

云证书管理服务

最佳实践

文档版本 09
发布日期 2024-12-17



版权所有 © 华为云计算技术有限公司 2025。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目录

1 CCM 最佳实践汇总	1
2 SSL 证书相关最佳实践	3
2.1 在华为云/腾讯云/阿里云进行 DNS 解析	3
2.2 网站启用 HTTPS 加密	7
2.3 一键部署 SSL 证书至云上	9
2.3.1 场景说明	9
2.3.2 一键部署 SSL 证书至 CDN	9
2.3.3 一键部署 SSL 证书至 WAF	11
2.3.4 一键部署 SSL 证书至 ELB	13
2.4 使用函数工作流自动获取并更新 ECS 服务器证书	16
3 私有证书相关最佳实践	26
3.1 私有证书管理最佳实践	26
3.1.1 私有证书生命周期管理	26
3.1.2 私有证书状态	27
3.1.3 私有证书轮换	27
3.2 私有 CA 管理最佳实践	28
3.2.1 私有 CA 层次结构设计	28
3.2.2 私有 CA 状态	31
3.2.3 私有 CA 生命周期管理	33
3.2.4 证书吊销列表管理	34
3.2.5 私有 CA 轮换	36
3.3 PCA 代码示例最佳实践	36
3.3.1 前提条件	36
3.3.2 私有 CA 管理代码示例	37
3.3.2.1 创建 CA	37
3.3.2.2 删除 CA	39
3.3.2.3 禁用 CA	40
3.3.2.4 启用 CA	41
3.3.2.5 导出 CA 证书	42
3.3.2.6 取消删除 CA	43
3.3.2.7 获取 CA 详情	44
3.3.2.8 查询 CA 配额	45

3.3.3 私有证书管理代码示例.....	46
3.3.3.1 申请证书.....	46
3.3.3.2 删除证书.....	48
3.3.3.3 导出证书.....	49
3.3.3.4 吊销证书.....	50
3.4 企业内网身份认证体系建立.....	51

1 CCM 最佳实践汇总

本文汇总了云证书管理服务（CCM）的常见应用场景，并为每个场景提供详细的方案描述和操作指南，以帮助您实现在不同场景下证书的部署与管理。

CCM 最佳实践

表 1-1 CCM 最佳实践

分类	相关文档
SSL证书相关	在华为云/腾讯云/阿里云进行DNS解析
	网站启用HTTPS加密
	一键部署SSL证书至云上
	使用函数工作流自动获取并更新ECS服务器证书
私有证书相关	私有证书管理最佳实践
	私有CA管理最佳实践
	PCA代码示例最佳实践
	企业内网身份认证体系建立

Solution as Code 一键式部署类最佳实践

为帮助企业高效上云，华为云Solution as Code萃取丰富上云成功实践，提供一系列基于华为云可快速部署的解决方案，帮助用户降低上云门槛。同时开放完整源码，支持个性化配置，解决方案开箱即用，所见即所得。

表 1-2 Solution as Code 一键式部署类最佳实践汇总

一键式部署方案	说明	相关服务
等保二级解决方案	帮您在华为云上快速部署等保二级合规解决方案，帮助客户快速、低成本完成安全整改，轻松满足等保二级合规要求。	CCM、WAF、HSS、MTD、CFW
等保三级解决方案	依托华为云自身安全能力与安全合规生态，为用户提供一站式的等保三级安全解决方案，轻松满足等级保护合规要求。	CCM、WAF、HSS、MTD、CFW、CBH、DBSS、CodeArts Inspector

2 SSL 证书相关最佳实践

2.1 在华为云/腾讯云/阿里云进行 DNS 解析

SSL证书提交申请后，需要进行域名验证。本文档将介绍如何完成DNS验证。

背景信息

SSL证书提交申请后，您需要进行域名授权验证。按照CA中心的规范，如果您申请了数字证书，您必须配合完成域名验证来证明您对所申请绑定的域名的所有权。当您按照要求正确配置域名验证信息，待域名授权验证完成，CA系统中心审核通过后，证书审核才可以进入下一个状态。

如果不完成域名验证，您的证书将无法通过审核，且您的证书申请将会一直显示“待完成域名验证”的状态。

DNS验证，是指在域名管理平台通过解析指定的DNS记录，来验证域名所有权的一种方式。当您申请证书时，“域名验证方式”选择的是“DNS验证”，则可参照本文档完成域名所有权的验证。

操作步骤

步骤1 获取证书的主机记录和记录值。详见[获取证书的主机记录和记录值](#)。

步骤2 DNS验证。

DNS验证即解析DNS记录，只能在域名管理平台即您的域名托管平台上进行解析，以下为您提供主流的三个域名服务商的解析方法，仅供参考，具体的请咨询您的域名服务商。

- 如果您的域名托管在华为云云解析服务，请参照[华为云DNS解析](#)完成DNS解析。
- 如果您的域名是在腾讯云申请的，即域名托管在腾讯云的解析服务，请参照[腾讯DNS解析](#)完成DNS解析。
- 如果您的域名是在阿里云申请的，即域名托管在阿里云的解析服务，请参照[阿里DNS解析](#)完成DNS解析。

步骤3 查看DNS验证是否生效。详见[验证DNS配置是否生效](#)。

----结束

获取证书的主机记录和记录值

步骤1 登录[管理控制台](#)。

步骤2 单击页面左上方的 ，选择“安全与合规 > 云证书管理服务”，进入云证书管理界面。

步骤3 在左侧导航栏选择“SSL证书管理”，并在SSL证书页面中待域名验证的证书所在行的“操作”列，单击“域名验证”，系统从右面弹出域名验证详细页面。

步骤4 在证书的域名验证页面，查看并记录“主机记录”、“记录类型”和“记录值”，如[图2-1](#)所示。

如果界面未显示，则请登录邮箱（申请证书时填写的邮箱）进行查看。

图 2-1 查看主机记录



----结束

华为云 DNS 解析

如果您是在华为云平台管理您的域名，请参考本部分进行操作。

步骤1 登录[管理控制台](#)。

步骤2 选择“网络 > 云解析服务”，进入“云解析”页面。

步骤3 在左侧树状导航栏，选择“域名解析 > 公网解析”，进入“公网域名”页面。

步骤4 在“公网域名”页面的域名列表中，单击添加的域名名称（多域名类型证书则添加主域名名称），进入该域名的记录集页面。

步骤5 在页面右上角，单击“添加记录集”，进入“添加记录集”页面，如[图2-2](#)所示。

说明

如果在“解析记录”的域名列表中，已存在域名“domain3.com”相应记录类型的记录值，直接在目标域名的“操作”列，单击“修改”，进入“修改记录集”页面。

图 2-2 添加记录集



表 2-1 添加记录集参数说明

参数名称	参数说明
主机记录	证书的“域名验证”页面，域名服务商返回的“主机记录”。
类型	证书的“域名验证”页面，域名服务商返回的“记录类型”。
别名	选择“否”。
线路类型	选择“全网默认”。
TTL (秒)	一般建议设置为5分钟。TTL值越大，则DNS记录的同步和更新越慢。
值	证书的“域名验证”页面，域名服务商返回的“记录值”。 说明 记录值必须用英文引号引用后粘贴在文本框中。
其他的设置保持不变。	

步骤6 单击“确定”，记录集添加成功。

当记录集的状态显示为“正常”时，表示记录集添加成功。

说明

- 该DNS配置记录在证书颁发或吊销后才可以删除。
- 请您务必检查是否正确配置了DNS记录，DNS没有配置正确是无法签发证书的。

步骤7 验证完成后，CA机构可能还需要一段时间审核域名信息。在此期间，证书状态为“待完成域名验证”。

CA机构审核通过后，证书审核才可以进入“待完成组织验证”状态。

----结束

腾讯 DNS 解析

如果您需要绑定华为云SSL证书的域名是在腾讯云申请的，您需要先在腾讯云DNS解析控制台中添加解析记录，完成DNS验证。

步骤1 登录腾讯云DNS解析控制台。

步骤2 在“域名解析列表”中，选择需要添加解析记录的域名，单击操作栏的“解析”，进入该域名的“记录管理”页面。

步骤3 单击“添加记录”，填写以下记录信息。

- 主机记录：[获取证书的主机记录和记录值](#)中获取的“主机记录”。
- 记录类型：[获取证书的主机记录和记录值](#)中获取的“记录类型”。
- 线路类型：选择“默认”类型，否则会导致部分用户无法解析。
- 记录值：输入该域名在华为云SSL证书服务控制台获取的DNS记录值，具体的操作步骤如[获取证书的主机记录和记录值](#)。
- MX优先级：不需要填写。
- TTL：为缓存时间，数值越小，修改记录各地生效时间越快，默认为600秒。

步骤4 单击“保存”，完成添加。

----结束

阿里 DNS 解析

如果您需要绑定华为云SSL证书的域名是在阿里云申请的，您需要先在阿里云DNS解析控制台中添加解析记录，完成DNS验证。

步骤1 登录阿里云DNS解析控制台。

步骤2 在域名解析页面，选择“全部域名”页签，单击需要添加解析记录的域名名称，进入解析设置页面。

步骤3 单击“添加记录”，填写以下记录信息。

- 记录类型：[获取证书的主机记录和记录值](#)中获取的“记录类型”。
- 主机记录：[获取证书的主机记录和记录值](#)中获取的“主机记录”。
- 解析线路：选择“默认”类型，否则会导致部分用户无法解析。
- 记录值：输入该域名在华为云SSL证书服务控制台获取的DNS记录值，具体的操作步骤如[获取证书的主机记录和记录值](#)。
- TTL：为缓存时间，数值越小，修改记录各地生效时间越快，默认为600秒。

步骤4 单击“确定”，完成添加。

----结束

验证 DNS 配置是否生效

根据不同记录类型、操作系统，选择以下命令验证DNS配置是否生效。

本文档以主机记录值为“_dnsauth.domain.com”为例。

- 记录类型为“TXT”
 - Windows系统：

```
nslookup -q=TXT _dnsauth.domain.com
```
 - Linux系统：

```
dig TXT _dnsauth.domain.com
```
 - MACOS系统：

```
dig TXT _dnsauth.domain.com
```

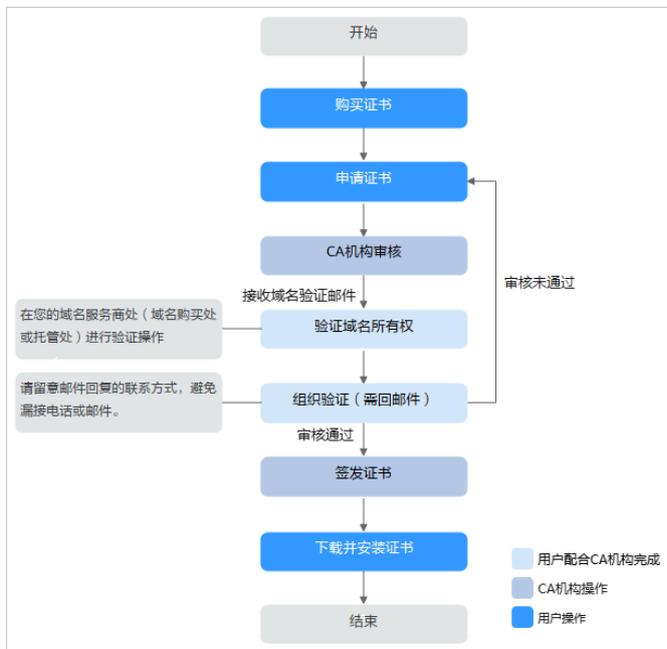
- 记录类型为“CNAME”
 - Windows系统：
nslookup -q=CNAME _dnsauth.domain.com
 - Linux系统：
dig CNAME _dnsauth.domain.com
 - MACOS系统：
dig CNAME _dnsauth.domain.com

如果界面回显的记录值（text的值）与域名服务商返回的“记录值”一致，说明域名授权验证配置已经生效。

2.2 网站启用 HTTPS 加密

SSL证书保障网站通信安全，同时也是网站可信的“身份证”，本文档介绍SSL证书从选购到安装的流程，帮助您的网站通过HTTPS加密协议来传输数据。

图 2-3 证书使用流程



操作步骤

步骤1 您需要根据您实际的业务场景需求购买相应的证书，详细操作请参见[购买SSL证书](#)。

图 2-4 如何选购证书



- 步骤2** 成功购买证书后，您需要申请证书，即为证书绑定域名、填写证书申请人的详细信息并提交给证书颁发机构审核，详细操作请参见[申请SSL证书](#)。
- 步骤3** 提交审核后，证书颁发机构将向您填写的邮箱发送一封验证邮件，您需要根据申请书时填写的域名验证方式进行域名验证，详细操作请参见[验证域名所有权](#)。
- 步骤4** 当您申请的是OV和EV类型证书时，域名验证完成后，CA机构将向您填写的邮箱发送一封组织验证邮件。CA机构将根据您选择的验证方式与企业/组织进行联系，确认企业/组织是否发起了此次的证书订单申请。详细操作请参见[组织验证](#)。
- 步骤5** 不同证书的签发周期如[表 证书审核周期](#)所示，请您耐心等待证书签发。

说明

域名验证和组织验证都需要您配合CA机构完成，若您未及时配合会影响证书的签发周期。

表 2-2 证书审核周期

证书类型	审核周期
EV	CA机构人工审核信息。 在信息正确的情况下审核周期一般为7~10个工作日。
OV	CA机构人工审核信息。 在信息正确的情况下审核周期一般为3~5个工作日。
DV	无人工审核。 CA机构签发系统自动检查域名授权配置，DNS配置正确的情况下（需要您自行排查DNS配置是否正确）可在数小时内快速颁发。

- 步骤6** 证书签发后，请将证书下载到本地，详细操作请参见[下载SSL证书](#)。
- 步骤7** 安装SSL证书，让服务器开启HTTPS加密通信，不同Web服务器安装SSL证书的具体操作不同，以下介绍了几种在主流Web服务器上安装SSL证书的方法，请根据您的需要进行选择：

- 在Tomcat上安装SSL证书的详细指导操作请参见[如何在Tomcat上安装SSL证书?](#)。
- 在Nginx上安装SSL证书的详细指导操作请参见[如何在Nginx上安装SSL证书?](#)。
- 在Apache上安装SSL证书的详细指导操作请参见[如何在Apache上安装SSL证书?](#)。
- 在IIS上安装SSL证书的详细指导操作请参见[如何在IIS上安装SSL证书?](#)。
- 在Weblogic上安装SSL证书的详细指导操作请参见[在Weblogic服务器上安装SSL证书](#)。

----结束

相关问题

- [各证书之间的区别](#)
- [申请证书时，如何填写证书中绑定的域名?](#)
- [域名验证完成后，为什么SSL证书还是停留在“待完成域名验证（申请进度为40%）”的状态?](#)
- [收到CA机构的邮件或电话如何处理?](#)

2.3 一键部署 SSL 证书至云上

2.3.1 场景说明

您可以将在云证书管理服务中已签发的SSL证书、在第三方处购买后上传到云证书管理服务托管的SSL证书一键部署至华为云产品CDN、WAF或ELB中，实现网站HTTPS化，提高云产品访问数据的安全性。

SSL 证书工作原理

SSL证书是用户在Web服务器与浏览器以及客户端之间建立加密通道，通过配置和应用SSL证书来启用HTTPS协议，来保证互联网数据传输的安全。

图 2-5 SSL 证书工作原理



2.3.2 一键部署 SSL 证书至 CDN

前提条件

- 在华为云云证书管理服务申请了SSL证书且证书状态为“已签发”，或者已将在其他平台中签发的SSL证书上传至云证书管理服务中且证书状态为“托管中”。
- 已开通CDN（Content Delivery Network，内容分发网络）服务。

约束与限制

申请证书时，如果“证书请求文件”选择的是“自己生成CSR”，那么签发的证书不支持一键部署到云产品。如需在对应云产品中使用证书，可以先将证书下载到本地，然后再到对应云产品中上传证书并进行部署。

部署至CDN加速域名，需要加速域名开启HTTPS，才能在部署证书功能中完成对该域名的部署。

添加 CDN 加速域名

将SSL证书部署至CDN之前，需要在CDN中将与SSL证书匹配的域名添加为CDN加速域名，详细操作步骤请参见[添加CDN加速域名](#)。

部署 SSL 证书至 CDN

步骤1 登录[管理控制台](#)。

步骤2 单击页面左上方的，选择“安全与合规 > 云证书管理服务”，进入云证书管理界面。

步骤3 在左侧导航栏选择“SSL证书管理 > SSL证书列表”，进入SSL证书列表页面。

步骤4 在目标证书所在行的“操作”列，单击“部署证书”，系统从右面弹出证书部署详细页面，如[图2-6](#)所示。

图 2-6 部署证书



步骤5 在部署证书页面的“部署详情”下，选择“CDN”页签。

图 2-7 选择 CDN



步骤6 选择当前证书中需要部署的域名，并单击“操作”列的“部署”或“重新部署”。

如需部署多个域名，则从域名列表中勾选所有待部署的域名，并单击列表左上角的“批量更新”。

步骤7 （可选）如果当前域名已经部署过该证书，单击“重新部署”后，界面会弹出重复部署仍会收费的提示框，确认无误后，单击“确定”。

步骤8 在弹出的确认框中，确认无误后，勾选“我已知悉本次部署将产生如上费用，部署成功后按需扣费，不支持退款”，单击“确认”。

图 2-8 部署证书提示信息



部署成功后，界面将提示部署成功，请前往部署记录查看。

----结束

到期替换证书

CA机构签发的SSL证书有效期仅1年，您重新购买或续费的新证书签发后，请参照[部署SSL证书至CDN](#)将新证书部署至CDN。

2.3.3 一键部署 SSL 证书至 WAF

前提条件

- 在华为云云证书管理服务申请了SSL证书且证书状态为“已签发”，或者已将在其他平台中签发的SSL证书上传至云证书管理服务中且证书状态为“托管中”。
- 已开通Web应用防火墙（Web Application Firewall，WAF）服务。

约束与限制

- 申请证书时，如果“证书请求文件”选择的是“自己生成CSR”，那么签发的证书不支持一键部署到云产品。如需在对应云产品中使用证书，可以先将证书下载到本地，然后再到对应云产品中上传证书并进行部署。

在 WAF 中添加防护域名

将SSL证书部署至WAF之前，需要在WAF中将SSL证书匹配的域名添加为WAF防护域名，WAF不同模式下添加防护域名的操作请参见：

- [WAF云模式添加防护域名](#)
- [WAF独享模式添加防护域名](#)

须知

添加防护域名时，“对外协议”需要勾选HTTPS协议。

部署 SSL 证书至 WAF

步骤1 登录[管理控制台](#)。

步骤2 单击页面左上方的 ，选择“安全与合规 > 云证书管理服务”，进入云证书管理界面。

步骤3 在左侧导航栏选择“SSL证书管理 > SSL证书列表”，进入SSL证书列表页面。

步骤4 在目标证书所在行的“操作”列，单击“部署证书”，系统从右面弹出证书部署详细页面，如图2-9所示。

图 2-9 部署证书



步骤5 在部署证书页面的“部署详情”下，选择“WAF”页签。

图 2-10 选择 WAF



步骤6 单击企业项目或区域名称右侧的 ，选择部署的企业项目或区域。

步骤7 选择当前证书中需要部署的域名，并单击“操作”列的“重新部署”。

如需部署多个域名，则从域名列表中勾选所有待部署的域名，并单击列表左上角的“批量更新”。

步骤8 （可选）如果当前域名已经部署过该证书，单击“重新部署”后，界面会弹出重复部署仍会收费的提示框，确认无误后，单击“确定”。

步骤9 在弹出的确认框中，确认无误后，勾选“我已知悉本次部署将产生如上费用，部署成功后按需扣费，不支持退款”，单击“确认”。

图 2-11 部署证书提示信息



部署成功后，对应域名的“部署模式”刷新为“已部署”。

----结束

到期替换证书

CA机构签发的SSL证书有效期仅1年，您重新购买或续费的新证书签发后，请参照[部署SSL证书至WAF](#)将新证书部署至WAF。

2.3.4 一键部署 SSL 证书至 ELB

前提条件

- 在华为云云证书管理服务申请了SSL证书且证书状态为“已签发”，或者已将在其他平台中签发的SSL证书上传至云证书管理服务中且证书状态为“托管中”。
- 已开通弹性负载均衡（Elastic Load Balance，ELB）服务。

约束与限制

- 通过SCM更新ELB中的证书，可以更新部署在ELB监听器下证书，即在SCM控制台更新对应ELB中证书的内容及私钥，更新成功后，ELB将自动对该证书部署的监听器实例完成证书内容及私钥的更新。

- ELB中使用的证书如果指定了多个域名，更新证书前需要注意SCM证书的域名与其是否完全匹配。如果不完全匹配，则在SCM中执行更新证书操作后，会同时将ELB中使用的证书域名更新为当前SCM中证书的域名。

示例：SCM中证书的主域名及附加域名为example01.com, example02.com, ELB中证书的域名为example01.com, example03.com, 在SCM中执行更新证书操作后，会将该ELB中证书的域名更新为example01.com, example02.com。

- 申请证书时，如果“证书请求文件”选择的是“自己生成CSR”，那么签发的证书**不支持**一键部署到云产品。如需在对应云产品中使用证书，可以先将证书下载到本地，然后再到对应云产品中上传证书并进行部署。

创建负载均衡器和监听器

在部署SSL证书之前，需要在ELB中创建负载均衡器和监听器，详细操作请参见：

- 创建负载均衡器
 - [创建共享型负载均衡器](#)
 - [创建独享型负载均衡器](#)
- [添加HTTPS监听器](#)

在 ELB 配置 SSL 证书

首次在ELB中部署SSL证书，需要在ELB中完成证书的配置，后续才可通过云证书管理服务一键将SSL证书部署至ELB，因此如果您未在ELB中配置过证书，请参见[在ELB中创建证书](#)在ELB中完成证书的配置。

须知

创建证书时，填写的域名需要与SSL证书的域名一致。

部署 SSL 证书至 ELB

步骤1 登录[管理控制台](#)。

步骤2 单击页面左上方的，选择“安全与合规 > 云证书管理服务”，进入云证书管理界面。

步骤3 在左侧导航栏选择“SSL证书管理 > SSL证书列表”，进入SSL证书列表页面。

步骤4 在目标证书所在行的“操作”列，单击“部署证书”，系统从右面弹出证书部署详细页面，如[图2-12](#)所示。

图 2-12 部署证书



步骤5 在部署证书页面的“部署详情”下，选择“ELB”页签。

图 2-13 选择 ELB



步骤6 单击区域名称右侧的 ▾，选择部署的区域。

步骤7 选择当前证书中需要部署的域名，并单击“操作”列的“更新证书”。

如需更新多个域名，则从域名列表中勾选所有待更新的域名，并单击列表左上角的“批量更新”。

步骤8（可选）如果当前域名已经部署过该证书，单击“更新证书”后，界面会弹出重复部署仍会收费的提示框，确认无误后，单击“确定”。

步骤9 在弹出的确认框中，确认无误后，勾选“我已知悉本次部署将产生如上费用，部署成功后按需扣费，不支持退款”，单击“确认”。

图 2-14 更新证书提示信息



页面出现证书更新成功提示，表示SSL证书更新至ELB服务成功。

----结束

到期替换证书

CA机构签发的SSL证书有效期仅1年，您重新购买或续费的新证书签发后，请参照[部署SSL证书至ELB](#)将新证书更新至ELB。

2.4 使用函数 workflows 自动获取并更新 ECS 服务器证书

应用场景

本文以[表 示例信息](#)所示为例介绍如何通过使用函数 workflows 自动获取并更新ECS服务器证书。对于续费签发的新证书，无需手动重新部署，即可更新ECS服务器证书。

表 2-3 示例信息

Web服务器类型	Nginx
代码编辑语言	Python 3.9

约束与限制

- 已开通弹性云服务器（Elastic Cloud Server，ECS），且在ECS中配置了SSL证书。
- SSL证书为云证书管理服务中购买且续费的证书。

步骤一：创建委托

使用函数 workflows 更新ECS服务器证书需要将SCM FullAccess、IAM ReadOnlyAccess权限授权给函数 workflows 服务。

步骤1 登录[管理控制台](#)。

步骤2 单击页面左上方的，选择“管理与监管 > 统一身份认证服务”，进入统一身份认证服务界面。

步骤3 在左侧导航栏选择“委托”，并在委托界面右上角单击“创建委托”，进入创建委托界面。

步骤4 在创建委托界面，按[表 创建云服务委托参数说明](#)所示设置委托信息，如[图 创建云服务委托](#)所示。

图 2-15 创建云服务委托

表 2-4 创建云服务委托参数说明

参数	配置说明
委托名称	自定义委托名称。
委托类型	选择“云服务”。
云服务	选择“FunctionGraph”。
持续时间	选择“永久”。
描述	可选填，自定义需要的信息。

步骤5 单击“下一步”，进入委托授权界面。

步骤6 选择并勾选需要授权函数工作流的“SCM FullAccess”、“IAM ReadOnlyAccess”权限。

图 2-16 选择权限



步骤7 单击“下一步”，设置权限的作用范围。

步骤8 单击“确定”，委托创建成功。

----结束

步骤二：使用空白模板创建函数

步骤1 登录[管理控制台](#)。

步骤2 单击页面左上方的 ，选择“计算 > 函数 workflow”，进入函数 workflow 界面。

步骤3 单击函数 workflow 界面右上方的“创建函数”，进入创建函数界面。

步骤4 按表 [创建空白事件函数参数配置](#) 所示信息创建空白函数，如 [图 2-17 创建空白事件函数](#) 所示。

图 2-17 创建空白事件函数



图 2-17 展示了创建空白事件函数的配置界面。界面包含以下配置项：

- 基本信息**
- * 函数类型**：包含“事件函数”和“HTTP函数”两个选项卡，当前选中“事件函数”。
- * 区域**：包含一个下拉菜单，用于选择部署区域。下方有提示文字：“不同区域的资源之间内网不互通。请就近选择靠近您业务的区域，可以降低网络时延、提高访问速度。”
- * 函数名称**：输入框中显示“AutoupdateSSL”。下方有提示文字：“可包含字母、数字、下划线和中划线，以大小写字母开头，以字母或数字结尾，长度不超过60个字符。”
- 委托名称**：包含一个下拉菜单，显示“FunctionGraph”，右侧有“创建委托”按钮。
- * 企业项目**：包含一个下拉菜单，显示“default”，右侧有“查看企业项目”按钮。
- 运行时**：包含一个下拉菜单，显示“Python 3.9”，右侧有“查看Python函数开发指南”按钮。

表 2-5 创建空白事件函数参数配置

参数	配置说明
函数类型	选择“事件类型”。
区域	选择需要部署代码的区域。
函数名称	自定义函数名称。

参数	配置说明
委托名称	选择 步骤一：创建委托 创建的委托名称。
企业项目	如果您已开通企业项目，选择需要添加函数的企业项目即可。 如果您未开通企业项目，将无法看到企业项目的选项。若需开通请参见 如何开通企业项目 ，无需开通企业项目请跳过此项。
运行时	选择函数编写语言，此处示例选择“Python 3.9”。

步骤5 单击“创建函数”，跳转至函数界面，创建空白函数成功。

----结束

步骤三：创建定时触发器

创建定时触发器，在固定时间间隔触发函数。

步骤1 在函数界面，选择“设置 > 触发器”，进入触发器页签。

步骤2 单击“创建触发器”，按表 [配置定时触发器](#) 所示信息创建定时触发器，如图 [创建定时触发器](#) 所示。

图 2-18 创建定时触发器

创建触发器

触发器类型

DDS、GAUSSMONGO、DIS、LTS、Kafka、TIMER触发器可创建数加起来最多10个，您已创建0个。

* 定时器名称

支持字母、数字、下划线和中划线，必须以字母开头，且长度不能超过64个字符

* 触发规则 固定频率 Cron表达式

单位为秒和分钟时，输入值不能超过60；单位为小时时，输入值不能超过24；单位为天时，输入值不能超过30。

* 是否开启

附加信息

0/2,048

表 2-6 配置定时触发器

参数	配置说明
触发器类型	选择“定时触发器（TIMER）”。
定时器名称	自定义定时器名称。
触发规则	设置为“固定频率”，具体频率请根据您的实际情况配置。
是否开启	<input checked="" type="checkbox"/>
附加信息	可选填，自定义需要的信息。

步骤3 单击“确定”，定时触发器创建成功。

----结束

步骤四：制作并配置函数依赖包

部署证书至ECS的函数代码需要依赖paramiko依赖包，您需要为函数制作并配置paramiko依赖包。

本小节以Python 3.9为例介绍制作和配置依赖包的方法。其他代码编辑语言制作依赖包的方法请参见[如何制作依赖包](#)。

为Python制作依赖包

步骤1 打包环境中的Python版本要和对应函数的运行时版本相同。

如Python 3.9建议使用3.9.0及以上版本，Python2.7建议使用2.7.12及以上版本，Python3.6建议使用3.6.3以上版本。

步骤2 执行如下命令，为Python 3.9安装paramiko依赖包，并指定此依赖包的安装路径为本地的/tmp/paramiko下。

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

步骤3 执行如下命令切换到/tmp/paramiko下。

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

步骤4 进入子目录直到site-packages路径下（一般路径为usr/lib64/python3.9/site-packages/），并执行如下命令。

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

所生成的包即为最终需要的依赖包。

📖 说明

如果需要安装存放在本地的wheel安装包，直接执行如下命令：

```
pip install piexif-1.1.0b0-py2.py3-none-any.whl --root /tmp/piexif  
//安装包名称以piexif-1.1.0b0-py2.py3-none-any.whl为例，请以实际安装包名称为准
```

配置依赖包

步骤5 登录[管理控制台](#)。

步骤6 单击页面左上方的 ，选择“计算 > 函数工作流”，进入函数工作流界面。

步骤7 在左侧导航栏选择“函数 > 依赖包管理”，进入依赖包管理界面。

步骤8 单击的“创建依赖包”，弹出“创建依赖包”对话框，按[表 配置依赖包](#)所示设置依赖包信息。

表 2-7 配置依赖包

参数	说明
依赖包名称	自定义的依赖包名称，用于识别不同的依赖包。
代码上传方式	选择“上传ZIP文件”。
文件上传	添加依赖包ZIP文件。
运行时语言	选择函数编写语言，此处示例选择“Python 3.9”。
描述	对于依赖包的描述信息，可以不填。

步骤9 单击“确定”，完成依赖包创建。

步骤10 在左侧导航栏选择“函数 > 函数列表”，进入函数列表界面。

步骤11 单击函数名称，进入函数详情界面。

步骤12 在代码页签，单击“代码依赖包”所在行的“添加依赖包”，弹出“选择依赖包”对话框。

步骤13 选择8创建的私有依赖包，单击“确定”，函数依赖包配置完成。

----结束

步骤五：在函数中配置代码源

在函数中配置代码源，本节以在线编辑的方式为例。更多创建代码源的方式请参见[创建程序包](#)。

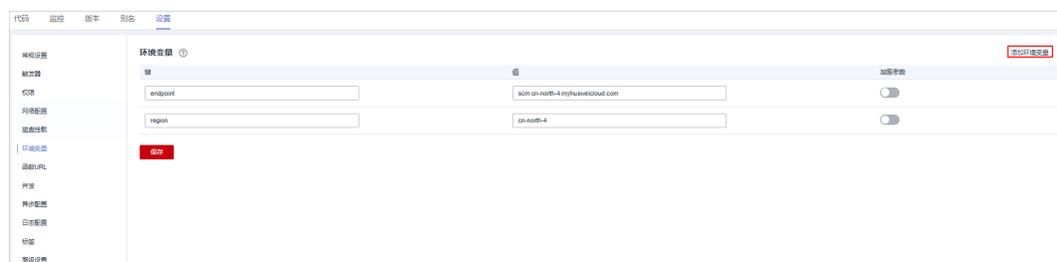
步骤1 在函数 workflow 界面左侧导航栏选择“函数 > 函数列表”，进入函数列表界面。

步骤2 单击函数名称，进入函数详情界面。

步骤3 选择“设置 > 环境变量”，进入环境变量页签，

步骤4 单击“添加环境变量”，如图 [设置环境变量](#) 所示，添加“endpoint”、“region”两个环境变量。

图 2-19 设置环境变量



环境变量1:

- 键: endpoint
- 值: scm.cn-north-4.myhuaweicloud.com

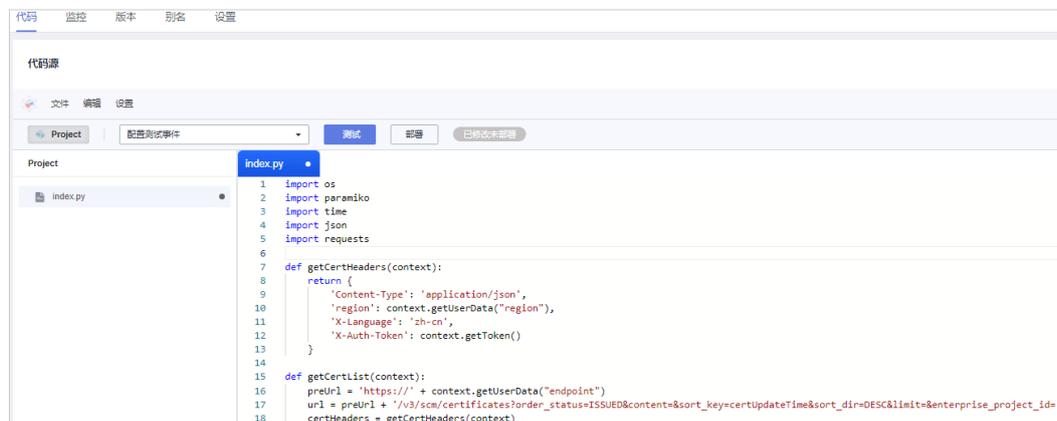
环境变量2:

- 键: region
- 值: cn-north-4

步骤5 单击“保存”，选择代码页签。

步骤6 在代码页签，如图 [添加代码](#) 所示，将以下两段代码整合添加到一个代码源文件。

图 2-20 添加代码



获取当前账号下SSL证书的应用程序代码示例如下:

```
import json
import requests
import datetime
import time

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

def isValidCert(cert):
    # TODO 用户可根据业务场景自定义 以下仅示例
    certDomain = cert.get('domain')

    # 判断是否是对应域名的续费证书
    if (certDomain != 'XXXX'):
        return False

    # 以下实例筛选出签发时间为昨天并且域名符合要求的证书
    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')

    # 获取证书签发时间
    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 需基于业务背景结合函数调用 以下仅示例
    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)
```

部署SSL证书到弹性服务器（Nginx）的应用程序代码示例如下：

```
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)

# 前提证书位置已经写入nginx.conf文件
cmd="/usr/local/nginx/sbin/nginx -s reload"
stdin,stdout,stderr = ssh.exec_command(cmd)
ssh.close()
```

说明

- 以上两段代码为示例，请根据您的实际情况修改后使用。代码中的“XXXX”表示域名，请改为您的实际域名。
- 以上代码中涉及的如下函数需要根据您的业务背景编辑代码且在做代码源合并时需要放在所有代码结尾部分。

```
def handler(event, context):  
    # TODO 需基于业务背景结合函数调用
```

步骤7 单击“测试”，测试函数，确认函数能正常执行。

测试函数的详细操作请参见[在线调试](#)。

步骤8 代码源添加并测试完成后，函数会根据定时触发器设置的触发规则运行，如有续费证书签发会被自动获取并更新至ECS。

步骤9 您可以在函数详情页选择“监控 > 指标”，进入监控指标页签，查看函数运行情况。

可以查看到“调用次数”、“运行时间”、“错误次数”和“被拒绝次数”等指标。有关监控更详细的说明请参见[函数监控](#)。

----结束

3 私有证书相关最佳实践

3.1 私有证书管理最佳实践

3.1.1 私有证书生命周期管理

私有证书生命周期管理操作说明如[表 私有证书生命周期管理操作说明](#)所示。

表 3-1 私有证书生命周期管理操作说明

操作	说明	备注
申请私有证书	申请私有证书，一般可根据实体在通信中扮演的角色，将私有证书分为客户端证书和服务器证书。申请私有证书前，用户需要有已创建好的可用于签发证书的私有CA。	<ul style="list-style-type: none">私有证书按个数收费，一经签发，不支持退费。私有证书的通用名称（common name）允许重复，建议您为私有证书取具有标识性的名称，以便区别。
导出私有证书	导出申请好的私有证书（包含私钥），用户可根据自己的需要，选择需要导出的格式。	私有证书的私钥需要妥善保管，若不慎泄露，请及时吊销和替换私有证书。 须知 若在证书链路径上，任何一个CA证书被永久删除了，则无法导出证书。
吊销私有证书	当私有证书因为某些因素，不再使用时，可将其吊销。及时吊销，可避免私有证书滥用。	证书滥用，将会引发安全问题，请给予重视。 须知 若父CA未启用证书吊销列表，则无法查询到私有证书是否已被吊销，意味着校验私有证书时，私有证书仍可通过吊销验证。

操作	说明	备注
删除私有证书	用户可根据自身需要对私有证书执行删除操作。	任何状态下的私有证书，皆可删除。 须知 此操作将会在数据库中，立即删除私有证书所有信息，属于不可逆操作，请谨慎执行。

3.1.2 私有证书状态

私有证书状态说明如表 [私有证书状态说明](#) 所示。

表 3-2 私有证书状态说明

私有证书状态	可导出证书	占用证书配额
已签发	是	是
已吊销	否	是
已过期	否	是

3.1.3 私有证书轮换

私有证书被部署在服务节点上（包含私钥），频繁用于加密通信，为了避免私钥的泄露，通常根据业务场景的安全级别要求，来设定私有证书的有效期以及私有证书轮换（即新私有证书替换旧私有证书）的周期。例如，当私有证书被用于高机密的加密会议场景时，私有证书的有效期可能是小时级别的，而当私有证书被用于部署在web服务器时，通常有效期为年级别（当前国际证书颁发机构签发的SSL证书，有效期基本都为1年）。

私有证书的轮换周期是根据私有证书的过期时间来设定的，基本原则是在旧私有证书过期前，将新私有证书替换到对应的工作节点上，避免因私有证书过期导致业务通信中断。

注意

- 私有证书即将过期时，请预留足够的缓冲时间，来确保私有证书被成功轮换，避免当出现不可控因素导致轮换失败时（这时需要一定的时间进行重试或者手动替换），旧私有证书过期导致业务中断。
- 若被成功替换下来的旧私有证书，还有较长的有效期时，将其吊销可避免被滥用。
- 若新私有证书的根CA与旧私有证书的根CA不一致，则需要将新私有证书信任的根CA加到根CA信任列表中。

3.2 私有 CA 管理最佳实践

3.2.1 私有 CA 层次结构设计

PCA服务中支持创建最多七级的CA层级结构，根CA最多可以有六级从属CA（即子CA或中间CA），但可以有任意数量的分支。一个好的CA层次结构设计，具有以下优势：

- 让整个PKI（公钥基础设施）体系的管理更加合理、安全。
- 可实现对证书的精细化控制。
- 可使PKI体系与自身的业务结构更加贴合，方便后续的迁移与扩展。

PCA服务中您可创建的每种结构的详细说明如表 [CA层次结构说明](#) 所示，您可以根据实际情况设计对应的CA层次结构。

表 3-3 CA 层次结构说明

CA层次结构	描述	说明
单层CA结构	通过根CA直接签发私有证书。	此种结构是不符合安全规范的，往往被用于非生产环境（不需要完整信任链的开发和测试）。 根CA将被频繁使用，密钥材料的泄露风险极高，一旦出现根CA密钥材料泄露的情况，其下所有证书都需要废弃，且所有终端都必须快速从信任的根证书列表中撤掉泄露的根CA，耗时耗力，且严重阻断业务。单层CA结构如 图 单层CA结构 所示。
两层CA机构	根CA用于签发二级从属CA，二级CA(路径深度设置为0)负责签发私有证书。	此种结构为比较常用的CA层次结构。 CA层次浅，证书链在传输和校验过程中可节省开销，且在根CA与私有证书之间做了隔绝，使用从属CA来签发证书，若单个从属CA泄露，只需将其吊销和替换其下所有的证书即可，不影响其他从属CA的使用，终端也无需撤掉根CA，缩小了泄露事件的影响范围。两层CA结构如 图 两层CA结构 所示。
三层CA结构	根CA用于签发二级从属CA，二级CA(路径深度设置为1)再向下签发三级从属CA，三级CA（路径长度设置为0）负责签发私有证书。	此种结构也是比较常用的CA层次结构，适合层次结构相对复杂的组织。 三层结构使得证书的分发管理得到精细化控制，PKI体系层次分明，且层次的适当扩展可有效地扩大根CA与私有证书的距离，能更好的保护根CA密钥材料的机密性。三层CA结构如 图 三层CA结构 所示。

CA层次结构	描述	说明
四层到七层的CA结构	根CA用于签发二级从属CA，二级CA（路径深度可设置范围为： $5 \geq \text{路径深度} \geq 2$ ）再向下签发三级从属CA，三级CA（路径深度可设置范围为： $4 \geq \text{路径深度} \geq 1$ ）再向下签发四级从属CA（路径深度可设置范围为： $3 \geq \text{路径深度} \geq 0$ ），以此类推，最后一级负责签发私有证书。	此类结构使用较少。 虽然深层次的结构能使得CA层次更加的细化，但由于层次的加深，导致服务端在与客户端交互时，需要传输较大的证书链，增加了网络传输开销和证书的校验时长。 四层到的CA结构如 图 四层到七层的CA结构 所示。

说明

从属CA的路径深度，即当前CA可以签发下级CA的层次数量，用于控制证书链深度。

图 3-1 单层 CA 结构

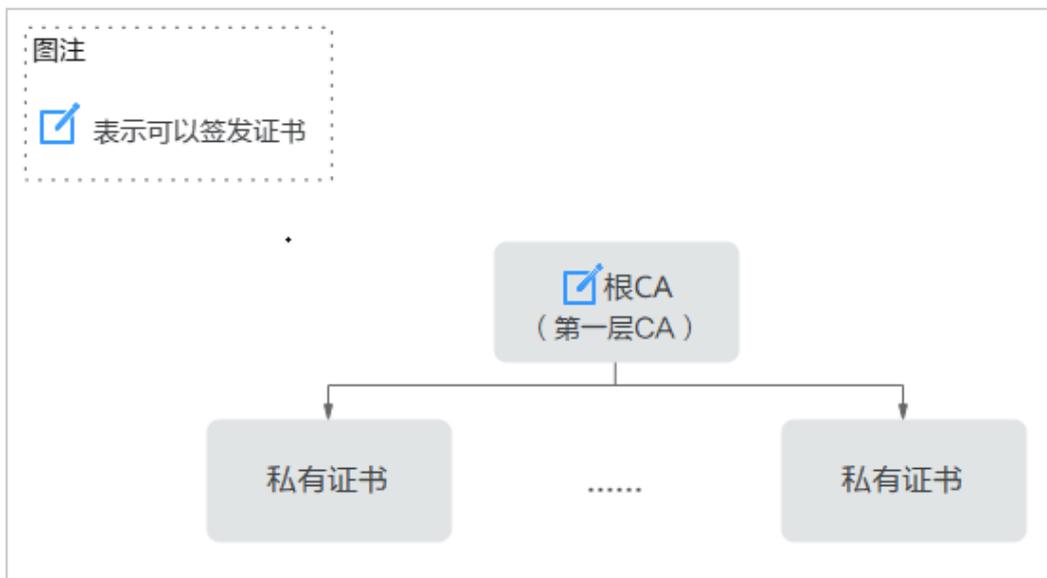


图 3-2 两层 CA 结构

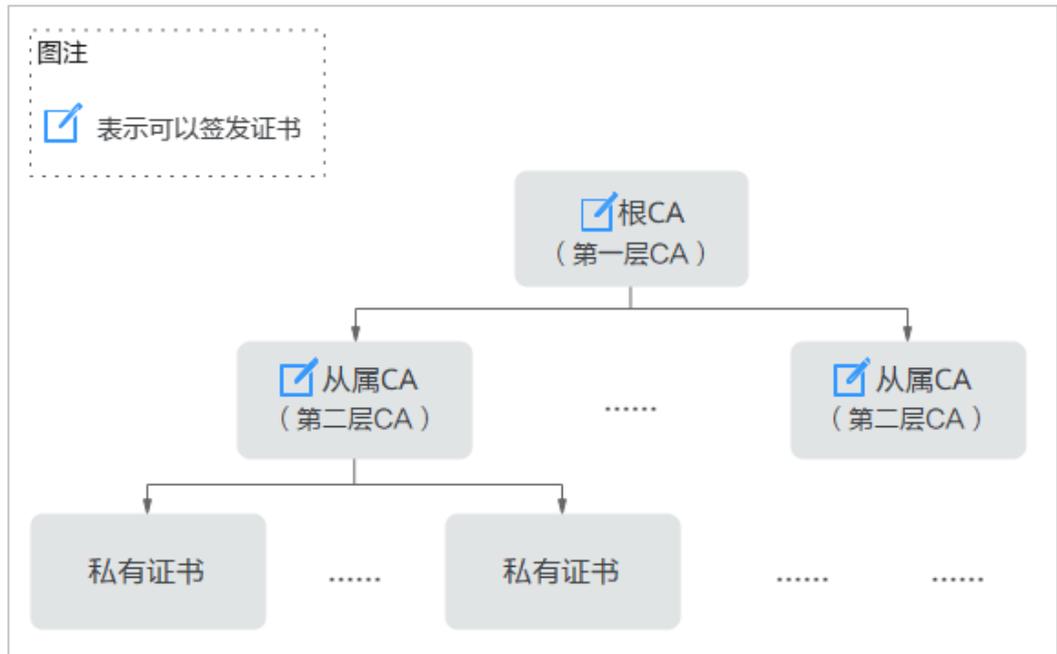


图 3-3 三层 CA 结构

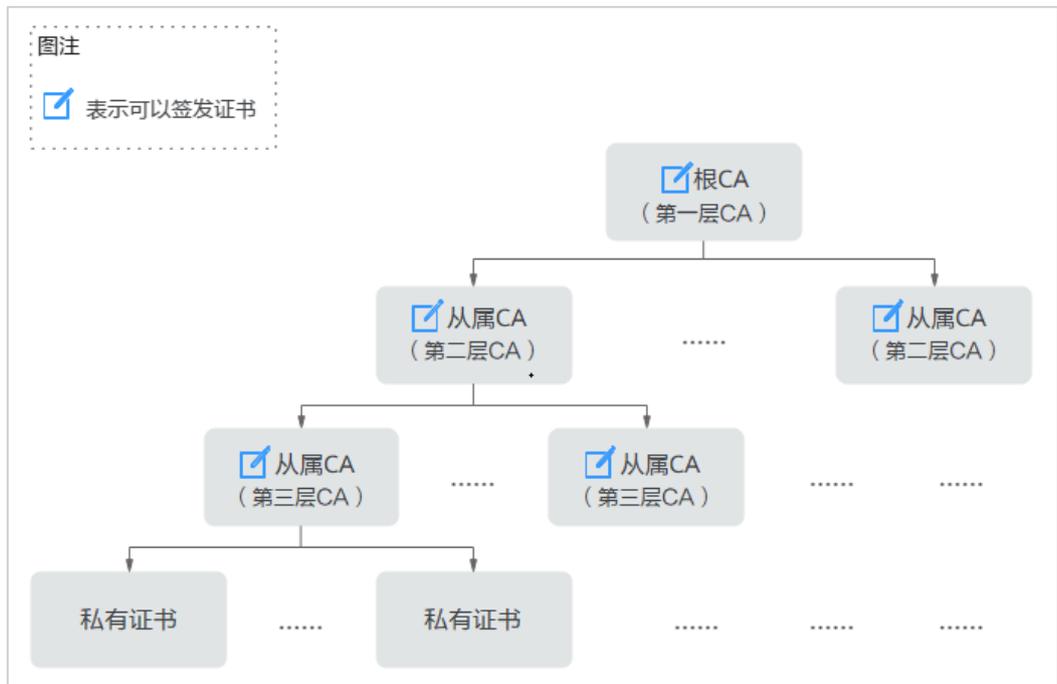
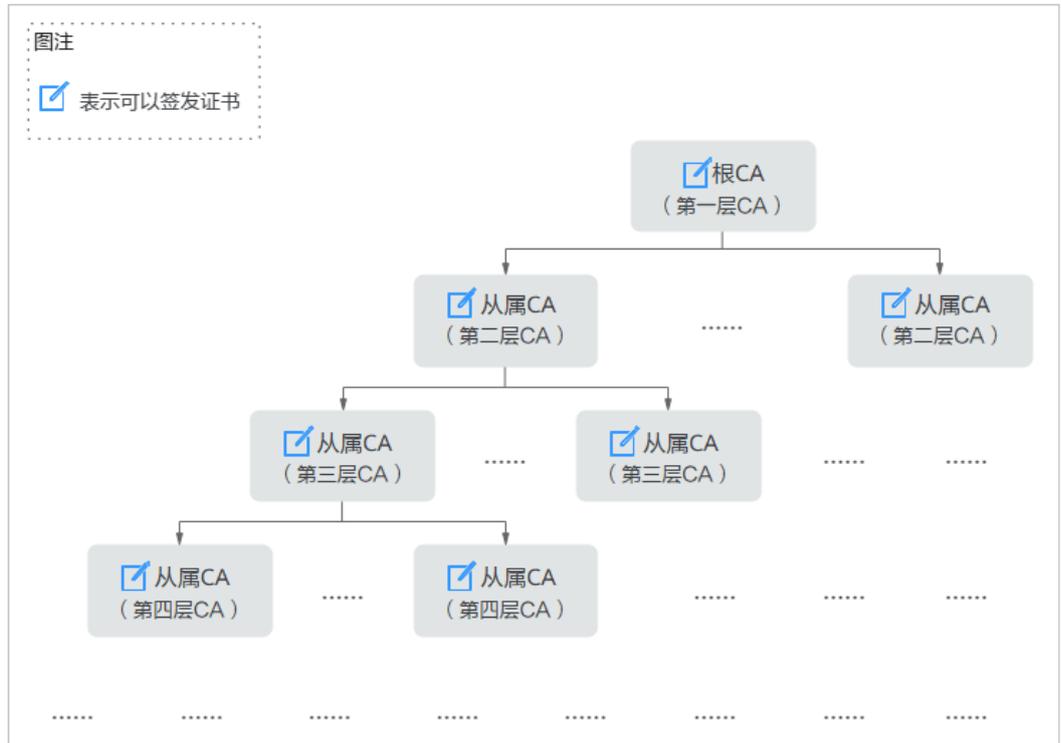


图 3-4 四层到七层的 CA 结构



3.2.2 私有 CA 状态

私有CA状态是PCA服务为了方便对私有CA进行生命周期管理，对私有CA在不同时期的状态作的描述，各个私有CA状态之间的关系如图 [私有CA状态关系](#) 所示。不同私有CA状态下，私有CA也具备不同的功能属性，如表 [私有CA状态的功能属性](#) 所示。

图 3-5 私有 CA 状态关系

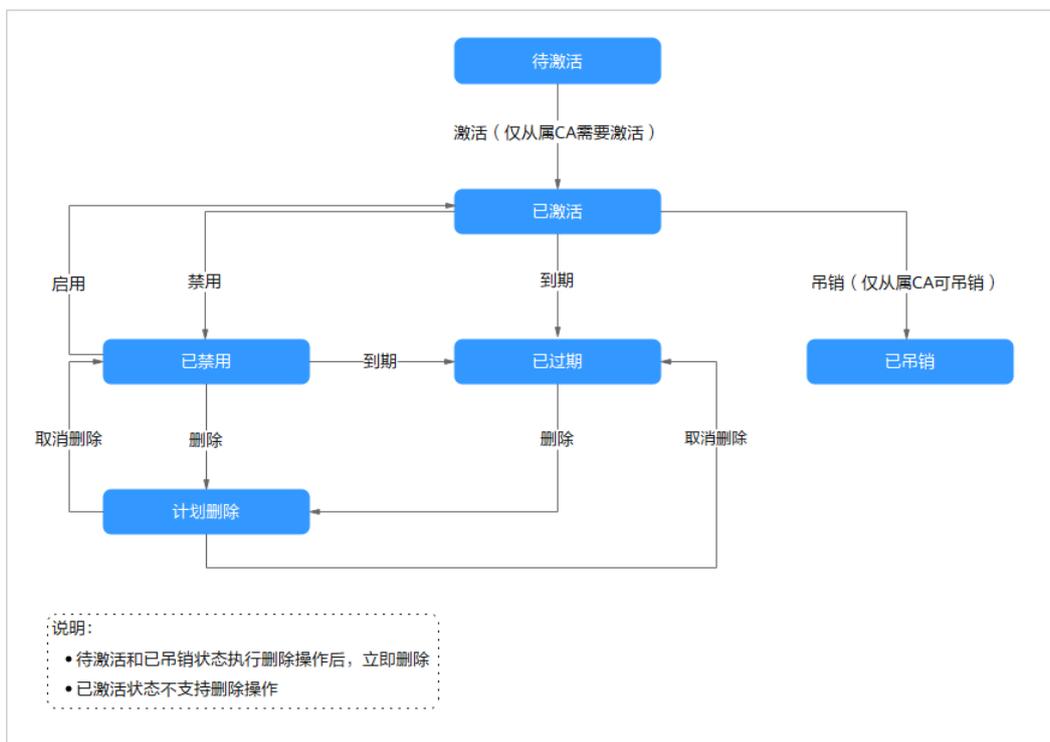


表 3-4 私有 CA 状态的功能属性

私有CA状态	签发证书	吊销证书	导出证书	导入证书	导出CSR	占用CA配额	是否收费
待激活	否	否	否	是	是	是	否
已激活	是	是	是	否	否	是	是
已禁用	否	是	是	否	否	是	是
计划删除	否	否	否	否	否	是	否
已过期	否	否	是	否	否	是	是
已吊销	否	否	否	否	否	是	否

注意

- 只有从属CA会处于“待激活”与“已吊销”状态。“待激活”状态下不收费，根CA是自签名证书，不存在“已吊销”状态的情况。
- 如果您对私有CA进行取消删除时，将会补收私有CA处于“计划删除”期间的费用。
- 处于“已过期”状态的私有CA，仍然占用CA配额，未删除前将会持续收费，若您不再使用，请尽快删除，避免被收费。

3.2.3 私有 CA 生命周期管理

创建 CA

私有CA分为根CA与从属CA，您可以指定将要创建的私有CA类型。其中，根CA是自签名证书，可直接创建。从属CA属于子CA，创建前需要先创建其父CA。因此，创建私有CA时，有以下三种情况：

- 创建根CA，创建成功后，证书为“已激活”状态。根CA的密钥用途固定为数字签名、签发证书、签发CRL，即可用于签发证书、吊销证书以及签发证书吊销列表，无法进行自定义。
- 创建从属CA，并直接激活，创建成功后，从属CA为“已激活”状态。其密钥用途默认与根CA一致，若有特殊要求，您也可以自定义从属CA密钥用途。
- 创建从属CA，但不立即激活，创建成功后，从属CA为“待激活”状态。该状态下从属CA无任何功能，只有激活后方可正常使用，该状态支持立即删除。

📖 说明

私有CA的通用名称允许重复，推荐您为不同的CA取带有标识性的名称，以便区别，如ROOT CA G0、ROOT CA G1。

激活 CA

将“待激活”状态的从属CA激活，使其可正常使用。从属CA一旦激活，将开始计费，且无法回到“待激活”状态。

禁用 CA

禁用私有CA签发证书的功能，但保留其吊销证书和签发证书吊销列表的功能。只有“已激活”状态的私有CA支持“禁用”操作，禁用后，私有CA状态为“已禁用”。

通常，在旧CA即将过期时，会将其禁用，从而保证新证书将不再通过其签发，而是由新CA来签发。保留吊销证书的功能，保证新、旧证书完成替换前，旧证书仍然可正常工作。

启用 CA

将处于“已禁用”状态的私有CA启用，恢复其签发证书的功能。启用“已禁用”的私有CA后，私有CA状态转为“已激活”。

删除 CA

对私有CA执行删除操作。鉴于私有CA的重要性，为避免误操作，PCA服务对不同状态下的私有CA，被执行删除操作后，处理的策略不同。

- “已禁用”和“已过期”状态：仅提供计划删除，可选的延迟删除周期为7~30天，在删除时间到之前，CA处于“计划删除”状态。处于“计划删除”状态的CA，可通过“取消删除”操作，将CA恢复到“已禁用”或“已过期”状态（删除前的状态）。一旦到了删除时间，CA将被**定时任务进行删除，且不可恢复**。
- “待激活”和“已吊销”状态：仅提供立即删除，即一旦执行删除操作，该状态下的CA将被立即删除，且不可恢复。
- “已激活”状态：不支持删除操作，若需要删除，需要先将其禁用，然后再执行删除操作。

⚠ 注意

私有CA被永久删除后，其下所有证书将无法执行“吊销”操作，其与其子CA下所有私有证书将无法执行“导出”操作，且无法再更新证书吊销列表，请谨慎操作！

- 执行删除操作前，请排查和确认私有CA是否仍在使用，删除后是否会导致自建的PKI体系不可用。
- 执行删除操作前，若私有CA确认已不再使用，应将其下所有未过期证书都进行吊销，并在所有终端中将其从信任列表中移除（若是从属CA，则应该将其吊销，然后再删除）。

取消删除 CA

将处于“计划删除”状态的私有CA，恢复为其执行删除之前状态。

⚠ 注意

执行“取消删除”操作后，PCA服务将会补收私有CA处于“计划删除”状态期间的费用，请谨慎操作！

吊销 CA

吊销不再使用或者密钥材料已暴露的从属CA。吊销后的从属CA，将完全失去所有功能，且无法再恢复。若其父CA启用了证书吊销列表配置，则可在证书吊销列表中查询其吊销信息。

⚠ 注意

- 吊销CA属于高危行为，请谨慎操作！
- 校验过程中需要查询证书吊销列表，方可验证正在校验的证书是否被吊销，否则，将可能与被吊销的证书进行通信，存在安全风险。
- 私有CA被吊销后，其与其子CA签发的所有证书都将不再被信任（被发布至证书吊销列表中时），即包含私有CA的所有证书链的校验都将会失败。

CA 过期处理

当私有CA过了有效期，后台定时任务会将私有CA状态置为“已过期”。

3.2.4 证书吊销列表管理

PCA服务中，对证书吊销列表（Certificate Revocation List，CRL）的管理有以下约束：

- 仅当创建私有CA时，启用了证书吊销列表配置，吊销证书后才会发布吊销列表。

须知

当父CA未启用CRL时，被吊销的证书不会被写进吊销列表中，意味着校证书时，证书仍可通过吊销验证，即存在安全隐患。有吊销证书需求的用户，请务必启用CRL配置。

- 当前只允许将CRL发布至用户授权的OBS桶中，不允许自定义其它地址。
- 启用CRL配置后，发布的CRL文件的访问策略与用户的OBS策略有关，用户可至对象存储服务（Object Storage Service, OBS），对已授权的OBS桶设置自定义访问策略。
- 证书一旦被吊销，将不可再恢复。
- 处于吊销状态的证书，将不被信任（被发布至CRL中）。
- 当证书被吊销后，PCA服务会在30分钟内将其写进CRL（当其父CA启用CRL时），并更新进OBS桶中。若发布CRL失败，会间隔15分钟后，再次尝试生成。
- 当发布新CRL的定时任务开启时，若私有CA已被删除或已过期、OBS桶被删除或授权被取消，定时任务将执行失败。
- 在CRL的有效期内，若私有CA未吊销过子证书，则只有过了有效期后（可能会延迟30分钟左右），才会生成新的CRL，有效期支持7~30天。
- 适合的吊销原因，可使证书吊销列表传达的吊销信息更准确。

PCA服务中默认的吊销原因为“UNSPECIFIED”，吊销原因可选值及其含义如[表 3-5 吊销理由及含义](#)所示。

表 3-5 吊销原因及含义

吊销理由	对应RFC 5280标准中的吊销理由码	含义
UNSPECIFIED	0	吊销时未指定吊销原因，为默认值
KEY_COMPROMISE	1	证书密钥材料泄露
CERTIFICATE_AUTHORITY_COMPROMISE	2	签发路径上，存在CA密钥材料泄露
AFFILIATION_CHANGED	3	证书中的主体或其他信息已经被改变
SUPERSEDED	4	证书已被取代
CESSATION_OF_OPERATION	5	证书或签发路径中的实体已停止运营
CERTIFICATE_HOLD	6	证书当前不应被视为有效，将来可能会生效
PRIVILEGE_WITHDRAWN	9	证书不再有权声明其列出的属性
ATTRIBUTE_AUTHORITY_COMPROMISE	10	担保证书属性的机构可能已受到损害

📖 说明

PCA服务中关于吊销理由的命名与国际标准存在差异，您可以通过吊销理由码查询[RFC 5280标准](#)中对吊销理由的相关描述。

3.2.5 私有 CA 轮换

- 私有CA轮转是指使用新的CA替换即将过期的CA的过程。
- 私有CA的管理者需要设置合适的私有CA有效期。
有效期过长，将增加密钥材料泄露的风险；有效期过短，将频繁进行私有CA轮换，增大业务开销。
- 私有CA是会过期的，为了保证业务的平滑切换，需要提前规划好私有CA的轮换方案。

操作步骤

步骤1 创建新的替代CA，同时禁用旧CA，不再使用旧CA签发证书。通过新CA签发新证书，用新签发的证书替换旧CA签发出的各类证书，并部署到对应的业务节点上。

须知

- 当旧CA替换完成前，业务系统应当同时信任新、旧CA。
- 当替换的是从属CA时，只要新签发的CA与旧CA信任的根CA是同一个，则业务节点无需做任何操作，即可同时信任新、旧CA。
- 当替换的是根CA时，CA替换开始前，需要将新的根CA预置到业务节点的受信任根证书列表中，才可保证新签发的证书被信任。

步骤2 当新证书的替换工作完成后，将旧证书全部吊销和删除，包括旧CA。

---结束

📖 说明

- 规划合理的定期私有CA轮换方案，可保障证书得到不断的更新，防范私钥被攻破；规划突发情况下的应急私有CA轮换方案，可避免业务的损失，如证书私钥暴露、签发机构不再可信等各类突发情况造成的损失。
- 新CA在主体名称上，应适当加些可识别的版本标记，如ROOT CA G0---->ROOT CA G1，以方便在私有CA轮换期间，对新旧CA的快速识别。

3.3 PCA 代码示例最佳实践

3.3.1 前提条件

请准备基础认证信息：

- ACCESS_KEY：华为云账号Access Key
- SECRET_ACCESS_KEY：华为云账号Secret Access Key
- DOMAIN_ID：华为云账号ID，详情请参见[华为云账号基本概念](#)

- CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端节点, 详情请参见[终端节点说明](#)

📖 说明

- 华为云SDK, 提供统一的SDK使用方式。通过添加依赖或下载的方式调用华为云API, 访问华为云应用、资源和数据。
- PCA服务统一SDK地址见[SDK中心](#)。
- 华为云统一SDK使用指导请参见[华为云开发者Java软件开发工具包 \(Java SDK\)](#)。

3.3.2 私有 CA 管理代码示例

3.3.2.1 创建 CA

每个用户可以创建1000个CA。

创建私有CA相关参数详情请参见[创建CA参数说明](#)。

```
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;

/**
 * 创建CA
 */
public class CreateCertificateAuthorityExample {
    /**
     * 基础认证信息:
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储在较大安全风险, 建议在配置文件或环境变量中密文存放, 使用时解密, 确保安全;
     * 本示例ak和sk保存在环境变量中为例, 运行本示例前请先在本地环境中设置环境变量
     HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息, PCA为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2.初始化SDK, 传入认证信息及CCM服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
            .withCredential(auth)
            .withEndpoint(CCM_ENDPOINT).build();

        // 3. 构造请求参数
        // (1) 需要创建的CA证书类型:ROOT (根CA)、SUBORDINATE (从属CA)
        String CAType = "ROOT";
        // (2) CA密钥算法
        String keyAlgorithm = "RSA2048";
```

```
// (3) 签名哈希算法
String signatureAlgorithm = "SHA512";

/*
 * (4) CA有效期定义
 * - type: 时间类型, 可选: "YEAR"、"MONTH"、"DAY"、"HOUR"
 * - value: 对应的值
 */
Validity validity = new Validity();
validity.setType("YEAR");
validity.setValue(20);

/*
 * (5) 定义CA的唯一标识信息
 * - organization: 组织名称
 * - organizationalUnit: 部门名称
 * - country: 国家缩写, 仅限两个字符, 如中国-CN
 * - state: 省市名称
 * - locality: 城市名称
 * - commonName: CA名称 (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) 吊销列表配置信息
 * - enabled: 是否启用CRL配置
 * - obsBucketName: OBS桶名称, 用于发布CRL, 需要已授权!!!
 * - crlName: 证书吊销列表文件名, 不传入时默认取CA ID作为文件名
 * - validDays: 证书吊销列表更新周期
 */
CrlConfiguration crlConfiguration = new CrlConfiguration();
crlConfiguration.setEnabled(false);
crlConfiguration.setObsBucketName("your OBS buck name");
crlConfiguration.setCrlName("your CRL file name");
crlConfiguration.setValidDays(7);

// (7) 请求体各属性赋值
CreateCertificateAuthorityRequestBody requestBody = new CreateCertificateAuthorityRequestBody();
requestBody.setType(CAType);
requestBody.setKeyAlgorithm(keyAlgorithm);
requestBody.setSignatureAlgorithm(signatureAlgorithm);
requestBody.setValidity(validity);
requestBody.setDistinguishedName(subjectInfo);
requestBody.setCrlConfiguration(crlConfiguration);

// 4、构造请求体
CreateCertificateAuthorityRequest request = new
CreateCertificateAuthorityRequest().withBody(requestBody);

// 5、开始发起请求
CreateCertificateAuthorityResponse response;
try {
    response = ccmClient.createCertificateAuthority(request);
} catch (Exception e) {
    throw new RuntimeException(e.getMessage());
}

// 6、获取创建成功的CA的ID
String cald = response.getCald();

System.out.println(cald);
}
```

```
}
```

3.3.2.2 删除 CA

只有“待激活”、“已吊销”、“已禁用”和“已过期”状态的私有CA可进行删除操作。

“待激活”与“已吊销”状态的私有CA将会被直接删除，“已禁用”和“已过期”状态的私有CA只能设置计划删除（延迟删除）。

相关参数详情请参见[删除CA参数说明](#)。

```
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;

/**
 * 删除CA
 * (1) 只有待激活、已吊销、已禁用和已过期状态下的CA可进行删除操作
 * (2) 待激活与已吊销状态的私有CA将会被直接删除，已禁用和已过期状态下只能设置计划删除（延迟删除）
 */
public class DeleteCertificateAuthorityExample {
    /**
     * 基础认证信息：
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储存在较大安全风险，建议在配置文件或环境变量中密文存放，使用时解密，确保安全；
     * 本示例ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量
     HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息，PCA为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2.初始化SDK，传入认证信息及CCM服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
            .withCredential(auth)
            .withEndpoint(CCM_ENDPOINT).build();

        // 3、构造请求参数
        // (1) 需要删除的CA证书的ID
        String cald = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";
        // (2) 需要延迟删除的时间。注：该参数为String
        String pendingDays = "7";

        // 4、构造请求体
        DeleteCertificateAuthorityRequest request = new DeleteCertificateAuthorityRequest()
            .withCald(cald)
            .withPendingDays(pendingDays);

        // 5、开始发起请求
        DeleteCertificateAuthorityResponse response;
        try {
            response = ccmClient.deleteCertificateAuthority(request);
        } catch (Exception e) {
```

```
        throw new RuntimeException(e.getMessage());
    }

    // 6、获取响应消息,删除成功后,无响应内容,返回的状态码为204
    System.out.println(response.getHttpStatusCode());
}
}
```

3.3.2.3 禁用 CA

禁用私有CA, 禁用状态下不可用于签发证书, 但可吊销证书和签发吊销列表。

相关参数详情请参见[禁用CA参数说明](#)。

```
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;

/**
 * 禁用CA, 禁用状态下不可用于签发证书, 但可吊销证书和签发吊销列表
 */
public class DisableCertificateAuthorityExample {
    /**
     * 基础认证信息:
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储在较大安全风险, 建议在配置文件或环境变量中密文存放, 使用时解密, 确保安全;
     * 本示例ak和sk保存在环境变量中为例, 运行本示例前请先在本地环境中设置环境变量
     HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息, PCA为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2.初始化SDK, 传入认证信息及CCM服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
            .withCredential(auth)
            .withEndpoint(CCM_ENDPOINT).build();

        // 3、构造请求参数
        // 需要禁用的CA证书ID
        String cald = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";

        // 4、构造请求体
        DisableCertificateAuthorityRequest request = new DisableCertificateAuthorityRequest()
            .withCald(cald);

        // 5、开始发起请求
        DisableCertificateAuthorityResponse response;
        try {
            response = ccmClient.disableCertificateAuthority(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6、获取响应消息,禁用成功后,无响应内容,返回的状态码为204
    }
}
```

```
        System.out.println(response.getHttpStatusCode());
    }
}
```

3.3.2.4 启用 CA

启用私有CA，将“已禁用”状态的CA启用，CA状态置为“已激活”状态。

相关参数详情请参见[启用CA参数说明](#)。

```
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;

/**
 * 启用CA，将禁用状态下的CA启用为激活状态 */
public class EnableCertificateAuthorityExample {
    /**
     * 基础认证信息：
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储在较大安全风险，建议在配置文件或环境变量中密文存放，使用时解密，确保安全；
     * 本示例ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量
     HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息，PCA为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2.初始化SDK，传入认证信息及CCM服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
            .withCredential(auth)
            .withEndpoint(CCM_ENDPOINT).build();

        // 3、构造请求参数
        // 需要启用的CA的ID
        String cald = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";

        // 4、构造请求体
        EnableCertificateAuthorityRequest request = new EnableCertificateAuthorityRequest()
            .withCald(cald);

        // 5、开始发起请求
        EnableCertificateAuthorityResponse response;
        try {
            response = ccmClient.enableCertificateAuthority(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6、获取响应消息,启用成功后，无响应内容，返回的状态码为204
        System.out.println(response.getHttpStatusCode());
    }
}
```

3.3.2.5 导出 CA 证书

导出私有CA证书，包含证书体与证书链。

📖 说明

根证书为自签名证书，其证书链为null。

相关参数详情请参见[导出CA证书参数说明](#)。

```
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;

/**
 * 导出CA证书，包含证书体与证书链。根证书为自签名证书，其证书链为null
 */
public class ExportCertificateAuthorityExample {
    /**
     * 基础认证信息：
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储在较大安全风险，建议在配置文件或环境变量中密文存放，使用时解密，确保安全；
     * 本示例ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量
     HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息，PCA为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2.初始化SDK，传入认证信息及CCM服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
            .withCredential(auth)
            .withEndpoint(CCM_ENDPOINT).build();

        // 3、构造请求参数
        // 需要导出的CA证书的ID
        String cald = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";

        // 4、构造请求体
        ExportCertificateAuthorityCertificateRequest request = new
        ExportCertificateAuthorityCertificateRequest()
            .withCald(cald);

        // 5、开始发起请求
        ExportCertificateAuthorityCertificateResponse response;
        try {
            response = ccmClient.exportCertificateAuthorityCertificate(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6、获取响应消息
        // (1) 获取证书体，pem格式
        String caCertificate = response.getCertificate();
        // (2) 获取证书链，pem格式
        String caCertificateChain = response.getCertificateChain();
    }
}
```

```
        System.out.println(response);
    }
}
```

3.3.2.6 取消删除 CA

取消私有CA的计划删除，将“计划删除”状态的私有CA恢复为删除前的状态。

相关参数详情请参见[取消删除CA参数说明](#)。

```
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;

/**
 * 取消CA的计划删除，将计划删除状态下的CA恢复为禁用状态
 */
public class RestoreCertificateAuthorityExample {
    /**
     * 基础认证信息：
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储在较大安全风险，建议在配置文件或环境变量中密文存放，使用时解密，确保安全；
     * 本示例ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量
     HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息，PCA为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2.初始化SDK，传入认证信息及CCM服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
            .withCredential(auth)
            .withEndpoint(CCM_ENDPOINT).build();

        // 3、构造请求参数
        // 需要操作的CA证书的ID
        String cald = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";

        // 4、构造请求体
        RestoreCertificateAuthorityRequest request = new RestoreCertificateAuthorityRequest()
            .withCald(cald);

        // 5、开始发起请求
        RestoreCertificateAuthorityResponse response;
        try {
            response = ccmClient.restoreCertificateAuthority(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6、获取响应消息,取消计划删除成功后，无响应内容，返回的状态码为204
        System.out.println(response.getHttpStatusCode());
    }
}
```

```
}
```

3.3.2.7 获取 CA 详情

获取私有CA详情，包括私有CA名称、部门名称、类型和状态等信息。

相关参数详情请参考[获取CA详情参数说明](#)。

```
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;

/**
 * 获取CA证书详情
 */
public class ShowCertificateAuthorityExample {
    /**
     * 基础认证信息:
     * - ACCESS_KEY: 华为云账号 Access Key
     * - SECRET_ACCESS_KEY: 华为云账号 Secret Access Key
     * - DOMAIN_ID: 华为云账号 ID
     * - CCM_ENDPOINT: 华为云 CCM 服务 (PCA 属于 CCM 下的微服务) 访问终端地址
     * 认证使用的 ak 和 sk 硬编到代码中或明文存储在较大安全风险，建议在配置文件或环境变量中密文存放，使用时解密，确保安全；
     * 本示例 ak 和 sk 保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量 HUAWEICLOUD_SDK_AK 和 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. 准备访问华为云的认证信息，PCA 为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2. 初始化 SDK，传入认证信息及 CCM 服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
            .withCredential(auth)
            .withEndpoint(CCM_ENDPOINT).build();

        // 3. 构造请求参数
        // 需要查询的 CA 证书的 ID
        String cald = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";

        // 4. 构造请求体
        ShowCertificateAuthorityRequest request = new ShowCertificateAuthorityRequest()
            .withCald(cald);

        // 5. 开始发起请求
        ShowCertificateAuthorityResponse response;
        try {
            response = ccmClient.showCertificateAuthority(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 6. 打印响应消息
        // 获取证书状态，其它属性值的获取，根据其对应的 Getter 函数获取即可
        String caStatus = response.getStatus();

        System.out.println(response);
    }
}
```

```
}
```

3.3.2.8 查询 CA 配额

查询CA配额总数和已使用的CA配额数。

相关参数详情请参见[查询CA配额参数说明](#)。

```
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;

/**
 * 查询CA证书配额
 */
public class ShowCertificateAuthorityQuotaExample {
    /**
     * 基础认证信息:
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储在较大安全风险,建议在配置文件或环境变量中密文存放,使用时解密,确保安全;
     * 本示例ak和sk保存在环境变量中为例,运行本示例前请先在本地环境中设置环境变量
     HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息,PCA为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2.初始化SDK,传入认证信息及CCM服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
            .withCredential(auth)
            .withEndpoint(CCM_ENDPOINT).build();

        // 3、构造请求体
        ShowCertificateAuthorityQuotaRequest request = new ShowCertificateAuthorityQuotaRequest();

        // 4、开始发起请求
        ShowCertificateAuthorityQuotaResponse response;
        try {
            response = ccmClient.showCertificateAuthorityQuota(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }

        // 5、获取配额使用情况
        List<Resources> quotas = response.getQuotas().getResources();
        // 配额总量
        int caQuota = quotas.get(0).getQuota();
        // 已使用的配额
        int used = quotas.get(0).getUsed();

        System.out.println(response);
    }
}
```

```
}
```

3.3.3 私有证书管理代码示例

3.3.3.1 申请证书

请求签发私有证书，需要拥有处于“已激活”状态的私有CA。

相关参数详情请参见[申请证书参数说明](#)。

```
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;

/**
 * 签发私有证书，需要拥有处于激活状态的私有CA
 */
public class createCertificateExample {
    /**
     * 基础认证信息：
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储在较大安全风险，建议在配置文件或环境变量中密文存放，使用时解密，确保安全；
     * 本示例ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息，PCA为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2.初始化SDK，传入认证信息及CCM服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
            .withCredential(auth)
            .withEndpoint(CCM_ENDPOINT).build();

        // 3、构造请求参数
        // (1) 用于签发证书的CA的ID，该CA需要处于激活状态 (ACTIVED)
        String issuerId = "3a02c7f6-d8f5-497e-9f60-18dfd3eeb4e6";
        // (2) 证书密钥算法
        String keyAlgorithm = "RSA2048";
        // (3) 签名哈希算法
        String signatureAlgorithm = "SHA512";

        /**
         * (4) 证书有效期定义
         * - type: 时间类型，可选: "YEAR"、"MONTH"、"DAY"、"HOUR"
         */
    }
}
```

```
* - value: 对应的值
*/
Validity validity = new Validity();
validity.setType("MONTH");
validity.setValue(2);

/*
 * ( 5 ) 定义CA证书的唯一标识信息
 * - organization: 组织名称
 * - organizationalUnit: 部门名称
 * - country: 国家缩写, 仅限两个字符, 如中国-CN
 * - state: 省市名称
 * - locality: 城市名称
 * - commonName: CA名称 ( 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 ) 密钥用法, 服务器证书通常只赋予keyAgreement与digitalSignature, 为可选值
 * - digitalSignature : 数字签名;
 * - nonRepudiation : 不可抵赖;
 * - keyEncipherment : 密钥用于加密封钥数据;
 * - dataEncipherment : 用于加密数据;
 * - keyAgreement : 密钥协商;
 * - keyCertSign : 签发证书;
 * - cRLSign : 签发吊销列表;
 * - encipherOnly : 仅用于加密;
 * - decipherOnly : 仅用于解密。
 */
List<String> keyUsages = new ArrayList<>();
keyUsages.add("digitalSignature");
keyUsages.add("keyAgreement");

/*
 * ( 7 ) 主体备用名称: 暂时支持DNS、IP、URI与EMAIL, 为可选值
 * SubjectAlternativeName:
 *   type: 类型
 *   value: 对应值
 */
List<SubjectAlternativeName> subjectAlternativeName = new ArrayList<>();
// a、添加备用DNS
SubjectAlternativeName alterNameDNS = new SubjectAlternativeName();
alterNameDNS.setType("DNS");
alterNameDNS.setValue("*.example.com");
subjectAlternativeName.add(alterNameDNS);
// b、添加备用IP
SubjectAlternativeName alterNameIP = new SubjectAlternativeName();
alterNameIP.setType("IP");
alterNameIP.setValue("127.0.0.1");
subjectAlternativeName.add(alterNameIP);
// b、添加备用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 ) 请求体各属性赋值
// 各属性的取值约束, 请查阅: https://support.huaweicloud.com/api-cm/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、构造请求体
CreateCertificateRequest request = new CreateCertificateRequest()
    .withBody(requestBody);

// 5、开始发起请求
CreateCertificateResponse response;
try {
    response = ccmClient.createCertificate(request);
} catch (Exception e) {
    throw new RuntimeException(e.getMessage());
}

// 6、获取响应消息
String certId = response.getCertificateId();
System.out.println(certId);
}
}
```

3.3.3.2 删除证书

删除私有证书，将会立即删除，不支持计划删除。

相关参数详情请参见[删除证书参数说明](#)。

```
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;

/**
 * 删除证书，将会立即删除，不支持计划删除
 */
public class DeleteCertificateExample {
    /**
     * 基础认证信息：
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储在较大安全风险，建议在配置文件或环境变量中密文存放，使用
     用时解密，确保安全；
     * 本示例ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量
     HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息，PCA为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2.初始化SDK，传入认证信息及CCM服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
```

```
        .withCredential(auth)
        .withEndpoint(CCM_ENDPOINT).build();

// 3、构造请求参数
// (1) 需要删除的私有证书的ID
String certId = "5554a381-af92-4336-a943-811396c87616";

// 4、构造请求体
DeleteCertificateRequest request = new DeleteCertificateRequest().withCertificateId(certId);

// 5、开始发起请求
DeleteCertificateResponse response;
try {
    response = ccmClient.deleteCertificate(request);
} catch (Exception e) {
    throw new RuntimeException(e.getMessage());
}

// 6、获取响应消息,删除成功后,无响应内容,返回的状态码为204
System.out.println(response.getHttpStatusCode());
}
}
```

3.3.3.3 导出证书

导出私有证书, 包含证书体与证书链。可根据具体需求导出对应格式的证书。

相关参数详情请参见[导出证书参数说明](#)。

```
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;

/**
 * 导出私有证书, 包含证书体与证书链。可根据具体需求导出对应格式的证书
 */
public class ExportCertificateExample {
    /**
     * 基础认证信息:
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务(PCA属于CCM下的微服务)访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储在较大安全风险, 建议在配置文件或环境变量中密文存放, 使用时解密, 确保安全;
     * 本示例ak和sk保存在环境变量中为例, 运行本示例前请先在本地环境中设置环境变量
     HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息, PCA为全局服务
        final GlobalCredentials auth = new GlobalCredentials()
            .withAk(ACCESS_KEY)
            .withSk(SECRET_ACCESS_KEY)
            .withDomainId(DOMAIN_ID);

        // 2.初始化SDK, 传入认证信息及CCM服务的访问终端地址
        final CcmClient ccmClient = CcmClient.newBuilder()
            .withCredential(auth)
            .withEndpoint(CCM_ENDPOINT).build();

        // 3、构造请求参数
```

```
// (1) 需要导出的终端实体证书ID
String certId = "5554a381-af92-4336-a943-811396c87616";

/*
(2) 定义导出格式 (SDK仅支持非压缩下的)
- isCompressed: 是否压缩, 类型为String, 可选值true、false, SDK仅支持false
- type: 导出形式, SDK调用目前仅支持以下形式
  APACHE: apache服务器推荐使用此参数;
  NGINX: nginx服务器推荐使用此参数;
  OTHER: 下载PEM格式证书, 推荐使用此参数。
*/
ExportCertificateRequestBody requestBody = new ExportCertificateRequestBody();
requestBody.setType("NGINX");
requestBody.setIsCompressed("false");

// 4、构造请求体
ExportCertificateRequest request = new ExportCertificateRequest()
    .withCertificateId(certId)
    .withBody(requestBody);

// 5、开始发起请求
ExportCertificateResponse response;
try {
    response = ccmClient.exportCertificate(request);
} catch (Exception e) {
    throw new RuntimeException(e.getMessage());
}

// 6、获取响应消息
// (1) 获取证书体, pem格式
String certificate = response.getCertificate();
// (2) 获取证书链, pem格式
String certificateChain = response.getCertificateChain();
System.out.println(response);
}
}
```

3.3.3.4 吊销证书

吊销证书后, 将清除该证书所有的记录, 包括私有CA的记录, 且无法恢复, 请谨慎操作。

吊销私有证书相关参数详情请参见[吊销证书参数说明](#)。

```
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;

/**
 * 吊销私有证书
 */
public class RevokeCertificateExample {
    /**
     * 基础认证信息:
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - DOMAIN_ID: 华为云账号ID
     * - CCM_ENDPOINT: 华为云CCM服务 (PCA属于CCM下的微服务) 访问终端地址
     * 认证使用的ak和sk硬编到代码中或明文存储在较大安全风险, 建议在配置文件或环境变量中密文存放, 使用时解密, 确保安全;
     * 本示例ak和sk保存在环境变量中为例, 运行本示例前请先在本地环境中设置环境变量
     HUAWEICLOUD_SDK_AK和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.准备访问华为云的认证信息，PCA为全局服务
    final GlobalCredentials auth = new GlobalCredentials()
        .withAk(ACCESS_KEY)
        .withSk(SECRET_ACCESS_KEY)
        .withDomainId(DOMAIN_ID);

    // 2.初始化SDK，传入认证信息及CCM服务的访问终端地址
    final CcmClient ccmClient = CcmClient.newBuilder()
        .withCredential(auth)
        .withEndpoint(CCM_ENDPOINT).build();

    // 3、构造请求参数
    // (1)需要吊销的终端实体证书ID
    String certId = "5554a381-af92-4336-a943-811396c87616";

    // (2)填写吊销原因（枚举值，请参考API文档查询对应参数），对应的吊销代码将会被写入吊销列表中
    // 默认值：UNSPECIFIED
    RevokeCertificateRequestBody requestBody = new RevokeCertificateRequestBody();
    requestBody.setReason("UNSPECIFIED");

    // 4、构造请求体
    RevokeCertificateRequest request = new RevokeCertificateRequest()
        .withCertificateId(certId)
        .withBody(requestBody);

    // 5、开始发起请求
    RevokeCertificateResponse response;
    try {
        response = ccmClient.revokeCertificate(request);
    } catch (Exception e) {
        throw new RuntimeException(e.getMessage());
    }

    // 6、获取响应消息,吊销成功后，无响应内容，返回的状态码为204
    System.out.println(response.getHttpStatusCode());
}
}
```

3.4 企业内网身份认证体系建立

应用场景

企业或组织可以通过云证书管理服务（Cloud Certificate Manager，CCM）提供的私有证书管理服务（Private Certificate Authority，PCA）建立自己完整的CA层次体系，实现在内部签发和管理自签名私有证书。

本文介绍从私有证书管理服务申请私有证书到部署私有证书的操作流程。

背景知识

建立组织内部完整的CA层次体系称为自建PKI体系。

在自建PKI体系时，需要根据使用场景提前设计好结构，以便后续管理：

- CA层次的选择
CA层次决定了创建各级从属CA时所需设置的路径长度，以限制其向下签发从属CA的能力，同时影响证书链的长度。

恰当的CA层次设计，有利于CA的管理。PCA服务支持最多可创建七层CA结构，不同CA结构的区别请参见[私有CA层次结构设计](#)

- 证书的轮换

证书有效期到期前需要进行新旧证书的替换，避免因证书过期导致业务中断，新旧证书替换的过程即为证书的轮换。

证书有效期的长短决定了证书轮换的周期，合适的证书有效期设置可降低密钥材料泄露的风险和减少证书轮换的成本，关于PCA有效期设置可参考[PCA证书有效期](#)。

- 证书吊销管理

当证书出现密钥泄漏或者因某些因素不再被使用时，需要将其吊销。因此在创建CA时，需要开启证书吊销列表，同时在自身业务的检验证书环节中要检验证书是否已被吊销。

步骤一：创建私有根 CA 或从属 CA

根CA和从属CA皆通过以下操作创建，创建私有CA详细操作说明请参见[创建私有CA](#)

步骤1 登录[管理控制台](#)。

步骤2 单击页面左上方的 ，选择“安全与合规 > 云证书管理服务”，并在左侧导航栏选择“私有证书管理 > 私有CA”进入私有CA管理界面。

步骤3 在私有CA列表左上角，单击“创建CA”，进入创建CA界面。

步骤4 配置私有CA信息。

1. 配置基本信息。



基本信息

* CA类型 根CA 创建根CA，用于建立新的CA层次结构。
 从属CA 创建从属CA，用于在现有的CA层次结构中增加新的层次。

* 密钥算法 RSA2048

* 签名哈希算法 SHA256

* 有效期 年

到期时间: 2032/11/09 14:42:29 GMT+08:00

2. 配置证书唯一标识名称（Distinguished Name，DN）信息。



证书唯一标识名称 (DN)

* CA名称 (CN) PCA TEST ROOT G0 * 国家和地区 CN

* 省市 Guangdong * 城市 Shenzhen

* 公司名称 (O) Huawei Technologies Co., Ltd. * 部门名称 (OU) IT

3. 配置证书吊销信息。



步骤5 单击“下一步”，进入确认信息页面。

步骤6 确认信息以及价格无误后，单击“确认并创建”，完成创建私有CA操作。

----结束

步骤二：使用内部私有 CA 激活从属 CA

从属CA创建后需要进行激活，以下的激活操作以内部私有CA激活两层CA结构的从属CA为例，使用外部私有CA激活从属CA的操作步骤请参见[激活私有CA](#)。

步骤1 登录[管理控制台](#)。

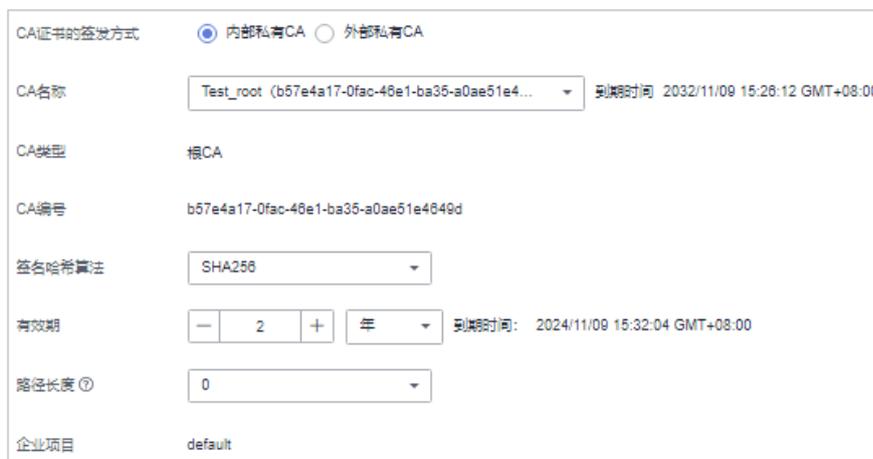
步骤2 单击页面左上方的 ，选择“安全与合规 > 云证书管理服务”，并在左侧导航栏选择“私有证书管理 > 私有CA”进入私有CA管理界面。

步骤3 在待激活的私有从属CA所在行的“操作”列，单击“激活”，系统从右面弹出激活CA详细页面，如[图3-6](#)所示，填写激活CA相关信息。

说明

路径长度根据设计的CA层次结构进行设置，不同CA结构的路径长度设置请参见[私有CA层次结构设计](#)

图 3-6 内部私有 CA



步骤4 确认填写的信息无误后，单击“确定”。

----结束

步骤三：创建私有证书

创建私有证书相关操作详细说明请参见[申请私有证书](#)

步骤1 登录[管理控制台](#)。

步骤2 单击页面左上方的☰，选择“安全与合规 > 云证书管理服务”，并在左侧导航栏选择“私有证书管理 > 私有证书”进入私有证书管理界面。

步骤3 在私有证书列表的左上角，单击“申请证书”，进入申请证书界面。

步骤4 填写申请证书的相关信息。

图 3-7 申请证书-系统生成文件

The screenshot shows the 'System-generated file' tab of the certificate application interface. It is divided into two main sections: 'Certificate Configuration' and 'Select Issuing CA'.

Certificate Configuration:

- 证书配置:** Certificate name (CN) is 'Authentication'.
- 高级配置:** Includes options for '密钥算法' (RSA2048), '签名哈希算法' (SHA256), '密钥用法' (digitalSignature, nonRepudiation), and '增强型密钥用法' (Service Identity, Client Identity).
- 自定义扩展字段:** A text input field for custom extensions.
- 配置证书AltName信息:** A table with one entry: '1' with type 'IP address' and an empty value field. A '+ 添加' button is present below.

Select Issuing CA:

- CA名称 (CN):** Test_subordinate (31c80865-535c-45f3-afe0-3f3...)
- 到期时间:** 2023/11/09 15:52:34 GMT+08:00
- 类型:** 从属CA
- CA编号:** 31c80865-535c-45f3-afe0-3f315a459ec7
- 有效期:** 1 year.

步骤5 确认信息以及价格无误后，单击“确定”。

申请成功后，系统将返回到私有证书页面，在页面右上角弹出“申请证书xxx成功！”，则说明私有证书申请成功。

----结束

步骤四：信任根 CA

在安装私有证书之前，需要根据实际验证需求将根CA加入客户端或服务器受信任的根证书颁发机构中。

- **单向验证**

当服务端无需校验客户端的证书身份时（互联网上大部分公开的网站不校验客户端证书），为了使得客户端信任服务端证书，需要将服务端证书的根CA加入到客户端受信任的根证书颁发机构中。

- **双向验证**

当服务端与客户端皆需校验对方的证书时，需要双方将对方的根CA加入到自己的受信任的根证书颁发机构中。

将私有根CA证书下载至本地，详细操作请参见[导出私有CA证书](#)，获得“根CA名称_certificate.pem”根CA证书文件。

根据不同的操作系统选择以下方式，将根CA加入受信任的根证书颁发机构中：

说明

以信任根CA“PCA TEST ROOT GO”为例。

- **Windows系统**

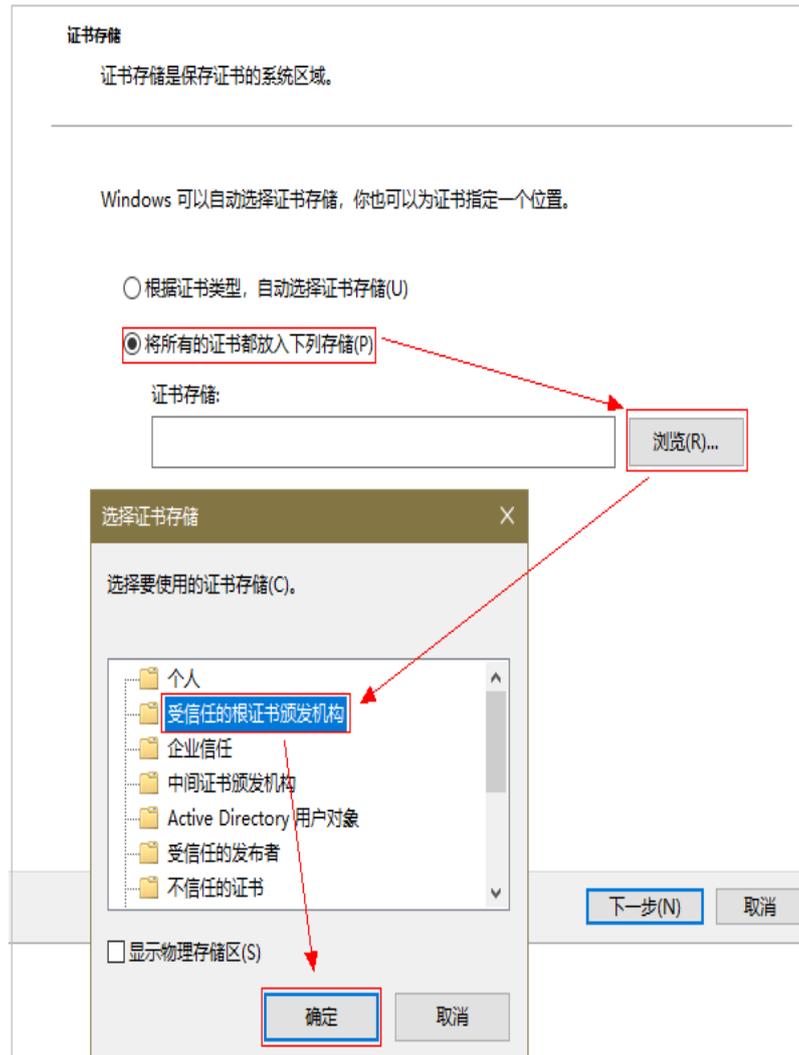
- a. 将根CA证书文件后缀由“.pem”改为“.crt”，双击证书文件，根CA证书信息显示该根证书不受信任。

图 3-8 根 CA 不受信任



- b. 单击“安装证书”，根据使用场景选择证书存储位置，单击“下一步”。
- c. 选择“将所有证书都放入下列存储（P）”，单击“浏览”，选择“受信任的根证书颁发机构”，单击“确定”，如[图 存储根证书](#)所示。

图 3-9 存储根证书



- d. 单击“下一步”，再单击“确定”，会有弹窗提示“Windows将信任该私有根CA证书颁发的所有证书”，单击“是”。
- e. 双击根CA证书文件，此时根CA证书信息显示系统已信任该根CA证书，表示根CA加入受信任的根证书颁发机构成功。

图 3-10 信任根 CA



- **Linux系统**

不同版本的Linux操作系统中，根CA证书存放路径以及操作方法不一致，需要您根据实际情况进行操作。以下操作以CentOS 6版本的Linux系统为例：

- 将根CA证书文件复制到“/home/”路径下。
- 当服务器未安装“ca-certificates”时，使用如下命令安装“ca-certificates”。

yum install ca-certificates

- 使用如下命令将根CA证书复制到“/etc/pki/ca-trust/source/anchors/”路径下。

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

- 使用如下命令将根CA证书添加到根证书信任文件中。

update-ca-trust extract

- 使用如下命令查看根CA证书是否添加成功信息，查看到新添加的根CA证书信息表示根CA加入受信任的根证书颁发机构成功，如[图 新添加的根CA证书](#)所示。

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

图 3-11 新添加的根 CA 证书



说明

当openssl版本过低时，可能导致配置无法生效，可尝试使用yum update openssl -y 命令更新openssl版本。

• macOS系统

- a. 打开mac的启动台，选择“钥匙串”。
- b. 输入密码登录到“钥匙串”。
- c. 将需要信任的根CA证书文件拖入钥匙串中，此时拖入的根CA证书会显示不被系统信任。
- d. 选中根CA证书文件，单击鼠标右键选择“显示简介”。
- e. 选择“信任>使用此证书时”，选择“始终信任”，单击“关闭”。
- f. 输入密码使信任根CA证书配置生效。
- g. 在“钥匙串”主页查看根CA证书，证书显示被信任表示根CA加入受信任的根证书颁发机构成功。

步骤五：安装私有证书

在客户端安装私有证书（以在Windows系统部署证书为例）

步骤1 登录[管理控制台](#)。

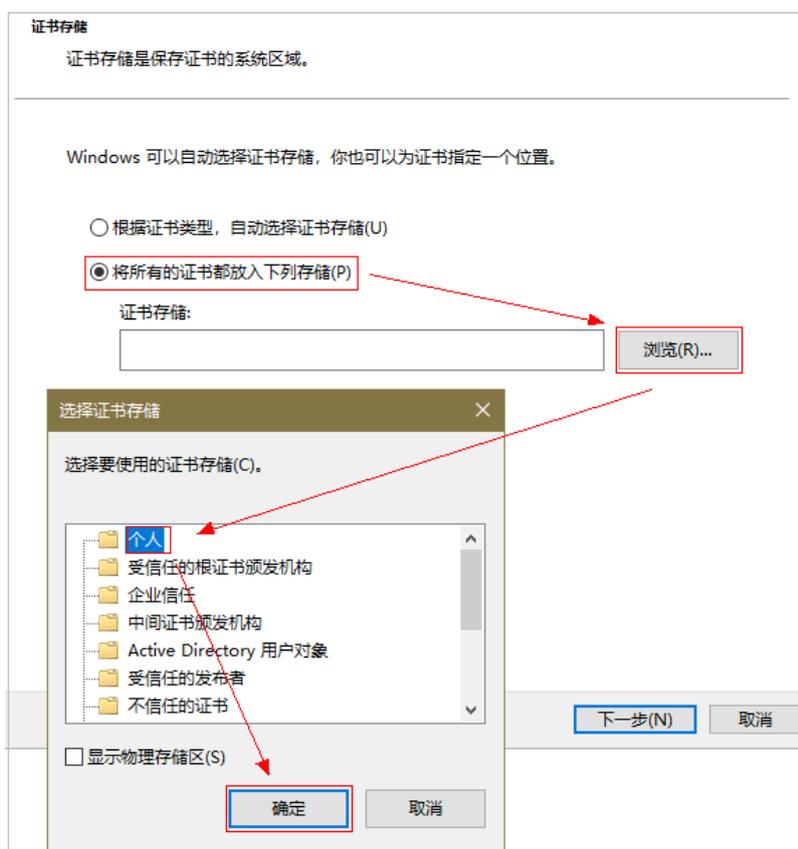
步骤2 单击页面左上方的 ，选择“安全与合规 > 云证书管理服务”，并在左侧导航栏选择“私有证书管理 > 私有证书”进入私有证书管理界面。

步骤3 在目标证书所在行的“操作”列，单击“下载”，进入下载证书页面。

步骤4 选择证书下载格式为“IIS”，单击“下载证书”。

- 步骤5** 解压下载的证书文件压缩包“client_iis.zip”，解压后，获得证书文件“server.pfx”和私钥密码文件“keystorePass.txt”。
- 步骤6** 双击证书文件“server.pfx”，根据使用场景选择证书存储位置，单击“下一步”。
- 步骤7** 确认要导入的证书文件名，单击“下一步”。
- 步骤8** 输入从私钥密码文件“keystorePass.txt”中获取的密码，单击“下一步”。
- 步骤9** 选择“将所有的证书放入下列存储（P）”，单击“浏览”，选择“个人”，单击“确定”如图 存储私有证书所示。

图 3-12 存储私有证书



- 步骤10** 单击“下一步”，单击“完成”，出现弹窗提示证书“导入成功”，证书安装成功。

----结束

在服务器安装私有证书

在服务器安装私有证书的操作与国际标准SSL证书安装操作相同，请根据服务器类型选择相应的安装参考示例：

- 在Tomcat上安装SSL证书：操作示例详情请参见[在Tomcat服务器上安装SSL证书](#)。
- 在Nginx上安装SSL证书：操作示例详情请参见[在Nginx服务器上安装SSL证书](#)。
- 在Apache上安装SSL证书：操作示例详情请参见[在Apache服务器上安装SSL证书](#)。
- 在IIS上安装SSL证书：操作示例详情请参见[在IIS服务器上安装SSL证书](#)。

- 在Weblogic上安装SSL证书：操作示例详情请参见[在Weblogic服务器上安装SSL证书](#)。
- 在Resin上安装SSL证书：操作示例详情参见[在Resin服务器上安装SSL证书](#)。