4GA1UdDwEB
/wQEAwIBhjANBgkqhkiG9w0BAQsFAAOCAQEAOD+K+pcqvb0JZ+5Cl6nz0ch3k/+I
0uRqhkxrBnD6xeYuW0lBtHGQhrCL3Xy9rDxHix4va5lb7g6L5p8/W+Jg0f5Y4qPm
dNBFZs3nICG92Km6ynGtPyleuoeE8FT4l3uqOwconsMW9bcQYVpPENJd/q5J6QEU
WmOraUnLcrJnsSBl4BPnVBkGOnsTLgkEhJ19n/LkMAQ3RlQC5IS+mgb28y1SA6V5
YZcHQY/hatNKfslpewtcpOn2zQAG8lxwqc5Qa+Pla60E5XzEphjMqiI9BzKFKgx4
5xjF47HeWrOAZR6asSOLSB3Ke3s9wTmWlv+eFNV4t5K+je5dpnOBvE3dUg==
-----END CERTIFICATE-----

# ACCVRAIZ1
-----BEGIN CERTIFICATE-----
MIIH0zCCBbugAwIBAgIIXsO3pkN/pOAwDQYJKoZIhvcNAQEFBQAwQjESMBAGA1UE
AwwJQUNDVlJBSVoxMRAwDgYDVQQLDAdQS0lBQ0NWMQ0wCwYDVQQKDARBQ0NWMQsw
CQYDVQQGEwJFUzAeFw0xMTA1MDUwOTM3MzdaFw0zMDEyMzEwOTM3MzdaMEIxEjAQ
BgNVBAMMCUFDQ1ZSQUlaMTEQMA4GA1UECwwHUEtJQUNDVjENMAsGA1UECgwEQUND
VjELMAkGA1UEBhMCRVMwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCb
qau/YUqXry+XZpp0X9DZlv3P4uRm7x8fRzPCRKPfmt4ftVTdFXxpNRFvu8gMjmoY
HtiP2Ra8EEg2XPBjs5BaXCQ316PWywlxufEBcoSwfdtNgM3802/J+Nq2DoLSRYWo
G2ioPej0RGy9ocLLA76MPhMAhN9KSMDjIgro6TenGEyxCQ0jVn8ETdkXhBilyNpA
lHPrzg5XPAOBOp0KoVdDaaxXbXmQeOW1tDvYvEyNKKGno6e6Ak4l0Squ7a4DIrhr
IA8wKFSVf+DuzgpmndFALW4ir50awQUZ0m/A8p/4e7MCQvtQqR0tkw8jq8bBD5L/
0KIV9VMJcRz/RROE5iZe+OCIHAr8Fraocwa48GOEAqDGWuzndN9wrqODJerWx5eH
k6fGioozl2A3ED6XPm4pFdahD9GILBKfb6qkxkLrQaLjlUPTAYVtjrs78yM2x/47
4KElB0iryYl0/wiPgL/AlmXz7uxLaL2diMMxs0Dx6M/2OLuc5NF/1OVYm3z61PMO
m3WR5LpSLhl+0fXNWhn8ugb2+1KoS5kE3fj5tItQo05iifCHJPqDQsGH+tUtKSpa
cXpkatcnYGMN285J9Y0fkIkyF/hzQ7jSWpOGYdbhdQrqeWZ2iE9x6wQl1gpaepPl
uUsXQA+xtrn13k/c4LOsOxFwYIRKQ26ZIMApcQrAZQIDAQABo4ICyzCCAscwfQYI
KwYBBQUHAQEEcTBvMEwGCCsGAQUFBzAChkBodHRwOi8vd3d3LmFjY3YuZXMvZmls
ZWFkbWluL0FyY2hpdm9zL2NlcnRpZmljYWRvcy9yYWl6YWNjdjEuY3J0MB8GCCsG
AQUFBzABhhNodHRwOi8vb2NzcC5hY2N2LmVzMB0GA1UdDgQWBBTSh7Tj3zcnk1X2
VuqB5TbMjB4/vTAPBgNVHRMBAf8EBTADAQH/MB8GA1UdIwQYMBaAFNKHtOPfNyeT
Vf2W6oHlNsyMHj+9MIIBcwYDVR0gBIIBajCCAWYwggFiBgRVHSAAMIIBWDCCASIG
```

📖 NOTE

If the OpenSSL version is too old, the configuration may not take effect. You can run the **yum update openssl -y** command to update the OpenSSL version.

- **macOS**
  a. Open the MacOS startup console and select **Keychain Access**.
  b. Enter the password to log in to **Keychain Access**.
  c. Drag and drop the target root CA certificate into **Keychain Access**. The root CA certificate now is untrusted by the system.
  d. Right-click the root CA certificate to load its details.
  e. Click **Trust**, select **Always Trust** for **When using this certificate**, and click **Close**.
  f. Enter the password to make the configuration of the trusted root CA certificate take effect.
  g. View the root CA certificate on the Keychain Access window. If the certificate is trusted by the system, the root CA is successfully added to the trusted root CA store.

## Step 5: Installing a Private Certificate

Installing a Private Certificate on a Client (Using Windows as an Example)

**Step 1** Log in to the management console.

**Step 2** Click ☰ in the upper left corner of the page and choose **Security** > **Cloud Certificate Management Service**. In the navigation pane, choose **Private Certificate Management** > **Private Certificate**.

**Step 3** Locate the row containing the desired certificate. In the **Operation** column, click **Download**.

**Step 4** Select the **IIS** tab and click **Download Certificate**.

**Step 5** Decompress the downloaded certificate file package **client_iis.zip** to obtain certificate file **server.pfx** and private key password file **keystorePass.txt**.

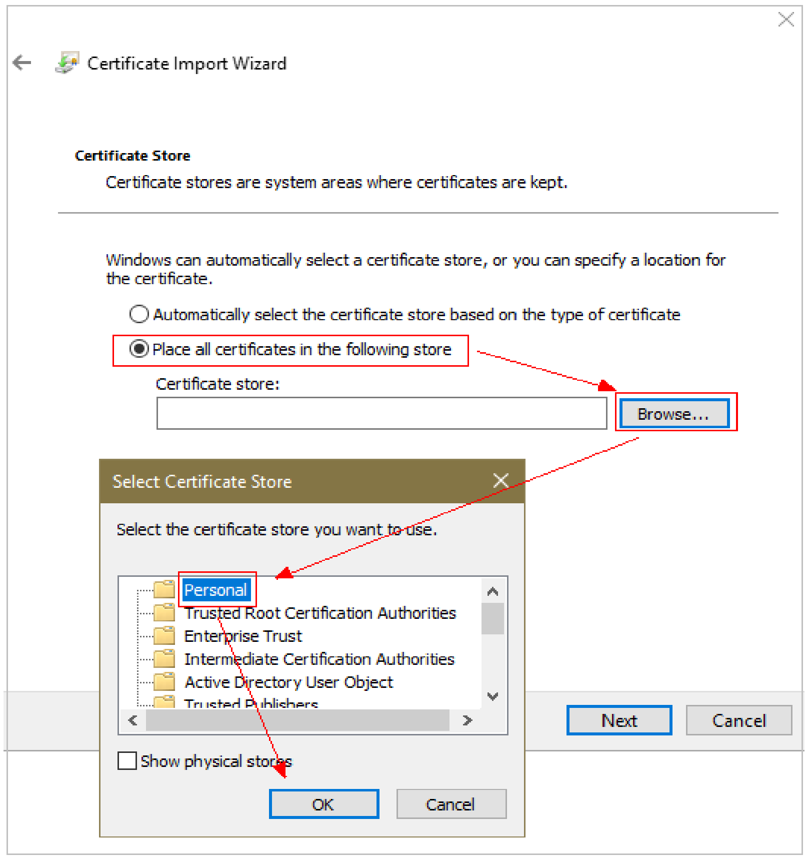
**Step 6** Double-click certificate file **server.pfx**, select a certificate storage location based on its usage, and click **Next**.

**Step 7** Confirm the name of the certificate file you want to import and click **Next**.

**Step 8** Enter the password obtained from private key password file **keystorePass.txt** and click **Next**.

**Step 9** Select **Place all certificates in the following store**, click **Browse**, select **Personal**, and click **OK**, as shown in **Figure 3-12**.

**Figure 3-12** Storing a private certificate


Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

Automatically select the certificate store based on the type of certificate [✓ Place all certificates in the following store]

Certificate store:

Select the certificate store you want to use.

Personal
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Active Directory User Object
Trusted Publishers
Show physical stores
OK
Cancel
Next
Cancel

**Step 10** Click **Next** and **Finish**. The certificate is installed when a dialog box is displayed indicating that the certificate is imported successfully.

**----End**

**Installing a Private Certificate on a Server**

The procedure for installing a private certificate on a server is the same as that for installing an international standard SSL certificate. The following lists some examples:

- **Installing an SSL Certificate on a Tomcat Server**

- **Installing an SSL Certificate on an Nginx Server**
- **Installing an SSL Certificate on an Apache Server**
- **Installing an SSL Certificate on an IIS Server**
- **Installing an SSL Certificate on a WebLogic Server**
- **Installing an SSL Certificate on a Resin Server**